

**Designing for Remixing:  
Supporting an Online Community of Amateur Creators**

by

Andrés Monroy-Hernández

S.M., Media Arts and Sciences, Massachusetts Institute of Technology (2007)  
B.S., Electronic Systems Engineering, Tecnológico de Monterrey (2001)

Submitted to the Program in Media Arts and Sciences,  
School of Architecture and Planning,  
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

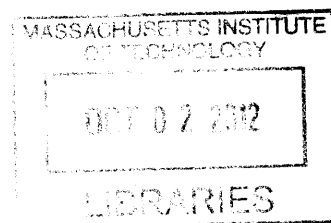
© This work is licensed under a Creative Commons Attribution 3.0 Unported License.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Author \_\_\_\_\_  
Program in Media Arts and Sciences  
August 20, 2012

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

Accepted by \_\_\_\_\_  
Patricia Maes  
Associate Academic Head  
Program in Media Arts and Sciences





**Designing for Remixing:  
Supporting an Online Community of Amateur Creators**

by  
Andrés Monroy-Hernández

Submitted to the Program in Media Arts and Sciences,  
School of Architecture and Planning,  
on August 20, 2012, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy in Media Arts and Sciences

**Abstract**

This work describes a framework for the design and study of an online community of amateur creators. I focus on remixing as the lens to understand the contexts and processes of creative expression as it is fostered within social media environments. I am motivated by three broad questions:

- 1) Process: how do people remix and what is the role of remixing in cultural production and social learning?
- 2) Conditions: what kind of attributes influence people's remixing practices?
- 3) Attitudes: what are people's attitudes toward remixing?

As part of this work, I conceived, developed and studied the Scratch Online Community: a website where young people share and remix their own video games and animations, as well as those of their peers. In five years, the community has grown to more than one million registered members and two million community-contributed projects.

In the spirit of the theme of this work, this dissertation remixes several articles and blog posts written by myself or in collaboration with others. Wherever possible, the sources of the material are noted.

Thesis Supervisor: Mitchel Resnick

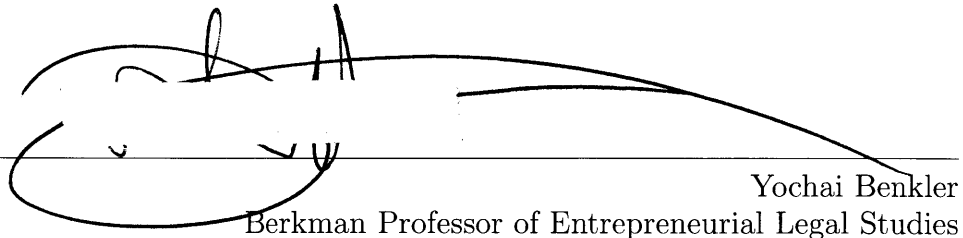
Title: LEGO Papert Professor of Learning Research, Program in Media Arts and Sciences

**Designing for Remixing:  
Supporting an Online Community of Amateur Creators**

by  
Andrés Monroy-Hernández

The following people will serve as readers for this thesis:

Thesis Reader



Yochai Benkler  
Berkman Professor of Entrepreneurial Legal Studies  
Harvard University

Thesis Reader

Robert C. Miller  
Associate Professor of Electrical Engineering and Computer Science  
MIT Computer Science and Artificial Intelligence Laboratory

# Acknowledgments

First I want to thank my advisor, Mitchel Resnick, for the opportunity to be part of this adventure, and for welcoming me into the Lifelong Kindergarten family. His wisdom, patience, and constant support made my time at the Media Lab one of the greatest experiences of my life.

I also want to thank my dissertation and my qualifying exams committee for helping me expand my disciplinary horizons. Yochai Benkler's cooperation group at the Berkman Center opened my eyes to a wide range of disciplines. Rob Miller's group at CSAIL and his students helped me venture into the field of Social Computing. Tim Berners-Lee helped me understand the spirit behind the Semantic Web.

Much of the work on this document was only possible because of the ideas, contributions, and support from several collaborators including Benjamin Mako Hill, Cecilia Aragon, danah boyd, Diana Aragon, Jazmin Gonzalez-Rivero, Jeff Nickerson, Kristina Olson, and Oshani Seneviratne.

The Scratch project is a group effort from the staff, the graduate, and undergraduate students at the Lifelong Kindergarten group, including Amos Blanton, Amon Millner, Brian Silverman, Chinua Shaw, Chris Garrity, Claudia Urrea, Eric Rosenbaum, Evelyn Eastmond, Han Xu, Gaia Carini, Jay Silver, John Maloney, Karen Brennan, Lance Vikaros, Leo Burd, Lis Sylvan, Michelle Chung, Mitchel Resnick, Natalie Rusk, Nick Bushak, Oren Zuckerman, Paula Bontá, Paul Medlock-Walton, Rachel Garber, Ricarose Roque, Rita Chen, Sayamindu Dasgupta, Stephanie Gayle, Tamara Stern, and Ubong Ukoh.

I also want to acknowledge the technical support I received from Anupom Syam, Anant Seethalakshmi, Ashok Kumar, Kemie Guaida, and Vladimir Vuksan; and the community-management help from Mark Goff and Franchette Vilorio.

Last but not least, I want to thank the four most important people in my life: Kristina, Andrea, mamá y papá.

# Contents

<b>Abstract</b>	<b>3</b>
<b>Acknowledgments</b>	<b>5</b>
<b>Front Matter</b>	<b>6</b>
Contents . . . . .	6
List of Figures . . . . .	9
List of Tables . . . . .	12
<b>1 Introduction</b>	<b>13</b>
1.1 Structure and Contributions of this Work . . . . .	15
1.2 Economic and Cultural Production . . . . .	18
1.3 Software Engineering . . . . .	19
1.4 Learning and New Media Literacy . . . . .	20
1.5 Ethical and Legal Challenges . . . . .	21
1.6 Social Computing System Design . . . . .	22
<b>2 Remixing Systems</b>	<b>24</b>
2.1 Video . . . . .	24
2.2 Images . . . . .	27
2.3 Audio . . . . .	29
2.4 Status Updates . . . . .	31
2.5 Source Code . . . . .	32
2.6 Programmable Media . . . . .	33
2.7 Summary . . . . .	36
<b>3 The Scratch Online Community</b>	<b>37</b>
3.1 Infrastructure Development . . . . .	38
3.1.1 Ideation . . . . .	38
3.1.2 User Experience . . . . .	40
3.1.3 Architecture . . . . .	45
3.1.4 Sharing from the Desktop . . . . .	53
3.1.5 Scale . . . . .	54
3.2 Participation Patterns . . . . .	60
3.2.1 Project making . . . . .	65

3.2.2	Interactions . . . . .	72
3.3	Discussion . . . . .	72
3.3.1	Learning through Online Community . . . . .	77
3.3.2	Sharing and collaboration . . . . .	78
<b>4</b>	<b>Process of remixing</b>	<b>80</b>
4.1	Case Studies . . . . .	81
4.1.1	Mesh Inc’s collaboration . . . . .	81
4.1.2	Jumping Monkey’s ripple effect . . . . .	83
4.1.3	Galaxyman’s “media franchise” . . . . .	84
4.2	Taxonomy . . . . .	88
4.2.1	Originality . . . . .	88
4.2.2	Generativity . . . . .	91
4.2.3	Measures . . . . .	94
<b>5</b>	<b>Conditions for Remixing</b>	<b>100</b>
5.1	System Attributes . . . . .	100
5.1.1	Modularity . . . . .	101
5.1.2	Attributability . . . . .	104
5.1.3	Openness . . . . .	104
5.2	Content Attributes . . . . .	105
5.2.1	Generativity . . . . .	106
5.2.2	Originality . . . . .	107
5.2.3	Results . . . . .	113
5.2.4	Limitations . . . . .	116
5.3	Discussion . . . . .	117
<b>6</b>	<b>Attitudes toward Remixing</b>	<b>119</b>
6.1	Study 1: How do people respond to remixing? . . . . .	119
6.1.1	Procedure . . . . .	120
6.1.2	Results . . . . .	120
6.1.3	Discussion . . . . .	121
6.1.4	Implications for design . . . . .	121
6.2	Study 2: When do creators accuse remixers of plagiarism? . . . . .	121
6.2.1	Procedure . . . . .	123
6.2.2	Results . . . . .	124
6.2.3	Discussion . . . . .	126
6.2.4	Implications for design . . . . .	127
6.3	Study 3: Are plagiarism complaints more common when remixes are more similar? . . . . .	127
6.3.1	Procedure . . . . .	128
6.3.2	Results . . . . .	128
6.3.3	Discussion . . . . .	128
6.3.4	Implication for design . . . . .	128
6.4	Study 4: Human and Machine Attribution . . . . .	129

6.4.1	Study 4a: Automatic Attribution . . . . .	130
6.4.2	Study 4b: Manual Crediting . . . . .	131
6.5	Study 5: Interviews with participants . . . . .	134
6.5.1	Methodology . . . . .	134
6.6	Discussion . . . . .	139
<b>7</b>	<b>Conclusions</b>	<b>142</b>
7.1	Summary and Contributions . . . . .	143
7.2	Design Implications . . . . .	144
7.3	Future Research . . . . .	147
7.4	Epilogue: MusicalMoon . . . . .	148
<b>Appendix A Entity Relationship Diagrams</b>		<b>150</b>
<b>Appendix B Project Attribute Tables</b>		<b>154</b>
<b>Bibliography</b>		<b>169</b>



# List of Figures

1-1	Popularity of the term “remix” . . . . .	14
1-2	The home page of the Scratch Online Community website . . . . .	16
1-3	Scratch programming environment . . . . .	17
1-4	Image macro mocking the recursive nature of remixing . . . . .	19
2-1	Download button and license statement of a YouTube video . . . . .	25
2-2	Remix button on YouTube . . . . .	25
2-3	Credit-giving on Vimeo . . . . .	26
2-4	Jumpcut’s web-based editor . . . . .	27
2-5	License selection on Flickr . . . . .	28
2-6	Distribution of Creative Commons licenses usage on Flickr . . . . .	28
2-7	Selecting a Creative Commons license in DeviantArt. . . . .	29
2-8	Sharing options in DeviantArt. . . . .	29
2-9	Front page and online editor of OPENSTUDIO. . . . .	30
2-10	ccMixer showing the Creative Commons license . . . . .	30
2-11	IndabaMusic displaying the number of remixes of a music file. . . . .	31
2-12	The “retweet” button, automating attribution on Twitter. . . . .	31
2-13	Sharing a friend’s post on Facebook, automatically attributed. . . . .	32
2-14	Sourceforge project page. . . . .	33
2-15	Forking graph on Github. . . . .	33
2-16	YTMND’s front page. . . . .	34
2-17	Newgrounds’ front page. . . . .	35
2-18	MyGame front page and page showing game edition (template). . . . .	35
2-19	Moose Crossing: play and scripting environments. . . . .	35
3-1	Scratch website (April 2006) . . . . .	39
3-2	Mockup of Scratch website page based on Flickr (May 2006). . . . .	41
3-3	Mockup of interface for uploading a Scratch project (May 2006). . . . .	42
3-4	Site map outlining all the sections of the website (May 2006). . . . .	43
3-5	Whiteboard outlining the elements of the website (June 2006). . . . .	44
3-6	First prototype of Scratch website (June 2006) . . . . .	45
3-7	Front page. . . . .	47
3-8	Project page. . . . .	48
3-9	Profile page (My Stuff). . . . .	49
3-10	Gallery page. . . . .	50

3-11	Project-browsing page. . . . .	51
3-12	Three remixing visualizations. . . . .	52
3-13	Initial architecture of ScratchR. . . . .	52
3-14	Initial Entity Relationship Diagram of the ScratchR database (ca. 2007). . . . .	53
3-15	Five years of monthly pageview counts. . . . .	54
3-16	Scaled-up Architecture of ScratchR. . . . .	55
3-17	Flag word cloud . . . . .	56
3-18	Community members can flag projects as “inappropriate”. . . . .	57
3-19	Administrators can censor, demote, protect, or feature a project. . . . .	58
3-20	Administrators’ control panel to deal with community-generated flags. . . . .	59
3-21	World map showing cities with most number of Scratch visitors. . . . .	61
3-22	Distribution of ages of project sharers. . . . .	61
3-23	Distribution of projects by account age. . . . .	62
3-24	Project creator’s age and account age monthly mean. . . . .	63
3-25	User age and account age monthly median (project creators only). . . . .	64
3-26	Projects and creators per month. . . . .	65
3-27	Example of a Scratch script in visual and textual form. . . . .	66
3-28	Script cloud . . . . .	67
3-29	Histogram of block usage . . . . .	68
3-30	Image cloud . . . . .	70
3-31	Project complexity and user’s and account’s age. . . . .	71
3-32	Monthly mean number of interactions per project . . . . .	74
3-33	Scratch users contribute to and learn from the online community. . . . .	79
4-1	BeepBop’s sprites project. . . . .	82
4-2	Remixing events started by “Jumping Monkey” . . . . .	85
4-3	Chocolate Bar episodic series. . . . .	87
4-4	Remixing taxonomy based on originality and generativity. . . . .	88
4-5	Inspirational remix. . . . .	89
4-6	Example of incremental remix. . . . .	90
4-7	Example of component-based remixing. . . . .	91
4-8	Sequence of iterations of a crowd-based remix. . . . .	93
4-9	Distribution of derivativeness. . . . .	95
4-10	The most remixed project created by a community member. . . . .	97
4-11	Maximum remix levels in remix chains . . . . .	98
4-12	In-degree distribution . . . . .	99
4-13	Map showing remixing connections. . . . .	99
5-1	Sociotechnical system attributes that facilitate remixing system . . . . .	101
5-2	Anatomy of a Scratch project. . . . .	102
5-3	Use of the “textbox” sprite from Scratch Resources. . . . .	103
5-4	Visual representation of the hypotheses promoted in this section . . . . .	109
5-5	Diagram of the relationship between originality and generativity . . . . .	115
6-1	Distribution of reactions to remixing when automatic credit . . . . .	131
6-2	Reactions to remixing when manual credit . . . . .	133

6-3	Word cloud of comments to remixes . . . . .	134
7-1	Pac-Man remix removed after to DMCA take down notice. . . . .	143
7-2	Instance of ScratchR for Portugal. . . . .	145
7-3	Scratch Online Community as seen by a 14-year-old community member. .	146
A-1	User tables . . . . .	151
A-2	Project tables . . . . .	152
A-3	Project tables . . . . .	153

# List of Tables

3.1	Table of project attributes . . . . .	73
5.1	Summary stats for variables used in analysis . . . . .	111
5.2	Regression models for generativity and originality . . . . .	114
6.1	Plagiarism accusations . . . . .	122
6.2	Taxonomy of logistic regression models on accuse.plag . . . . .	124
6.3	Table listing details of interviewees used in Study 2. ( $n = 12$ ) . . . . .	135
B.1	All Projects: <i>Male</i> . . . . .	155
B.2	All Projects: <i>Female</i> . . . . .	156
B.3	Remixes: <i>Male and Female</i> . . . . .	157
B.4	Remixes: <i>Male</i> . . . . .	158
B.5	Remixes: <i>Female</i> . . . . .	159
B.6	De Novo projects: <i>Male and Female</i> . . . . .	160
B.7	De Novo projects: <i>Male</i> . . . . .	161
B.8	De Novo projects: <i>Female</i> . . . . .	162
B.9	Only Collaborative Remixes: <i>Male and Female</i> . . . . .	163
B.10	Only Collaborative Remixes: <i>Male</i> . . . . .	164
B.11	Only Collaborative Remixes: <i>Female</i> . . . . .	165
B.12	Only Versioning Remixes: <i>Male and Female</i> . . . . .	166
B.13	Only Versioning Remixes: <i>Male</i> . . . . .	167
B.14	Only Versioning Remixes: <i>Female</i> . . . . .	168

# Chapter 1

## Introduction

*We are like dwarfs standing upon the shoulders of giants, and so able to see more and see farther than the ancients.*

—Bernard of Chartres, circa 1130

*A dwarf on a giant's shoulders sees farther of the two.*

—George Herbert, 1651

*If I have seen further it is by standing on the shoulders of giants.*

—Isaac Newton, 1676

Digital networked technologies are challenging the romantic notions of what it means to be a creator. Today's creators *sample*, *remake*, *fork*, *mash-up*, *collage*, and *appropriate*. They *remix*. For example, Wikipedia editors frequently *tweak* existing articles, computer programmers on GitHub often *fork* existing source code repositories, social media users on Twitter or Facebook regularly *retweet* or share other people's status updates, and even designers of physical objects on Instructables reuse existing blueprints or build on others' work. The digital artifacts of the social web are often the result of remixing.

Today the term remixing refers to the act of creating something new based on existing materials. Even before digital technology started to be used for creative expression, "remixing" described the mechanical process of combining physical source materials, something often used in fields like agriculture, chemistry, and manufacturing. For example, one of the first uses of the word remixing that I found<sup>1</sup> was in an 1839 gardening magazine describing the recombination of soils (Loudon, 1839). As digital music machines started to become more common, the term started to appear in this context. For example, the 1966 edition of a popular magazine for high fidelity audio devices described how remixing was originally used to create backups of tapes: "remixing began, historically, as a protective measure," then it became "the final step in making a record" (Davis, 1966). By the late 1970s, remixing was a prominent part of the music scene, and by the end of the century exploded in popularity

---

<sup>1</sup>From corpus of printed materials in English scanned (Michel et al., 2011).

(see Figure 1-1). Presumably, this resulted from the adoption of digital technologies that opened new forms of creative expression.



Figure 1-1: Popularity of the term “remix” based on word frequency (from Google NGram).

Although content reuse predates digital technologies, the advent of these tools has made remixing ubiquitous, more visible, and more derivational. The ubiquitousness of remixing comes from the extensive range of media and communities that have embraced this practice, from video, to text, to music, to code, to photographs, to CAD blueprints, among others. Today, “anybody can remix anything [...] and distribute it globally pretty much instantly” (Ferguson, 2010). In addition, remixing is more visible than before because, as with other activities that leave a digital trace, it can be captured and measured through computational means. For example, YouTube can detect when a video reuses an existing song, and displays this on the video’s web page. Last, remixes are more derivational because computers are, in essence, copying machines that allow creators to reuse verbatim and almost infinitely the bytes of the source materials, something impossible in the analog world. The term remixing is possibly a fad, but what it represents—the concept of digital reuse—has far-reaching and long-lasting implications. The rise of remixing embodies a significant shift in the way cultural and economic production work.

Remixing is not without controversy. As early as 1979, an article in *Billboard* magazine described the tensions around remixing between DJs and music producers: “The sparks started flying at the Disco VI international producers panel when Arista’s Audrey Joseph challenged those producers who won’t let DJs remix their records” (Media, 1979). By the year 2001, the controversy was no longer whether music producers would allow remixing, but how to handle the increased popularity of remixes. The magazine best known for its music popularity charts made the decision to stop counting the popularity of a remix toward the popularity of the song on which it is based on. *Billboard* made this decision because of the case of a particular remix gaining more popularity than the original song. The controversy forced the magazine to issue the following statement:

*[T]he enormous growth of ‘Real’ [the “original” song] largely comes from a re-worked version of the song featuring Ja Rule [the remixer]. The Ja Rule version soon became the track of choice at all formats, so we have now added Ja Rule’s name to the [...] charts. For the current chart week, 85% of the song’s audience comes from the remix version (Pietrolungo, 2001).*

This incident arguably marked the coming of age for remixing. Today, such debates continue and are even more visible.

This work focuses on these challenges and others brought up by remixing, with an emphasis on implications for the design of social, technical, and learning systems. Remixing defies the traditional models and assumptions of authorship and creativity. Its prominence also has deep implications for the design of sociotechnical and learning systems. In this dissertation I focus on the design of a particular system, a website, built to foster creative collaboration through remixing. As the person leading the construction of this website and the community that grew from it, I narrate the “behind the scenes” story from its start until its five-year anniversary. Then, I focus on examining what people did on this system; in particular, I examine how people engage in remixing, what makes remixing thrive, and the attitudes people have toward it.

The empirical setting for this work is the Scratch Online Community, a website (see Figure 1-2) I conceived and developed over the past five years in collaboration with others at the Lifelong Kindergarten research group. The website allows anyone, especially young people between eight and sixteen, to share their animated stories, interactive art, and video games. Participants use the Scratch development environment, a desktop application (see Figure 1-3), to create or remix projects by putting together images, music and sounds with visual programming blocks that control their behavior (Resnick et al., 2009). Scratch and its website enable young people to express themselves creatively, gain new media literacy skills, and learn core computational thinking concepts, all in a community of peers.

## 1.1 Structure and Contributions of this Work

In this introductory chapter, I provide an overview of the remixing phenomenon, its history, and the reasons why it is an area worthy of study. I do this by presenting the theoretical background that frames the discussions and analyses of the rest of this work.

The second chapter presents an overview of the state of the *remix culture* on the social Web. I examine the remixing-related features of popular social computing systems that have influenced directly or indirectly the design of the Scratch Online Community.

The third chapter is a *design brief* of the Scratch Online Community. In that chapter, I answer the question of *what an online remixing community looks like from the inside*. The chapter describes the design of the sociotechnical *infrastructure* of the Scratch website, and how people used it over the course of five years.

The fourth chapter centers on the question of *how people remix*. This chapter examines the *process of remixing* by introducing a remixing taxonomy across two dimensions: originality and generativity. Then I focus on a few case studies that use the taxonomy to examine remixing in different forms of collaborative work.

The fifth chapter concentrates on the question of *what influences remixing*. In this chapter, I investigate what *conditions are conducive to remixing*. First, I reason through what I think were the system attributes that supported remixing in Scratch. Then I present an



[home](#) [projects](#) [galleries](#) [support](#) [forums](#) [about](#) [Language](#)

[Login or Signup](#) for an account

Create and share your own interactive stories, games, music, and art

Check out the 2,664,562 projects from around the world!



To create your own projects:

[Download Scratch](#)

```
when I receive boom...
set whirl effect to 0
change x by pic ▶ in 10 to 30
change whirl effect by 90
wait 0.2 secs
set whirl effect to 0
```

### Featured Projects

[See more](#)

[Screen Writing 3](#)  
by ragsthebest

[Counting Money 2](#)  
by evanb

[Blobules](#)  
by angelicchariz...

### ScratchEd

Do you help people learn Scratch? Join ScratchEd, our new online community for educators.

[Find out more](#)

### Video Tutorials

Figure 1-2: The home page of the Scratch Online Community website <http://scratch.mit.edu> (July 2012).





Figure 1-3: Scratch programming environment (version 1.4). The leftmost column represents an inventory of possible “programming blocks” which are assembled into program code in the center column. The area in the top right represents the project as it will be displayed to a user interacting with the finished product. The bottom right column shows the available sprites which are controlled by the code.

in-depth study of how the conditions under which Scratch projects are created influences remixes' originality and generativity.

In the sixth chapter, I examine people's *attitudes toward remixing*. This chapter investigates *how people react to remixing*. In particular, I focus on the perspective of young people as creators of artifacts that are remixed in the Scratch Online Community, rather than only focusing on youth as remixers as much of the literature does.

The concluding chapter summarizes the four main *contributions* of this work: the creation of a working social computing system, the development of a one-million-member online community, the collection of a rich research data set of user interactions, and a framework to understand remixing based on a set of mixed-methods studies. This chapter also provides directions for future research.

This work relies primarily on quantitative and qualitative analyses of the Scratch Online Community. I base my findings and rhetorical arguments on interviews, case studies, experiments, statistical and network analyses. The quantitative analyses examine a large corpus of five years of log data from that includes more than one of million registered accounts, close to ten million comments, and two million interactive media objects. In addition, many findings are based and motivated by my personal experience as designer and participant of the Scratch Online Community from its start until its five-year anniversary.

In the spirit of remixing, this introductory chapter and the dissertation as a whole reuse several coauthored articles, blog posts, and unpublished write-ups created during the past few years by myself and many collaborators. I will indicate whenever possible the exact source of the text. Often, I have modified the original text to fit the structure of this document better. All errors are my own, and do not reflect my collaborators' efforts.

## 1.2 Economic and Cultural Production

Network information technologies have facilitated the emergence of social production as an alternative to markets and firms in what is known as *commons-based peer production* (Benkler, 2002). Many of today's products, services, and cultural icons are the result of peer-produced innovations (von Hippel, 2005). From the software that we find in our devices to the content we consume on sites like YouTube. The cultural practices have changed, and the boundaries between production and consumption have blurred (Bruns, 2007). These commons-based peer "prod-users" need access to the work of others for them to remix and share back the derivatives.

These creative activities, based on the idea of building new things by combining existing ones, are not new. For a long time artists have engaged in similar practices through "appropriation art", "pastiche", "collage", "sampling" and "bricolage". Furthermore, folk culture and oral traditions rely on the idea of remixing what others have made. As Manovich (2005) argues, ancient Rome was a remix of ancient Greece, exemplifying how remixing is part of cultural evolution. Similarly, Jenkins (2006) describes how remixing has been part of art practice for a long time:

*[T]he story of American arts in the 19th century might be told in terms of the mixing, matching, and merging of folk traditions taken from various indigenous and immigrant populations.*

Despite this common feeling that everything old is new again, the influence digital technologies have had in remixing practices is self-evident. Manovich (2005) has called remixing “a built-in feature of the digital networked media universe,” while Sinnreich et al. (2009) argues that these technologies have enabled people to create perfect copies and remix the source materials themselves rather than just being inspired by them. Jenkins (2007) argues that

*it was the emergence of participatory cultures of all kinds over the past several decades that has paved the way for the early embrace, quick adoption, and diverse use of platforms like YouTube.*

Not surprisingly, Internet culture is particularly amenable to remixing. The popular “image macros” for example are images that get remixed ad nauseam (see Figure 1-4). Users of influential websites like 4chan often refer to having a special folder on their computers where “they preserve images for future enjoyment or remixing” (Bernstein et al., 2011). As early as 2005, a survey of online US teenagers (Lenhart and Madden, 2005) found that more than half (57%) of them had created content “for the Internet,” of those, one-fifth (19%) reported being “content remixers” —slightly more than adults (18%). The survey found remixing to be “equally prevalent across genders, ages, and socioeconomic groups.”



Figure 1-4: Image macro mocking the recursive nature of remixing. The image itself is a remix of an image taken from a video of the rap artist “Xzbit.” (Quickmeme, 2011)

### 1.3 Software Engineering

Software is born digital so remixing is inherent in the way it develops. Engineers constantly try to devise ways of avoiding the duplication of efforts, “Good programmers know what to

write. Great ones know what to rewrite (and reuse)” Raymond (1999). Furthermore, two of the tenets of Free and Open Source Software are precisely the ability to remix and the opposition to any restrictions of this freedom:

*Creativity can be a social contribution, but only insofar as society is free to use the results. If programmers deserve to be rewarded for creating innovative programs, by the same token they deserve to be punished if they restrict the use of these programs (Stallman, 1985).*

Even proprietary software firms, those who put restrictions on the use of computer programs, encourage in-house remixing. Software engineers have a long history of studying code remixing or reuse, primarily with the goals of improving productivity and quality (Frakes and Terry, 1996). For example, they have developed quantitative measures for remixing, such as the model of software reuse based on Gaffney Jr and Durek (1989), that calculates the “amount of reuse” using the lines of code as the unit of analysis (Frakes and Terry, 1996).

$$\text{amount of reuse} = \frac{\text{lines of reused code in system or module}}{\text{total lines of code in system or module}}$$

This more formal approach to remixing is *component-based software engineering*, a discipline concerned with

*designing components and libraries to be reusable as-is, identifying and isolating reusable components from an existing code base, and making a searchable repository of components (Philip et al., 2012).*

This discipline is based on the use of well-documented libraries, modules, and API’s<sup>2</sup>, leading to what it is known as “service-oriented computing” (Papazoglou and Georgakopoulos, 2003).

The second approach, called “opportunistic programming”, means the type of software development “with little to no upfront planning about implementation details,” where “ease and speed of development are prioritized over code robustness and maintainability” (Brandt et al., 2008). This type of programming relies on “existing source code that was not specially packaged for reuse” (Philip et al., 2012), such as copying-and-pasting (Kim et al., 2004) from tutorials, discussion forums, emails, and Q&A websites.

## 1.4 Learning and New Media Literacy

The remix literacy observed among software developers is also widely popular in informal learning environments (Perkel, 2008). Cultural anthropologists and media scholars have documented how young people engage in creative practices through remixing. For example, Ito (2007) has described how children relate to media franchises such as Pokémon by “collecting their own set of cards and virtual monsters and combining them into a deck or battle

---

<sup>2</sup>Application Programming Interfaces

team that reflects a unique style of play,” showing how they can “master highly esoteric content, customization, connoisseurship, remixing.” Similarly, Jenkins (2006) has narrated how children have become active participants in media creation by remixing their favorite literary characters, such as the Harry Potter fan fiction communities he has studied.

These observations have motivated scholars to argue that remixing is a necessary skill to succeed in today’s society. For example, Jenkins et al. (2009) argues that “appropriation” is one of the core literacy skills for the 21st century, which he defines as the “ability to meaningfully sample and remix media content.” Jenkins proposes to support remixing by asking students to, for example,

*work in teams to think through what would be involved in transforming an existing media property (a book, film, television series, or comic book) into a video or computer game.*

Livingstone (2008) has described similar practices when studying Internet use but also presents the challenges that these creative practices, such as legal and ethical issues, have to handle.

Even before remixing was popular, experts had argued the importance of providing learners access to social environments with common resources on which to build. For example, Wenger (1998) stressed the importance of having access to a “shared repertoire of communal resources” to help shape a “community of practice.” Similarly, Bruckman (1998) argued that building on previous works “encourages people that they can and would like to make something of their own,” advocating for systems where “every object in the system functions as a possible model to learn from and be inspired by.”

Building on the idea of Constructionism — a learning philosophy based on the idea that some of the most valuable learning experiences occur when people engage in building personally meaningful objects in apprenticeships (Papert, 1980) — Turkle and Papert (1990) advocated learning through remixing or “bricolage,” where learners construct by “arranging and rearranging [...] a set of well-known materials,” rather than planning before time what they want to create. Similarly, Situated Learning advocates for legitimizing peripheral forms of participation, in particular through socialization, visualization, and *imitation* in apprenticeship-like learning environments (Lave and Wenger, 1991) that support the spirit of remixing.

## 1.5 Ethical and Legal Challenges

Although remixing opens new possibilities for learning, economic, and cultural production, it is also morally and legally contested. Judge Posner (2007) has articulated that plagiarism is highly context-dependent and that one can assess the ethics of appropriation by thinking through the lens of deception, perception, and social expectation. For a more grounded analysis, take the following three examples and the questions they pose.

- The creator of one of the most popular Harry Potter fan fiction community websites, a young girl, was sued for copyright infringement by the company who owns the

rights to the book and the movie. Eventually the company dropped the legal action and reached an agreement (Jenkins, 2006). What should young people’s rights and responsibilities be when remixing?

- A federal court judge determined that photographer Richard Price violated copyright law for creating a type of appropriation art that consisted of taking pictures of the famous Marlboro cowboy advertisements and printing them in a large format (Batts, 2011). What role does originality play in remixing?
- Partly because of the Digital Millennium Copyright Act, YouTube had to implement a system for automatic detection of remixing of copyrighted content. This led to 5% of YouTube videos being removed for copyright infringement (Cha et al., 2007). How should automated systems handle remixing?

Examples like these prompted Lessig (2008) to argue that “copyright extremism” “chills” innovation and creativity, especially among young people who often engage in these practices. Similarly, Benkler (2006) argues that if we want peer-production to flourish, we must figure out how to enable remixing:

*If we are to make this culture our own, render it legible, and make it into a new platform for our needs and conversations today, we must find a way to cut, paste, and remix present culture. It is precisely this freedom that most directly challenges the laws written for the twentieth-century technology, economy, and cultural practice.*

Such arguments led to the development of the Creative Commons, which, among other things, provide a set of human and machine-readable licenses that empower creators to have more control of their copyright and encourage them to release their work legally under more permissive licenses that foster amateur creativity.

At the core of examining the ethics of remixing lies the understanding of cooperation — how much people are willing to sacrifice their selfish and rational desires to obtain monetary and reputational gains, or to behave in altruistic and cooperative ways, which are the foundations of my work.

## 1.6 Social Computing System Design

YouTube’s automation example highlights one of Lessig’s important contributions to system design: “code is law” (Lessig, 2000). As I developed the Scratch website, one of the biggest challenges was to decide what features to implement. Examining other systems that support collaborative online creativity helped in the design of the website. This came from both my own experiences interacting with many websites as well as human-computer interaction research, more specifically the field of computer-supported collaborative work.

Wikipedia has been perhaps the most widely researched collaborative system. For example, Viegas et al. (2004) developed a visualization of Wikipedia edits that led to insights into the nature of the system and its editors’ ability to collaborate. Later Kittur and Kraut (2008)

studied the quality of Wikipedia's articles in relationship to various types of coordination mechanisms.

Similarly, analyses of open-source software development have led to insights into the mechanisms that lead to successful cooperative projects. Raymond (1999), for example, argued that one of the lessons to be learned from open-source software programmers is the importance of knowing what to rewrite and reuse. He describes how Linus Torvalds (the creator of Linux) did not "try to write Linux from scratch" instead "he started by reusing code and ideas from Minix, a tiny Unix-like OS for PC clones."

Researchers have also developed web mash-up tools that allow people to remix web content (Bolin et al., 2005; Wong and Hong, 2007). A study on one of those tools found that despite many of its users lacking programming skills, the tools were an effective way of searching and aggregating information (Zang and Rosson, 2008).

More specifically on online communities for remixing, Shaw and Schmitz (2006) developed a video remixing platform and studied the nature of the generative video segments intending to understand how to integrate automatic recommendation systems with user-driven suggestions. Similarly, Diakopoulos et al. (2007) analyzed users' participation in a video remixing website and documented how participants developed specific norms for appropriating other's work that were not encoded in the architecture of the website.

Additionally, a study of the music remixing online community ccMixter looked at the impact of a remixing contest in the community dynamics. The study found that the contests increased participation among newcomers but that they did not continue using the website after the contest (Cheliotis and Yew, 2009).

## Chapter 2

# Remixing Systems

*The idea was that anybody who used the web would have a space where they could write and so the first browser was an editor, it was a writer as well as a reader. Every person who used the web had the ability to write something.*

—Tim Berners-Lee, 2005

When I started developing the website for the Scratch Online Community, I looked around for similar systems to get inspiration. “Web 2.0” was in full swing. YouTube was less than two years old, and Flickr was close to three; both were gaining popularity but were not nearly as big as now. Those systems I examined influenced the design decisions that went into creating the Scratch website. In this chapter<sup>1</sup>, I list some of these systems and briefly examine how they approached remixing. This survey includes websites for sharing and remixing videos, images, audio, status updates, and programmable media. This list is not intended to be comprehensive, or to go into detail. My goal here is to give an overview of the state of remixing on the Web while the Scratch website was in development. In addition, my goal is to help frame subsequent analyses in the ecosystem of remixing systems.

### 2.1 Video

YouTube<sup>2</sup> is the largest video sharing website (Cha et al., 2007), acting as a host for a combination of amateur and professional content since 2005. It has become the quintessential media-sharing website, and, therefore, it has become a common target of copyright and licensing disagreements related to remixing. For example, the deletion from YouTube of a popular video of a baby dancing to one of Prince’s songs caused great controversy, and highlighted how copyright laws might stifle amateur creativity (Lessig, 2008). Since early

---

<sup>1</sup>Based on a co-authored poster paper with Oshani Seneviratne titled “Remix Culture on the Web: A Survey of Content Reuse on Different User-Generated Content Websites” available in Proceedings of Web Science 2010

<sup>2</sup><http://www.youtube.com/>



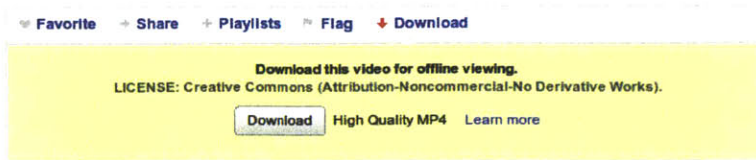


Figure 2-1: Download button and license statement of a YouTube video (2010).

2009, YouTube started to allow special “partners” to display a download button<sup>3</sup> and select the license for their videos (Tran, 2009), (see Figure 2-1). The licensing options were “personal,” “public domain,” and the various Creative Commons licenses. Partners had to be approved, however. Users could apply to become partners and YouTube would make a decision based on the size of the applicant’s audience (based on the popularity of their existing videos), among other metrics (YouTube, 2010). Originally, partners were primarily government entities (like the White House), universities, and other nonprofit organizations. Music labels and other companies joined later.

Today, anyone can publish his or her YouTube videos under a Creative Commons license, which enables a “remix” button under the video (see Figure 2-2). Clicking this button opens the video in a web-based editor allowing people to create derivatives right on their browser.

YouTube, under much pressure from copyright holders, implemented the first version of their content identification system on June 2007. This system allowed copyright holders to automatically identify remixes of their materials and decide to block the remix (sending a copyright infringement notice to the remixer), gather metrics of its popularity, or monetize from it by sharing some advertising revenue (Chen, 2007). Shortly after this system was put in place, 5% of all videos were deleted because of copyright infringement (Cha et al., 2007), including videos that had purchased the rights to use the materials. People whose remixes were taken down could file a complaint through YouTube’s legal department.

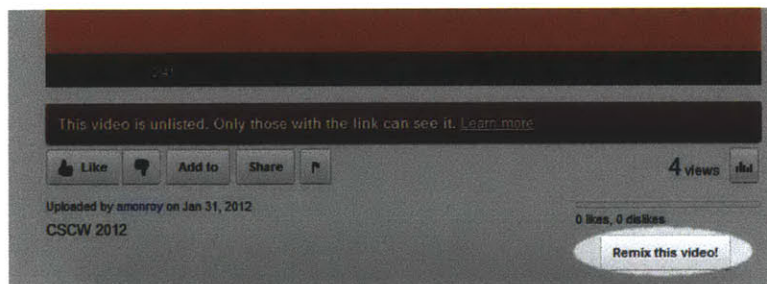


Figure 2-2: Remix button on YouTube (2012).

YouTube users who want to explicitly give credit or link to the source materials have repurposed the description and the “video reply” feature. The first consists of using the description of the video to reference manually the materials being remixed. The second is to

<sup>3</sup>Today there are several third-party tools to download YouTube videos.

use the YouTube video response to create a link between the remix and its source. “Video responses” need to be enabled by the person who uploaded the first video, though.

Similarly, embedding can be enabled or disabled on videos. This gives video creators the ability to allow external websites (such as blogs or social network sites) to remix their videos as part of other types of web documents.

Although YouTube is the largest video sharing website, Vimeo<sup>4</sup> has also gained much popularity, especially because at first it was the only website that allowed High Definition videos. Vimeo did not initially let users to set the license of their videos, and went as far as to prohibit people from uploading videos that are in the public domain, arguing that the “I have permission” part of their user agreement had to mean that the user had created the video. That seems to have changed now.

Vimeo, in its early days, implemented a feature that, YouTube still does not have. Vimeo allows video creators to give credit easily to other members of the Vimeo community (see Figure 2-3), which makes sense given the type of users Vimeo had attracted were more artistically inclined.

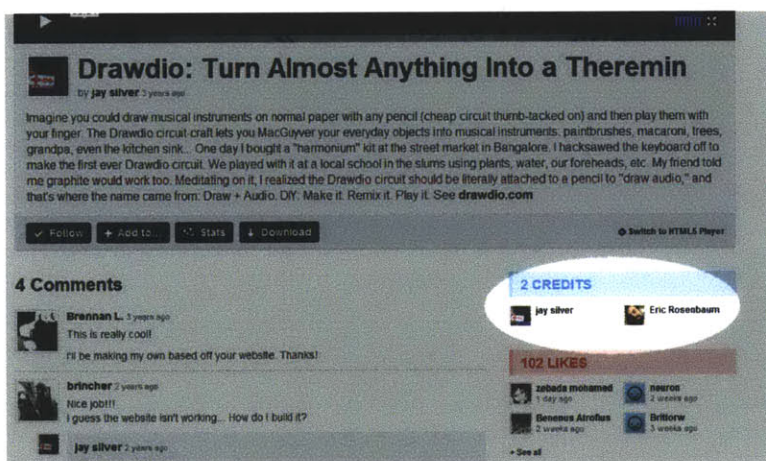


Figure 2-3: Credit-giving on Vimeo (2012).

Unlike YouTube, Jumpcut<sup>5</sup>, now defunct, actively encouraged remixing. Jumpcut was a commercial website later acquired by Yahoo that defined itself as “the easiest way to upload, edit, and share your video and photos.” Jumpcut (see Figure 2-4) was probably the first commercial website to actively encourage remixing through their technical and social features. The website had a web-based video editor that let users create sophisticated video mashups and an easy-to-understand definition of remixing with a preemptive response to authorship conflicts:

*Remixing is creating your own version of someone else’s movie, usually incorporating elements from the original and adding content or maybe just some of your*

<sup>4</sup><http://www.vimeo.com/>

<sup>5</sup><http://www.jumpcut.com/>

*own style and spicy goodness. It's an easy way to get started, and you can do it with the click of a button on any published movie. When you click "Remix," we'll pull back the curtain and show you what's behind the scenes. Then you can get busy being creative. Don't worry, you're not destroying someone else's work, you're just making your own copy. And if yours is better than the original, so be it. The community will tell you.*

Jumpcut had an automated system for giving attribution when remixing; however, interviews with some of their users found that people still felt the "moral obligation" to give explicit credit when remixing (Diakopoulos et al., 2007). The interviewees did not seem compelled to do the same when remixing content created by large companies.

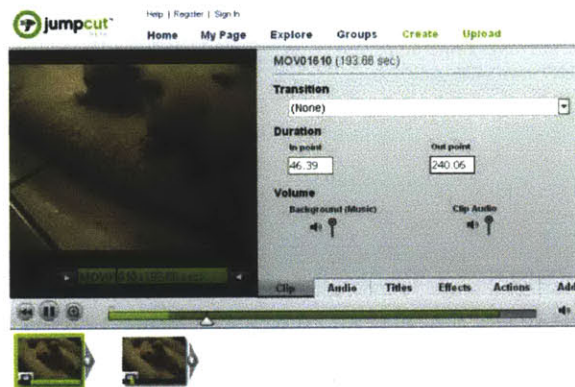


Figure 2-4: Jumpcut's web-based editor (2007).

## 2.2 Images

Flickr<sup>6</sup> is a popular photograph-sharing<sup>7</sup> website and one of the first commercial websites to adopt Creative Commons licenses. When choosing from one of the six available Creative Commons licenses (see Figure 2-5), people's photographs show the "some rights reserved" statement and a link to a web page explaining what those rights are. Five years since Flickr enabled Creative Commons licenses in 2004 more than 100 million photographs have been published under one of those licenses. Of those, 64% explicitly give the freedom to create derivative works or remixes (Thorne, 2009) (see Figure 2-6). Unless an alternative is chosen, Flickr's default license for photographs is a traditional "all rights reserved" license.

Besides photographs, image sharing also includes the sharing of drawings, paintings, and other types of graphic artwork. deviantArt<sup>8</sup> is arguably the most popular website for sharing this type of media<sup>9</sup>, with its more than 14 million members, and 100 million submissions (EvanitaEWM, 2010). deviantArt supports remixing in two ways. First, the

<sup>6</sup><http://flickr.com/>

<sup>7</sup>Flickr also allows people to share short videos but it is predominantly a photograph-sharing website

<sup>8</sup><http://deviantart.com/>

<sup>9</sup>deviantArt is primarily for sharing image but it allows other type of media.

### Select a default license

Don't forget to make sure that you have all the necessary rights and you won't be infringing on any third parties with any content that you license on Flickr. As per our [Community Guidelines](#), accounts are intended for members to share content that they themselves have created.

This will apply to everything you upload from now on. You can also change the license on all your existing public content in a [batch](#) if you wish.

- None (All rights reserved)
- Attribution-NonCommercial-ShareAlike Creative Commons
- Attribution-NonCommercial Creative Commons
- Attribution-NonCommercial-NoDerivs Creative Commons
- Attribution Creative Commons
- Attribution-ShareAlike Creative Commons
- Attribution-NoDerivs Creative Commons

You've previously [chosen to restrict](#) who can download your stuff. Selecting a Creative Commons license here will override that setting on future uploads.

SET DEFAULT LICENSE

Figure 2-5: License selection on Flickr (2010).

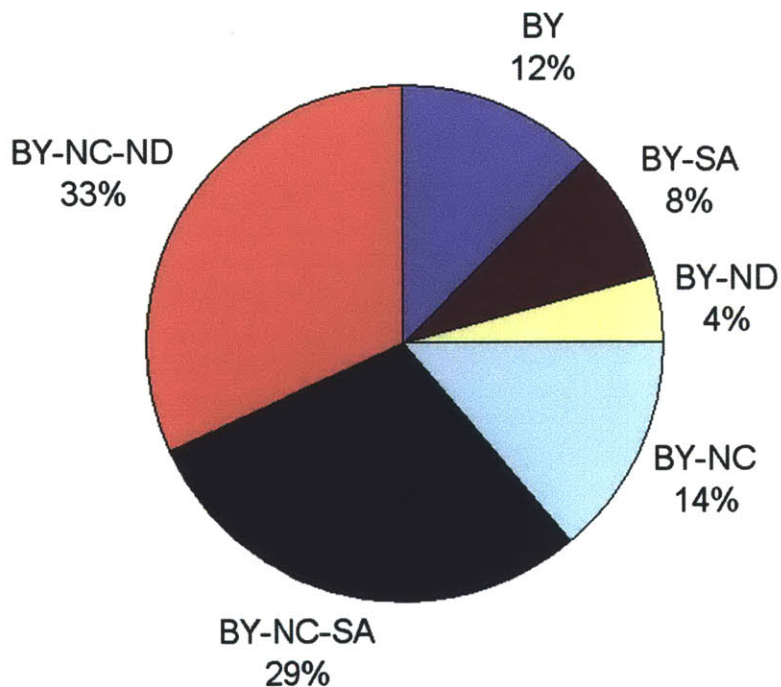


Figure 2-6: Distribution of Creative Commons licenses usage on Flickr (Source: Creative Commons blog).

Creative Commons licensing is built into their user interface (see Figure 2-7). Second, the website provides a mechanism to encourage remixing while giving automatic attribution to the source (see Figure 2-8). Over the years, several members of the Scratch community have been active participants of deviantArt, and vice versa. Even within deviantArt there are groups for Scratchers to hang out and share work to be used in their Scratch projects.

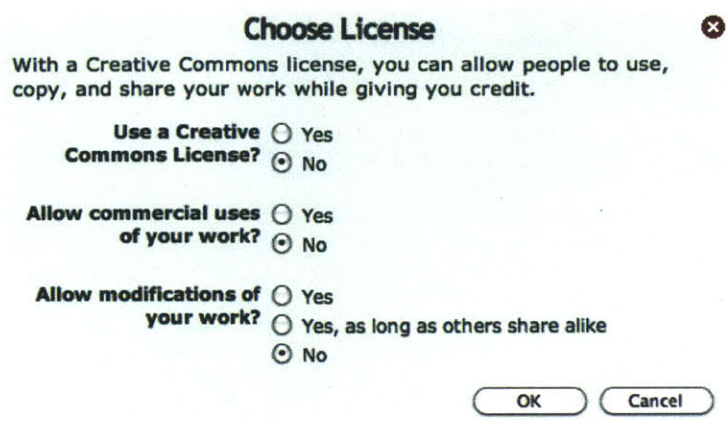


Figure 2-7: Selecting a Creative Commons license in DeviantArt.

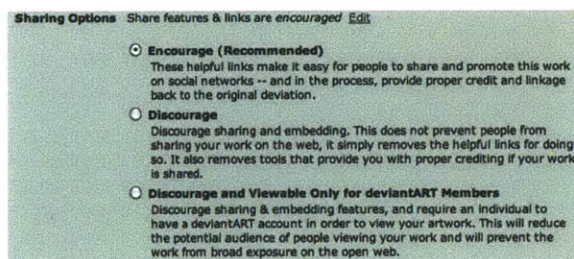


Figure 2-8: Sharing options in DeviantArt.

Finally, OPENSTUDIO, also from the MIT Media Lab, was a website that allowed people to create, remix, and sell images for virtual currency (Arikan, 2006). The website described itself as an experiment in creativity, collaboration, and capitalism. OPENSTUDIO let people draw using a simple online painting application (based on Java), then share their artworks with everyone. The drawings could be opened in their original format through the editor that allowed easy remixing. OPENSTUDIO (see Figure 2-9) hosted a lively community where creative collaboration and remixing took place.

## 2.3 Audio

ccMixer<sup>10</sup> is an audio-sharing website that lets people sample and remix bits of music.

<sup>10</sup><http://www.ccmixer.org/>

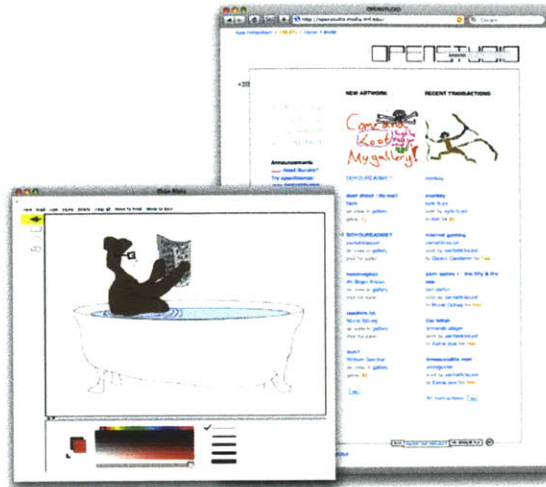


Figure 2-9: Front page and online editor of OPENSTUDIO.

ccMixer was explicitly designed to foster the use of Creative Commons licenses in an environment unconstrained by the legal limitations of other more commercial systems. ccMixer lets users choose from a list of Creative Commons licenses when uploading their work. Remixers are also prompted to identify the samples, or any other remixes, that were used in the composition. The remix inherits the most restrictive license from the sources used. The website links to all the individual components used in the remix (see Figure 2-10), essentially creating an attribution tree (Stone, 2009). The website also makes it explicit that the use of ccMixer content outside the website requires honoring the Creative Commons licenses associated with it. The managers of ccMixer experimented with the creation of

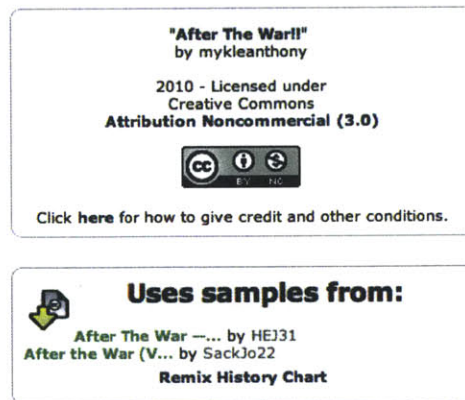


Figure 2-10: ccMixer showing the Creative Commons license and the source materials of a remix.

contests to promote engagement. This is one of the decisions that system designers are often confronted with: the use of incentives. The contests consisted in having users remix sounds donated by famous remixing artists. These contests attracted many participants

but most of them did not engage with the existing community members and promptly left once the contest was over (Cheliotis and Yew, 2009).

Unlike ccMixter, IndabaMusic<sup>11</sup> gives people web-based tools to create song remixes right on the browser. The website lets people upload audio files for which they, in theory, have use permission and select from one of three licensing options for those uploads: “Creative Commons Attribution,” “Creative Commons Attribution, Noncommercial,” and “All rights reserved.” This last one tells the community the user owns the file and is not granting anyone any special permission. Users can easily specify that a particular music file be used in a remix. The remixes of any music file are displayed prominently among other metrics of engagement such as number of posts (see Figure 2-11).

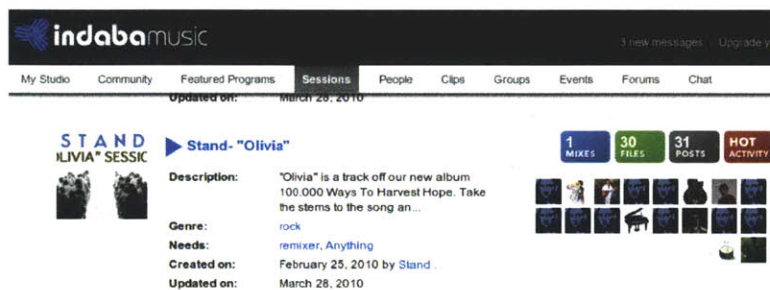


Figure 2-11: IndabaMusic displaying the number of remixes of a music file.

## 2.4 Status Updates

Myriad systems let people broadcast short messages online; Twitter<sup>12</sup> and Facebook<sup>13</sup> are the most popular ones. Millions of people flock to Twitter every day to share their latest

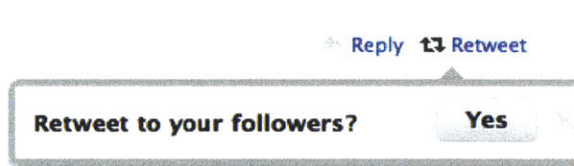


Figure 2-12: The “retweet” button, automating attribution on Twitter.

personal news, witty commentary, or links to interesting websites, among many other things. From the start, users had the need to rebroadcast other people’s messages. The user community organically developed various methods for this remixing, known as “retweeting” (boyd et al., 2010).

<sup>11</sup><http://www.indabamusic.com/>

<sup>12</sup><http://www.twitter.com/>

<sup>13</sup><http://www.facebook.com/>

Some people used prefix RT, short for retweet, followed by the name of the user being retweeted, others used the suffix “via,” also followed by the name of the retweeted user, and when modifying the original tweet people started using the prefix MT, short for modified tweet. After years of this emergent practice, in November 2009 Twitter formalized retweeting by adding a retweet button (see Figure 2-12). Users could still manually copy and retweet a message, but now they had the option of just pushing a button. Retweeting using the button meant to push the message, verbatim, to the timeline of those who follow the person doing the retweeting, with some text indicating who retweeted it. This feature was met with mixed reactions. Some Twitter users did not like the fact that they could not add their own commentary, and others complained that it was awkward to see someone who they do not follow appear in their timeline (Williams, 2009). Until this day, Twitter users continue using a mixture of automatic and manual retweeting.

Twitter terms of service gives Twitter an unrestricted and undefined license for all the content posted on their website, though the person who posted remains the official copyright holder (Inc., 2012). As a response to this, Identica<sup>14</sup> emerged as an alternative microblogging service that allowed users to post Twitter-like messages under a Creative Commons Attribution license. Like Twitter, identi.ca also gave users the ability to rebroadcast status updates with a button labeled “Repeat this notice?”

Like Twitter, Facebook owns the content posted on the website and does not allow users to specify any particular license for their posts. Facebook added a feature allowing users to re-share content in their activity streams, and specify whom it came from automatically (see Figure 2-13).

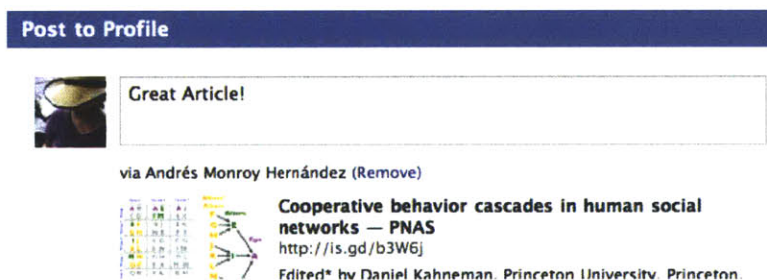


Figure 2-13: Sharing a friend’s post on Facebook, automatically attributed.

## 2.5 Source Code

Several systems allow people to share code. Github<sup>15</sup> and Sourceforge<sup>16</sup> are among the most popular. Both of these websites target professional and hobbyist developers, often involved in large software applications. SourceForge used to be described as “the world’s largest Open-Source software development website.” The website hosted more than 100,000

<sup>14</sup><http://www.identi.ca/>

<sup>15</sup><http://www.github.com/>

<sup>16</sup><http://sourceforge.net/>



projects and more than 1,000,000 registered people that used the website to manage projects, bugs, communications, and code (see Figure 2-14). Slowly, GitHub has taken over as the place for sharing source code. Part of the appeal of GitHub is the ability to remix a project, known as “forking,” and subsequently submit the changes in the fork back to its parent (see Figure 2-15).

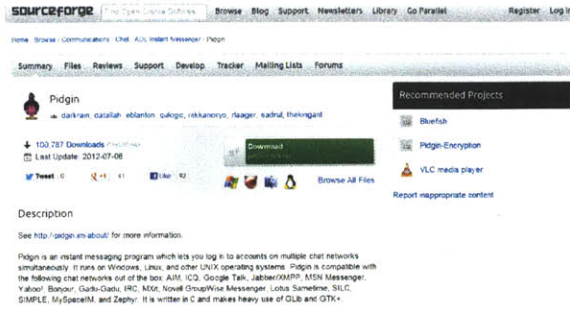


Figure 2-14: Sourceforge project page.

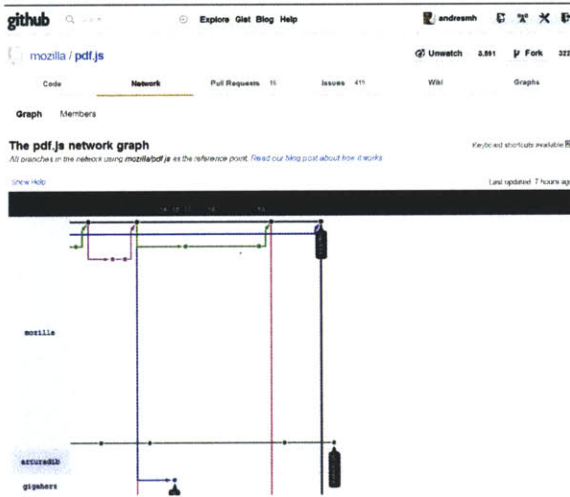


Figure 2-15: Forking graph on Github.

## 2.6 Programmable Media

By combining traditional media, such as images and sounds, with programming instructions, people can create what I refer to as “programmable media.” Examples of these media are animations, video games, and the many widgets that populate the Web. The code does not need to be sophisticated to produce engaging widgets. People often say that a little bit of code goes a long way.

One example of minimalistic widgets appears on the website YTMND<sup>17</sup> (see Figure 2-16),

<sup>17</sup><http://ytmnd.com/>

which stands for the expression “You are The Man Now, Dog.” This website lets people create simple animations by combining audio and images with an implicit program that puts both in an infinite loop. YTMND, which describes the website as a community for creative expression:

*YTMND is a site created for furthering the creativity of its users. It stems from an idea that, using sound, and image, and some text, the users can convey a point, funny, political, or otherwise, to the general media. By becoming a member, you can vote, comment, and make your own YTMNDs. YTMND is like a big family, full of entertainment, drama, and joy.*

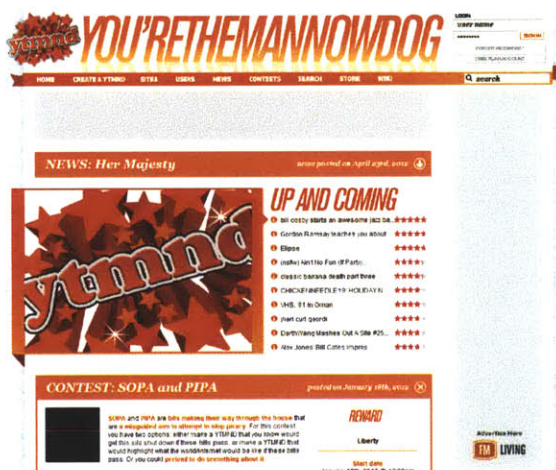


Figure 2-16: YTMND’s front page.

Among more sophisticated, although still focused on amateurs, Newgrounds<sup>18</sup> and MyGame<sup>19</sup> are two online communities for the sharing, and remixing interactive media as Adobe Flash. MyGame (Figure 2-18) has an option for novices to create games based on templates. These do not allow the creation of new behavior but let users put their own images on a game. To contribute to MyGame, Flash developers must use an API provided by MyGame. Newgrounds (Figure 2-17) gives more freedom than MyGame by allowing the upload of almost any kind of Flash game. Newgrounds has a section where people can contribute music or images for others to use. Newgrounds has an active community that often participates in collaborative project-making (Luther and Bruckman, 2008).

Also developed at the MIT Media Lab (and later moved to Georgia Tech), MOOSE (MUD Object-Oriented Scripting Environment) Crossing was a text-based MUD (Multi-User Dungeon). The system allowed children to interact and collaboratively build a virtual world (see Figure 2-19): “[u]sing a scripting language, participants can add behavior to objects (things, places, and creatures) in the simulated world” (Bruckman, 1998). MOOSE Crossing was probably one of the first online communities for children to engage in sharing programmable media. The spirit of MOOSE Crossing heavily inspired the Scratch website.

<sup>18</sup><http://newgrounds.com/>

<sup>19</sup><http://mygame.com/>

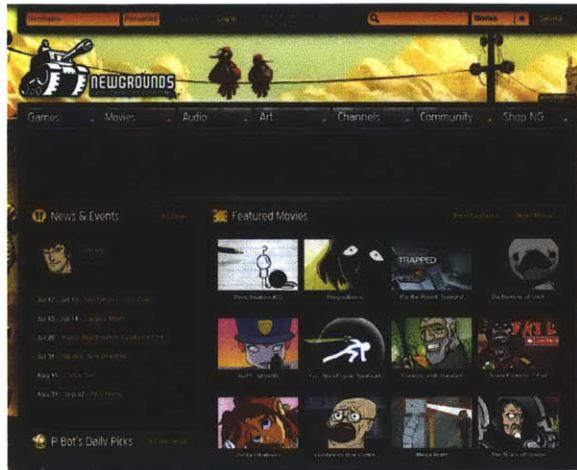


Figure 2-17: Newgrounds' front page.

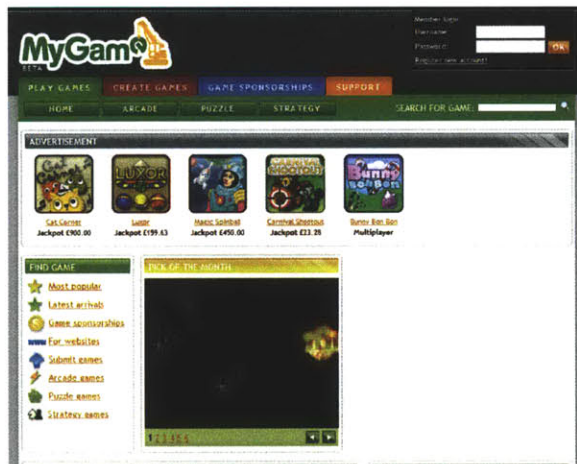


Figure 2-18: MyGame front page and page showing game edition (template).

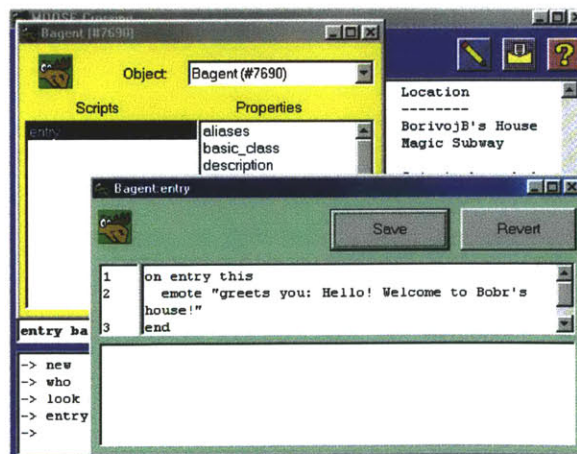


Figure 2-19: Moose Crossing: play and scripting environments.

## 2.7 Summary

I have presented a small set of websites to give some insights into what system designers have done to support the development of remix culture across various types of media. Content sharing and the associated challenges are not just limited to the sites outlined in this section. Much work is needed to solve the legal and social complexities of remixing. Based on these observations, I decided on several things for the design of the Scratch website:

1. Default to the most open license possible that enables community building through remixing. The Creative Common Attribution Share Alike license fit this criterion.
2. Display provenance to support navigation and discoverability.
3. Enable source code download for every project. Give people the tools to easily embed and remix content.

## Chapter 3

# The Scratch Online Community

*Literacy means both reading and writing, but most books and courses about computers only tell you about writing programs. Worse, they only tell about commands and instructions and programming-language grammar rules. They seldom give examples. But real languages are more than words and grammar rules. There's also literature — what people use the language for. No one ever learns a language from being told its grammar rules. We always start with stories about things that interest us. (Minsky, 1984).*

*... we have considered how mathematics might be learned in settings that resemble the Brazilian samba school, in settings that are real, socially cohesive, and where experts and novices are all learning. The Samba school, although not “exportable” to an alien culture, represents a set of attributes a learning environment should and could have. Learning is not separate from reality. The samba school has a purpose, and learning is integrated in the school for this purpose. Novice is not separated from expert, and the experts are also learning. (Papert, 1980).*

*I've found a great site called Scratch. It's about programming. You snap together blocks to create stories, games and animations. Then you can share your projects on the web!!! You can download it for free! But you have to become a member to share your projects. But membership is free as well!*

*—10-year-old boy*

In this chapter<sup>1</sup> I describe the development of the Scratch Online Community from my perspective as system designer, administrator, and active participant. I have divided this chapter in two parts. First I present the development of the *sociotechnical infrastructure* of the Scratch Online Community. I describe its conception, guiding principles, and development process. In the second part I examine how the community interacted with the

---

<sup>1</sup>partly based on previously published work (Monroy-Hernández, 2007; Monroy-Hernández and Resnick, 2008; Monroy-Hernández, 2011).

system and with one another, from its beta release until its fifth anniversary when it had accumulated more than 380,000,000 pageviews, 56,000,000 visits, 34,000,000 unique visitors, 2,000,000 submissions, and 1,000,000 registered accounts.

In the spring of 2006, as part of a project for the class “Creative Learning Technologies” at the MIT Media Lab, I proposed what I originally described as a “Flickr for Scratch” (Monroy-Hernández, 2006). In the first slide of a presentation I prepared, I summarized the project as “Scratch + Web 2.0 = ScratchR.” Clearly inspired by the “social imaginary” of Web 2.0 (O’Reilly, 2005), my idea was that ScratchR would be the underlying infrastructure to develop an online community of people sharing programmable media created with Scratch. More important, ScratchR would support the Constructionist learning philosophy on which Scratch was based.

Scratch had perfected its visual grammar by this time, but, using Minsky’s metaphor, it had yet to develop a rich literature. It already existed as a fully fledged development environment that allowed people to control the behavior of images, text, and sounds using visual programming blocks. However, Scratch did not have much of an online presence: its website consisted of a blog used primarily to share guides with educators who had access to an early version of the software (see Figure 3-1). Nor was the development environment, a key tool for creating images and animations, publicly available. Thus, it had not reached its potential for what an early design document called “deep shareability,” part of the original vision of Scratch:

*Youth are constantly looking at one another’s projects, trading ideas, sharing techniques. To fit into this context, the object architecture of Scratch supports what we call ‘deep shareability’ (Maloney et al., 2004).*

## 3.1 Infrastructure Development

Inspired by Minsky and Papert, with whose words I opened this chapter, my goal was that the Scratch Online Community would become a sort of “Samba school” where novices and experts would gather to read, write, and remix Scratch literature. I wanted to take Scratch from being “just” a good tool to a space where peers gather to create, share, remix, and even just “hang out”: a commons-based peer production community. This meant giving people access to an *audience*, potential *collaborators*, and a *repository* of inspirational creations that creators could learn and remix from.

### 3.1.1 Ideation

I drew ideas from previous research and commercial systems when outlining the features to implement and the design guidelines to follow. MOOSE Crossing was particularly influential because it also tried to realize Papert’s samba school metaphor by creating an online community of young people engaged in programming a virtual world (Bruckman, 1998). I also got ideas from social network sites like MySpace —the dominant social network site

# SCRATCH

## Welcome to the SCRATCH site!

Scratch is a new programming language that lets you create your own animations, games, and interactive art.

Scratch will be available for public release in Summer 2006.

Learn [more about Scratch](#).



[New Post](#)

[Latest Scratch cards](#)

- [Questions and Tips](#) (16)
- [Sensor Board](#) (3)
- [Share Your Projects](#) (9)

[Post a new entry](#)

### Recent Entries:

[Using Scratch CD](#)  
[Problem: asks for image file](#)  
[Project won't open?](#)  
[Problem opening zip file](#)  
[Presentation mode](#)  
[Where can I download Scratch?](#)  
[a simple game controller](#)  
[Valentine from Denis Coffey](#)  
[Erasing image background](#)  
[Copy a script to another sprite](#)

- [Quick Start Guide](#)
- [Scratch Intro Dance Video](#)
- [Project Ideas](#)
- [Frequently Asked Questions](#)
- [Help Screens](#)
- [Questions and Tips Discussion](#)
- [Ways to introduce Scratch](#)
- [Group activities](#)
- [Scratch Cards](#)
- [About Scratch](#)
- [Publications](#)

Scratch is being developed by the [Lifelong Kindergarten research group](#) at the [MIT Media Lab](#), in collaboration with [KIDS research group](#) at the UCLA Graduate School of Education & Information Studies.

[Subscribe](#)

Figure 3-1: Scratch website (April 2006)<sup>2</sup>

<sup>2</sup>The blog announced the release of Scratch for the summer of 2006 but the actual release date was moved to May 2007. Image from: <http://web.archive.org/web/20060424065311/http://weblogs.media.mit.edu/11k/scratch/>

at the time— Friendster, Xanga, and Facebook. I especially focused on websites that let people share artistic or creative works, such as Flickr and YouTube. From these existing systems, I created this short list of *activities and principles* that I thought would be essential for ScratchR to support:

*Creative socialization:*

Creative online communities should use the basic features of a social networking site, such as “friending,” to support creative endeavors, rather than socialization as the core activity. As I described in the introduction, years of learning science research show that working with others provides richer learning and social experiences, so it is essential for the system to allow light-touch and more involved forms of collaboration.

*Remixing:*

The platform should make it easy for people to reuse other’s work and let users know when and how this happens. Remixing is both practical and social.

*Participatory diversity:*

The system should give people freedom to move from “lurking” to contributing content, and back. The platform should acknowledge listening as a valid state of participation; for example, a simple view count allows creators achieve a sense of their otherwise silent audience. Similarly, supporting users to engage in “active production” means removing any barriers for them to do so. For example, although it is possible to upload content to a website by filling out a web form, sharing within the authoring environment lowers the barriers for contribution to the point that it could even replace storing in one’s hard drive.

*Serendipity and findability:*

Having a rich literature requires an easy browsing method that allows users to stumble upon content, as well as finding exactly what one wants when searching for it. The systems should provide highlight content, both from experts and novices, and provide hyperlinks connecting people, their creations, their feedback commentary, and across systems.

*Light-touch steering:*

The system should allow administrators to set the direction of the online community through shared activities, and by promoting interactions. Some ways to build this is to highlight specific activities on the website’s front page, and to organize challenges and themed-based creations, while still giving the freedom to ignore those recommendations.

### **3.1.2 User Experience**

The design process of the Scratch website involved creating mockups and prototypes. I created the first mockups based on Flickr. I took screenshots of some sections of Flickr, and simply edited them to show how the pages would look like in Scratch (see Figure 3-2). Similarly, I created a mockup of the user interface for uploading projects within the Scratch development environment (see Figure 3-3). Using those mockups, I had a brainstorming



session with some of my colleagues to outline the key elements of the website (see Figure 3-5). Once I had a clearer idea of what I was building, I created a site map to lay out the basic set of pages (see Figure 3-4). Then, I created a working prototype focusing only on the client-side user experience (i.e., HTML pages), without a backend infrastructure to store real data. I did this by remixing the design of the now defunct video-sharing website vSocial<sup>3</sup> (see Figure 3-6). I hosted this prototype on <http://scratchr.org/>, using a commercial web hosting solution.

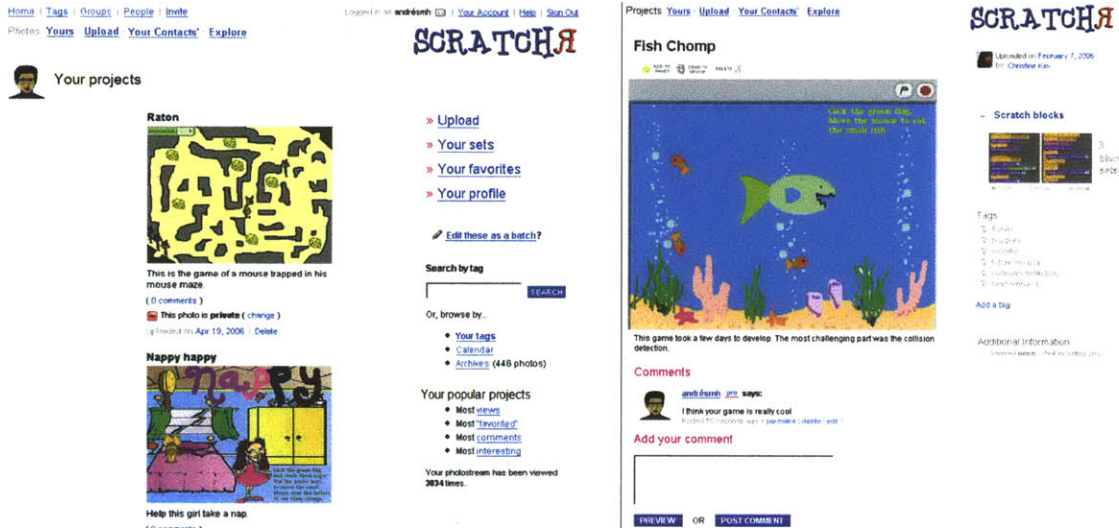


Figure 3-2: Mockup of Scratch website page based on Flickr (May 2006).

Based on these mockups and prototypes, I compiled the following list of graphics design guidelines that would inform the implementation of the first version of ScratchR. This list was heavily influenced by web usability studies with teenagers (Nielsen, 2005) and my own preferences of the moment regarding the make-up of a “modern” website.

- Aesthetics should be playful, but not “childish.” The website should feel young but still “cool” enough for teens. Similarly, the website should be intelligible to adults, such as parents and educators, so they can discover where they can get information that is more formal.
- Use “modern design,” including subtle use of gradients, shadows, reflections, light or white background color, slight “roundedness,” and “webify” the Scratch logo (at the time, the logo looked more plain).
- Visual elements should be defined only in the CSS<sup>4</sup> to reduce load time, complexity, and cost of maintenance.
- Layout and aesthetics should be minimal to avoid taking attention away from community contributions. This means relying on careful use of space, avoiding user interface

<sup>3</sup><http://vsocial.com/>, no longer available

<sup>4</sup>Cascading Style Sheet

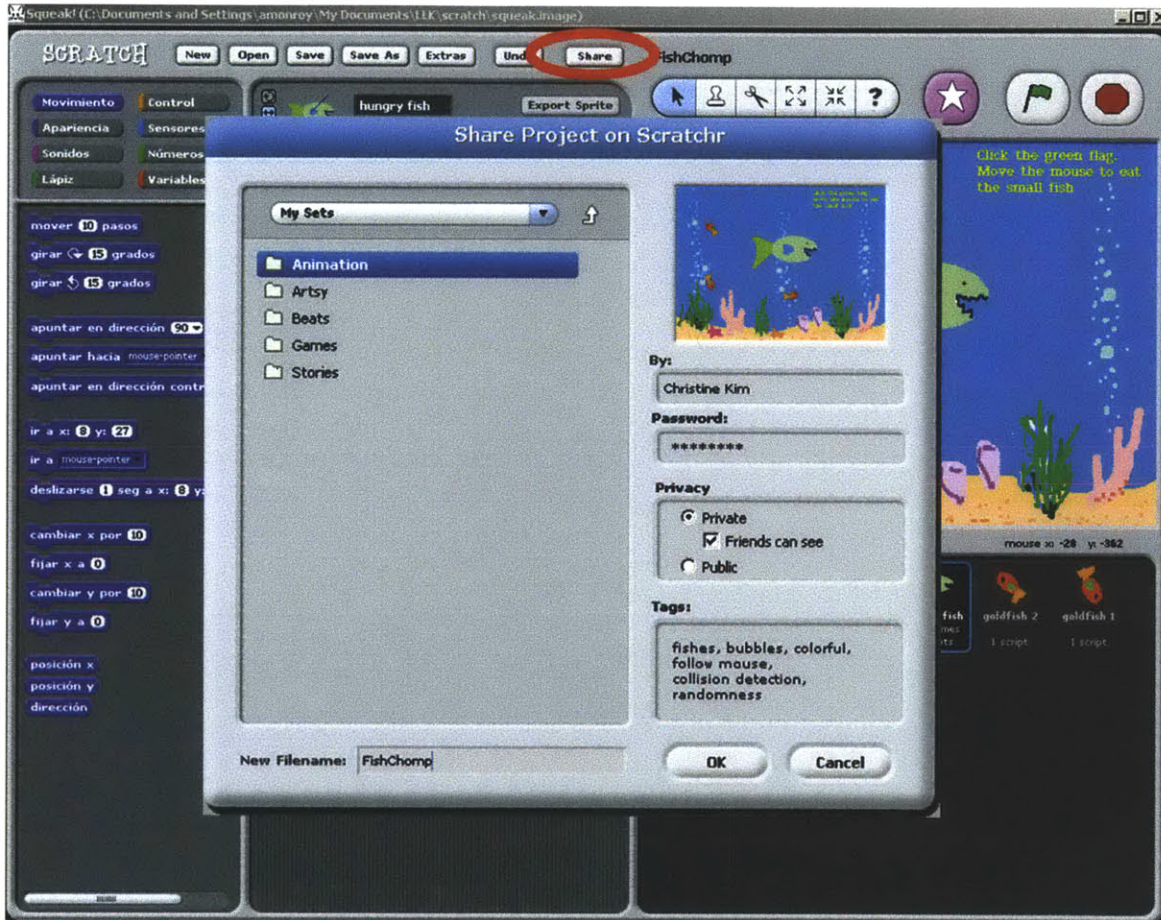


Figure 3-3: Mockup of interface for uploading a Scratch project (May 2006).

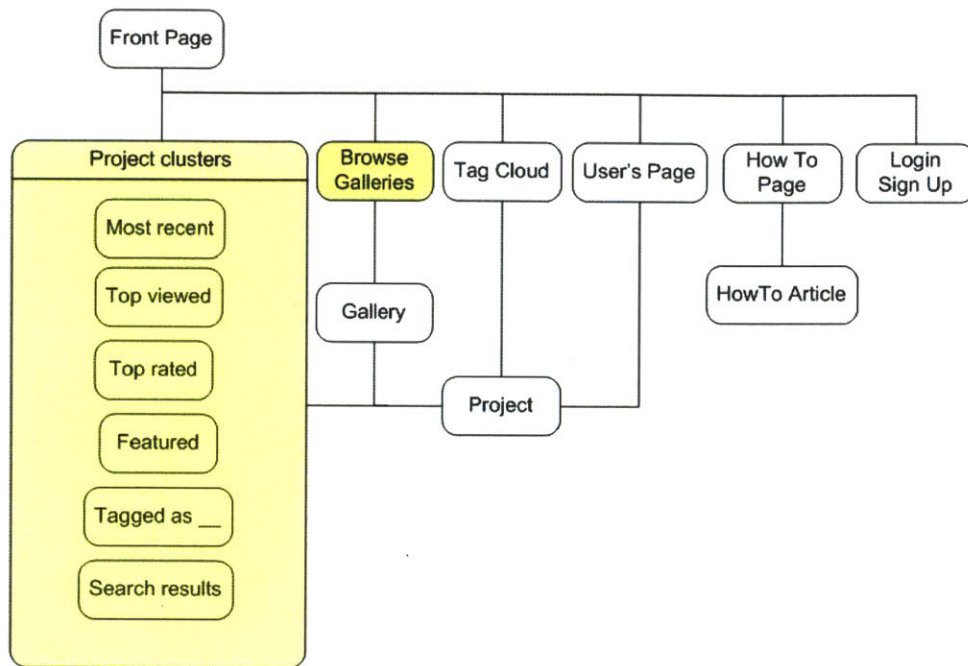


Figure 3-4: Site map outlining all the sections of the website (May 2006).

elements with gratuitous animations, and minimal use of text.

- Use big fonts to attract attention to important elements.
- Design should degrade nicely. No functionality should be lost across browsers, including older or less powerful Web clients.
- Ability to handle many languages.
- Visible tagline and description. Present a short (3 or 4 words) tagline. Before settling on “imagine, program, share,” we considered the following: “Programming for everyone,” “Now everyone can program,” “playful and powerful programming for people like you,” “ScratchR: the premiere place for posting playful projects and powerful programs,” “Creative tool for self-expression.”
- Visually describe the steps involved in using Scratch and the website. Use simple icons and some words. Each element of the graphic should link to the corresponding documentation section.
- Highlight novice and expert content to be inviting to both. Instructables<sup>5</sup> had a good model for this, as many of the projects on the front page were amateur-looking rather than professional, which might turn off some shy contributors.
- Show header and footer on all pages, linking to the core elements of the website such as the user portfolio page, home page, login, etc.

<sup>5</sup><http://instructables.com/>

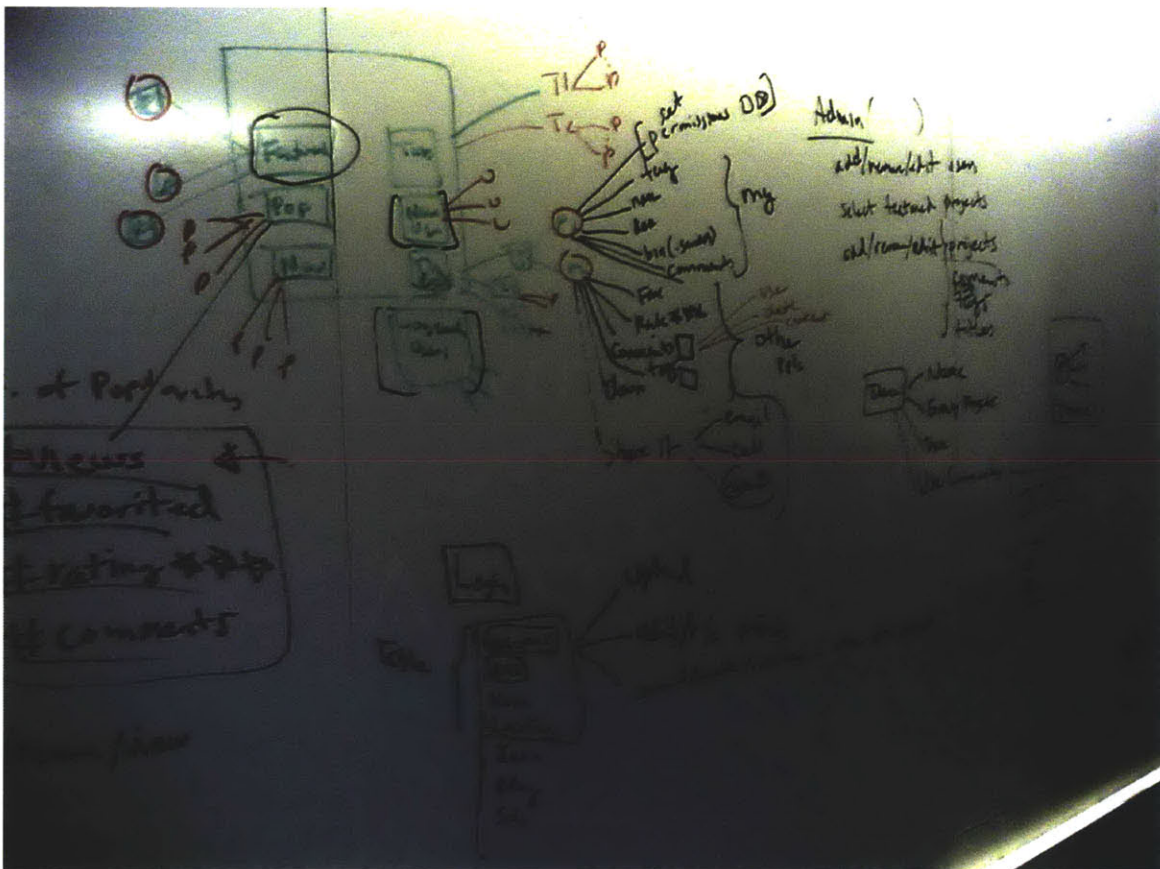


Figure 3-5: Whiteboard outlining the elements of the website (June 2006).

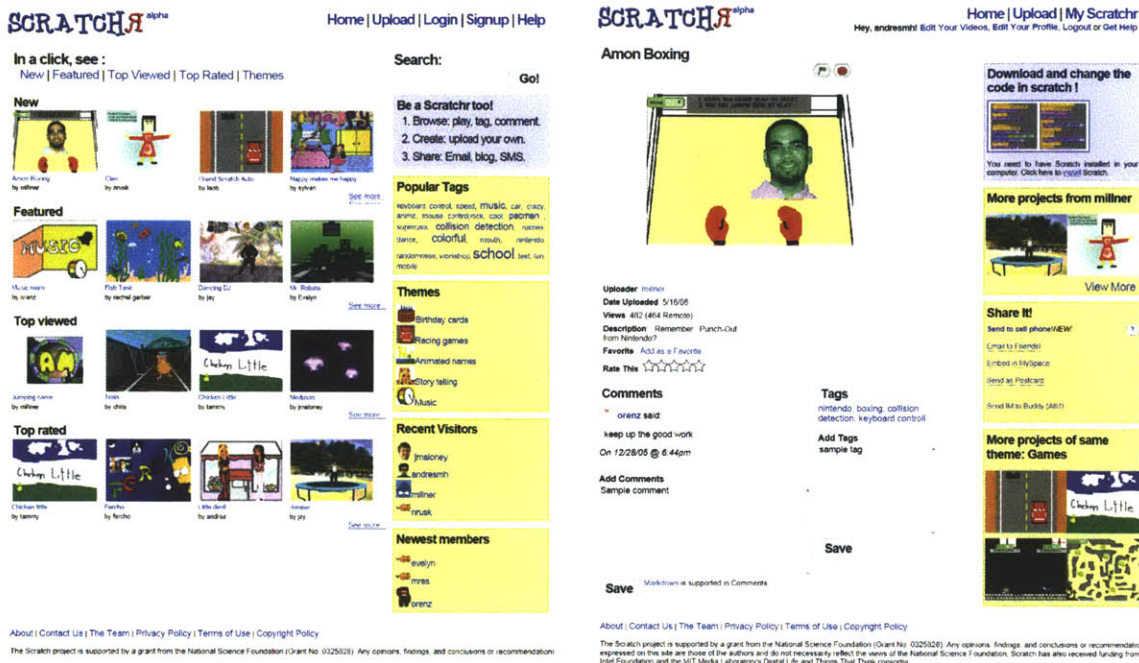


Figure 3-6: First prototype of Scratch website (June 2006)

### 3.1.3 Architecture

Once I had those prototypes and guidelines, I worked on implementing the first version of the website.<sup>6</sup> To do this, I outlined the following core components of the website:

1. A *repository* of projects and metadata. It is possible for anyone to download projects for modification and later re-upload to the website as a remix. Each project has its web page with its own display where people can interact with it and others.
2. A *friendship network* consisting of profile pages and unidirectional friendship connections. Profile pages list the friends, projects, and “favorited” projects for each user with his or her avatar image and the country of origin.
3. Diverse *interaction* activities such as commenting, downloading, tagging, “loving,” and “favoriting.”
4. Project *clustering*. Using tagging, galleries, and interaction counters (for example, number of “loveits”), group projects based on users’ or administrators’ input.
5. An *API*<sup>7</sup> to support external and internal services, including connectivity with the Scratch desktop application, and later community-driven websites.

<sup>6</sup>Special thanks to Ubong Ukoh and Han Xu, undergraduate research assistants, and Kemie Guadia, graphic designer.

<sup>7</sup>Application Program Interface

6. Discussion *forums* where community members help one another with technical problems, find collaborators, and talk about non-project topics to foster a sense of community on the website.

The current website includes four primary categories of pages: the front page (see Figure 3-7), the project page (see Figure 3-8), the user profile or MyStuff page (see Figure 3-9), the gallery page (see Figure 3-10), and the project-browsing pages (see Figure 3-11). The first version of the front page displayed: three projects chosen by the Scratch team; nine projects chosen by the community based on the number of views, downloads, and *love-it's*; three randomly picked “surprise projects”; and the latest three projects added to the website. In addition, the front page showed links to featured galleries, a biweekly themed gallery called Scratch Club, and a tag cloud of the website’s folksonomy. Finally, the front page also displayed the user names of the newest members and those who have recently logged in to the site.

The project page displayed the title and description of a given Scratch project with a full-featured and interactive version of the project using a Java applet<sup>8</sup> that reads the Scratch source file. The project page also allows registered users to download the source code of the project, tag the project to create folksonomies, “love it,” bookmark it by clicking favorite, and post comments. A few months after the release of the website, we added provenance information underneath each remix, showing links to its source project, and a link to a visualization of remixing connections (see Figure 3-12) in the case of “original” or “de novo” projects that had engendered remixes.

Other pages let users browse by various categories such as tags, number of views, number of love-it’s, or by galleries. Finally, the site has text-based forums where users engage in conversations, question how to do something in Scratch, or advertise their projects.

ScratchR is composed of these basic elements: a repository of Scratch projects, a database of metadata about those projects, a socially networked community, and an external application to handle a set of forums. Users can share (upload) or appropriate (download) Scratch projects to and from the repository. They can also participate in the community by tagging, commenting, bookmarking (called favorite in the interface), marking as inappropriate, and loving projects, as well as grouping projects in galleries. In addition, members can engage in discussion on the forums. These activities occur in a social network where users can connect by adding people to their list of friends. Nonregistered users of the site can browse the site on a read-only fashion. The initial navigation map included most features in the final version and was a useful guide through the implementation process.

I decided to build ScratchR using a web application framework that followed the Model View Controller pattern. This pattern was selected to make it easier to independently change the “look and feel” (view), the data manipulation (model), and the algorithms in between those two (controller). MVC also allows better organization of the code and eases collaboration with other developers. We evaluated a dozen frameworks, including Ruby on Rails, Django, Symphony, and others. I wanted a mature framework that provided basic functionality without the clutter of too many features, and took advantage of a widely used language.

---

<sup>8</sup>Special thanks to John Maloney and Brian Silverman for creating this.

[Login](#) or [Signup](#) for an account

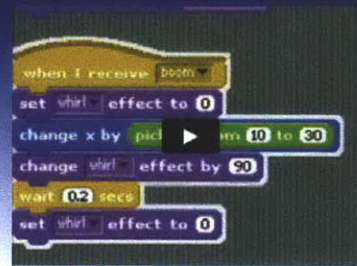
## Create and share your own interactive stories, games, music, and art

Check out the 2,677,171 projects from around the world!



To create your own projects:

[Download Scratch](#)



### Featured Projects

[See more](#)



[Pie 1s1s](#)  
by [hamstercake11](#)



[Dancing Squares...](#)  
by [dapontesgr](#)



[High School Com...](#)  
by [Cookie0227Ken...](#)

### ScratchEd



Do you help people learn Scratch? Join ScratchEd, our new online community for educators.

[Find out more](#)

### Video Tutorials



Check out our new collection of intro video tutorials.

[Learn more](#)

### Projects Selected by the\_hawk\_arisen

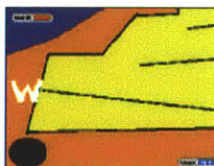
[Learn more](#)



[Cyber Defense](#)  
by [Zparx](#)



[FlashCraft beta 1.0](#)  
by [Blitz](#)



[Hawkshadow Comp...](#)  
by [lundfamily3](#)

### Scratch Tours



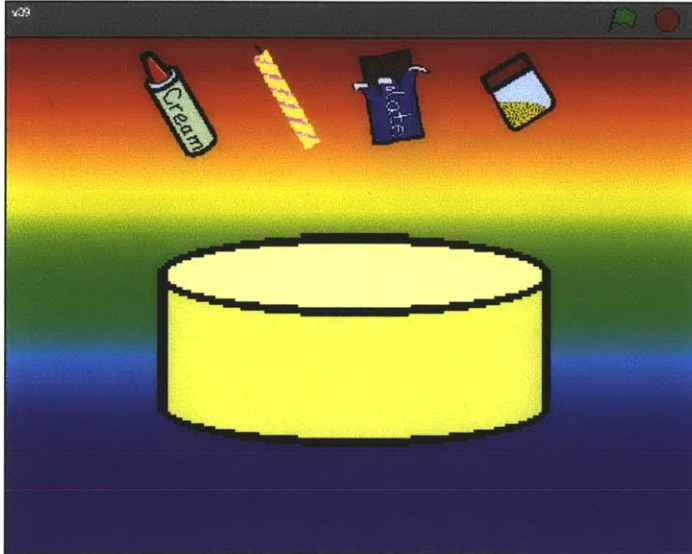
New to Scratch? Take a tour to see what Scratchers are creating and

Figure 3-7: Front page.

Login or Signup for an account

 search

## 6 Birthday



SampleProject... shared it 4 years, 8 months ago ©-© Some rights reserved

Based on [MyRedNeptune's](#) project

142 views, [2 taggers](#), 3 people love it, [1 remix](#) by 1 person, 11 downloads, in [1 gallery](#)

Love it? Add to my favorites? Flag as inappropriate?

### Comments

You need to be logged in to post comments

Add a Comment

0 / 500

Add

### Download this project!



Download the 15 sprites and 37 scripts of "6 Birthday" and open it in [Scratch](#)

### Project Notes

This is a birthday greeting card I made for my dad.

INSTRUCTIONS: Click on objects on the top of the screen to decorate the cake. You can press them in any order you like. When you've decorated the cake, you'll get a surprise. In the end you can watch my end credits if you wait five seconds after the sound stops. 8-) MORE IDEAS: -You can delete the end credits if you want to send this greeting card to someone. -You can recolor objects. -You can add more decorations to the cake. Feel free to upload your version of this greeting card. :-D

### Tags

[happy birthday](#)  
[my red neptune](#)

### Add Tags

Add

### Link to this Project

Embed



Figure 3-8: Project page.

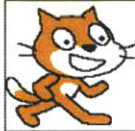


Login or Signup for an account

 search

**SampleProjectsTeam**

Location:  
 cambridge  
 MA  
 United States



No friends yet.

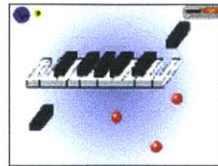
**Galleries**

-  [Scratch 1.2 Pote...](#)
-  [Amazing art](#)
-  [Me and my friend...](#)
-  [ELBARTO](#)

[See more](#) ▶

**SampleProjectsTeam's Projects**

[Subscribe](#)



[9 could simplify ...](#)  
 Comments: 4



[4 ForWodunne](#)  
 Comments: 3



[a CanYouDance - e...](#)  
 Comments: 6



[2 Slideshow](#)  
 Comments: 5



[2 Tamika](#)  
 Comments: 3



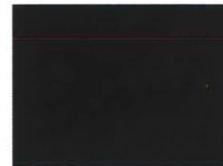
[1 Say sorry to yo...](#)  
 Comments: 4



[3 Mick](#)  
 Comments: 4



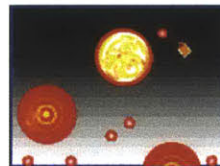
[5 WORDO](#)  
 Comments: 5



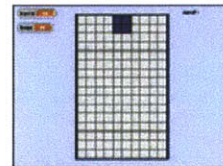
[5 SoundGraph](#)  
 Comments: 2



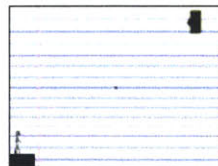
[7 WeatherSimulati...](#)  
 Comments: 4



[Bouncing Balls wi...](#)  
 Comments: 3



[7 Tetris](#)  
 Comments: 8



[6 Doodle](#)  
 Comments: 85



[My Little Game \(r...](#)  
 Comments: 7



[Pastor 1.0](#)  
 Comments: 31

1 | 2 | 3 | 4 | 5 | 6

Figure 3-9: Profile page (My Stuff).

[Login](#) or [Signup](#) for an account

## Newest Projects in Scratch 1.2 Potential Sample Projects

Sort by: [creator](#) | [title](#) | [creation date](#) | [addition date](#) |

[Subscribe](#)



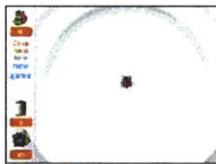
[7 Wodunne'sWo...](#)  
by [SampleProject...](#)



[6 PolarbearSc...](#)  
by [SampleProject...](#)



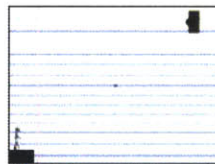
[8 DayDream](#)  
by [SampleProject...](#)



[5 Bug On a Pl...](#)  
by [SampleProject...](#)



[Pastor 1.0](#)  
by [SampleProject...](#)



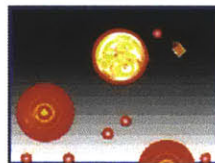
[6 Doodle](#)  
by [SampleProject...](#)



[2 Tamika](#)  
by [SampleProject...](#)



[a CanYouDance...](#)  
by [SampleProject...](#)



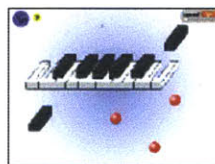
[Bouncing Ball...](#)  
by [SampleProject...](#)



[5 SoundGraph](#)  
by [SampleProject...](#)



[3 Mick](#)  
by [SampleProject...](#)



[9 could simpl...](#)  
by [SampleProject...](#)



Gallery owner: [SampleProject...](#)



Created: 4 years, 8 months ago

### Gallery description

We are trying to figure out which new projects should go in the sample projects library. This is a set of projects not made by us that we are considering for inclusion.

Figure 3-10: Gallery page.

## What the community has been remixing

Do you want to know how to [upload your project](#)?

- [most recent](#)
- [most viewed recently](#)
- [most loved recently](#)
- [most remixed recently](#)



### [I am against bullying. Please re-post if you care.](#)

By: [Ludburghmdm](#)  
Views: 168 | Lovelt's: 8 | Remixes: 73  
Description: Please repost if your are also against bullying.



### [A message to everyone...](#)

By: [bballuke](#)  
Views: 355 | Lovelt's: 11 | Remixes: 63  
Description: I know not everybody does this, but if you do, please stop. When you remix, be sure to give credit to me (BballLuke). Remember, FRONT PAGE!!! I want to thank everybody who remixed. This is now on the front page, and it only has 234 views (on the original) Log: July 13 Front Paged! July 15: ... [show more](#)



### [COLORING CONTEST 4](#)

By: [-Nightfire-](#)  
Views: 246 | Lovelt's: 9 | Remixes: 42  
Description: YOU MUST MUST MUST MUST MUST SUBMIT YOUR ENTRY HERE:  
<http://scratch.mit.edu/galleries/view/171532> Rules: -All programs are allowed -Don't do any major changes to the line art -Adding accessories is ok (better chance of winning!) -BE CREATIVE -Don't add too many random colors. Try to come up with a ... [show more](#)



### [HEROZ contest](#)

By: [yakman](#)  
Views: 163 | Lovelt's: 8 | Remixes: 29  
Description: rules explained in project



### [Minecraft CONTEST \[READ NOTES\]](#)

By: [BIG-red-BUTTON](#)  
Views: 358 | Lovelt's: 8 | Remixes: 28  
Description: EPIC: <http://scratch.mit.edu/projects/BIG-red-BUTTON/2667545> Music \* Mortal Kombat LOL CONTEST ENDS after 15 days ----- HOW TO ENTER? -remix this project -delete "sprite 1" -edit the stage(color,draw) -upload your entry -

Figure 3-11: Project-browsing page.



Figure 3-12: Three remixing visualizations.

This eliminated Ruby on Rails, as it was too new and the language was not as popular as others were. I seriously considered Django (which is based on Python), but I decided to go for a framework based on PHP because of its popularity, which would make it easier to find software developers, especially undergraduates. Picking a PHP-based framework made it easy to choose the LAMP stack: Linux for the underlying OS, Apache for the web server, and MySQL for storing the data. Five years later, I do not regret choosing this infrastructure.

The overall architecture of the web application is described in Figure 3-13, and the initial Entity Relationship Diagram (ERD) of the database is in Figure 3-14. For an updated version of the ERD please refer to Appendix A.

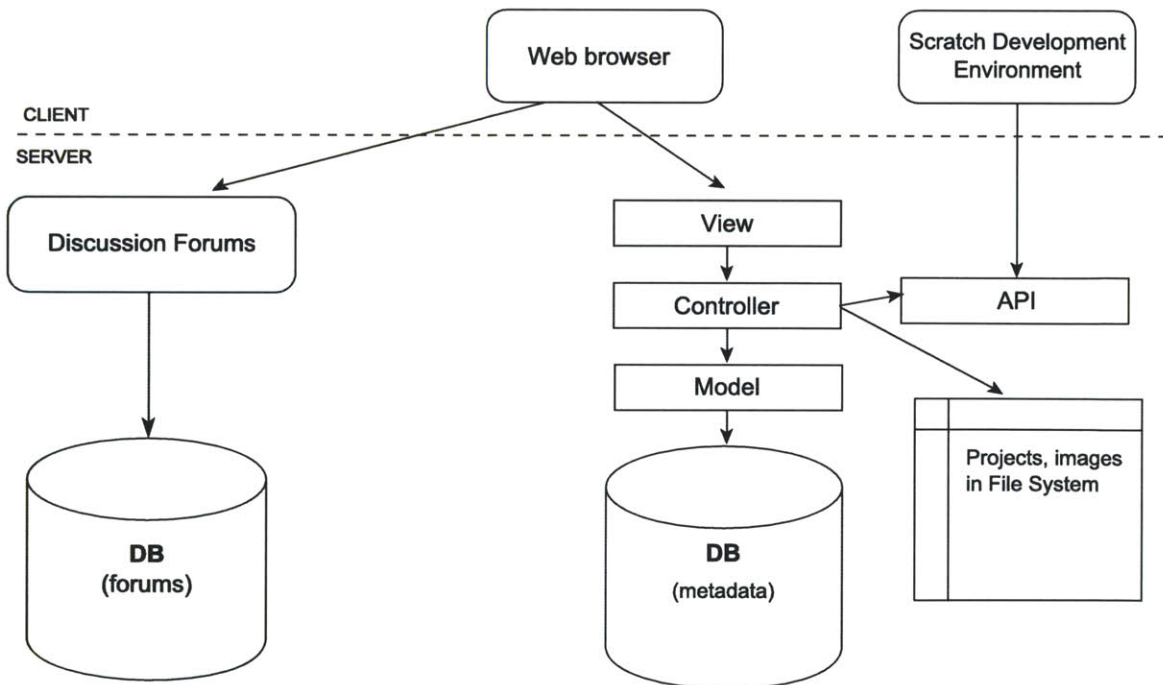


Figure 3-13: Initial architecture of ScratchR.



to creators in accordance with the license used for Scratch projects. This is the focus of later analyses.

### 3.1.5 Scale

Five years after its official release, the Scratch Online Community website handles more than 10,000,000 page views and 600,000 unique visitors every month (see Figure 3-15). This web traffic is more than half the page views of websites such as [newsweek.com](http://www.newsweek.com)<sup>9</sup>. Every second, the website receives up to 180 requests and it transfers 4MB. Five years since its inception, 2,426,894 project have been uploaded, representing about 2TB of files stored in the file system and 1GB of metadata stored in a MySQL database. To handle this level of activity, ScratchR, the website's underlying platform, had to implement the following infrastructure changes<sup>10</sup>:

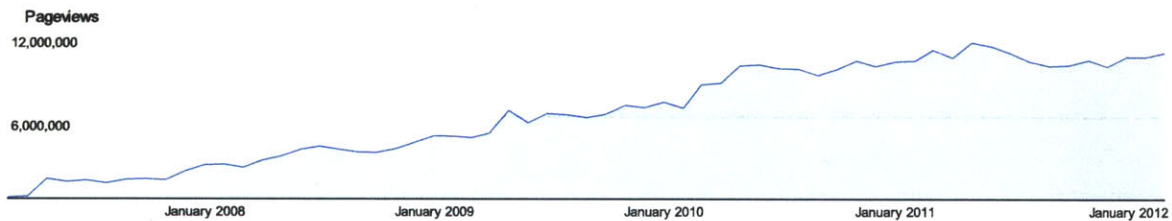


Figure 3-15: Five years of monthly pageview counts.

#### *Query optimization:*

The first bottleneck we encountered was complex database queries. First, we tried to simplify them but for many, adding new indices was helpful. The first thing we did was to identify the slowest queries, which helped us decide which database indices to add, or, occasionally, which queries to improve.

#### *Static content caching:*

We used the Varnish (see Figure 3-16) caching engine to avoid having to read each image from the server's hard drive and file many times, especially those heavily used such as popular projects and the images part of the UI elements of the website.

#### *Dynamic content caching:*

We used the Memcached engine (see Figure 3-16) to store the result of complex database queries in memory. Each query result would reside in memory for a specific period, specified in the configuration files, and based on its complexity. For example, the query to generate the tag cloud that appears on the front requires up to a minute to generate, as the front page is requested many times per second, and consequently puts severe stress on the server. Memcached will hold the results of this tag cloud, which does not change often, for several hours.

---

<sup>9</sup>04/2011 data from <http://www.quantcast.com/newsweek.com>

<sup>10</sup>Special thanks to Vladimir Vuksan for his help on this.

### Hardware:

The website originally ran on a single machine that hosted both the Apache and MySQL, the HTTP and database servers. As demand grew, we had to split those two services across two machines. We also had to add a server for caching and another for backups. Furthermore, the server for each service was fine-tuned depending on its use. The database server was configured with a large amount of memory and speedy disks, while the Web server got more processors.

### Internationalization:

More than half of the website's visitors came from outside the United States, this motivated us to enable translations for the user interface using the system Pootle, which allowed anyone to help with the translation. In addition to this, the projects displayed on the front page of the website were customized to match the country of the person visiting the website. This functionality was enabled only for Mexico because we had a local partner there.

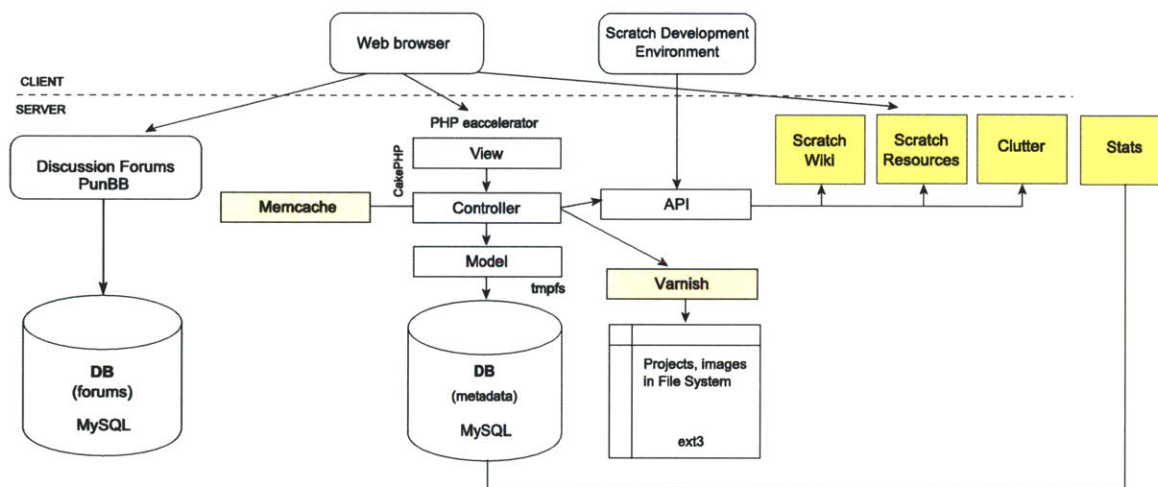


Figure 3-16: Scaled-up Architecture of ScratchR.

## Scaling Community Management

Scaling up was not only a technical challenge, it was also a challenge for moderation and community building. As people signed up and uploaded their first projects, I would welcome them to the community by commenting on their work, asking them questions about their projects, or making suggestions on how to improve it. Although I have not analyzed this quantitatively, I am convinced these early efforts helped create a sense of community. I was especially made aware of this when I started seeing others who I had never met mimicking this community-building behavior. Some members became so active that they were invited to become moderators of the discussion forums, and others became curators helping select a few projects on the front page (72 accounts were marked as curators as of March 2012). Because of popular demand, some in these roles have been selected through community vote.





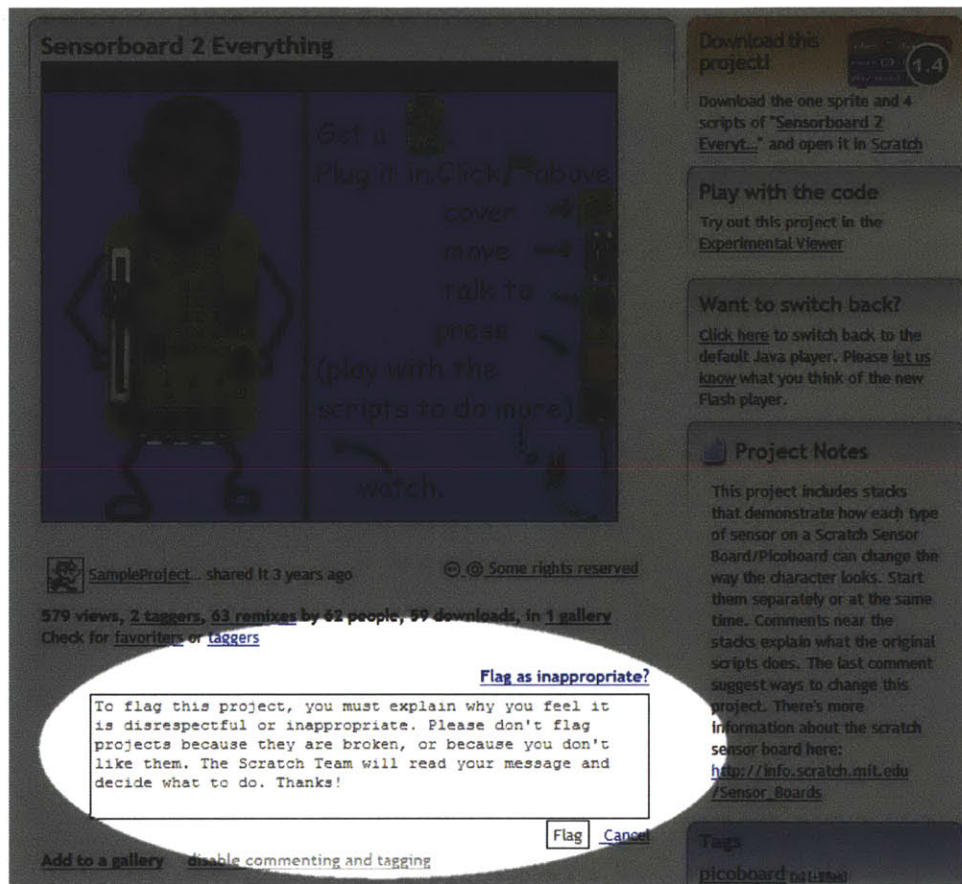


Figure 3-18: Community members can flag projects as “inappropriate”.

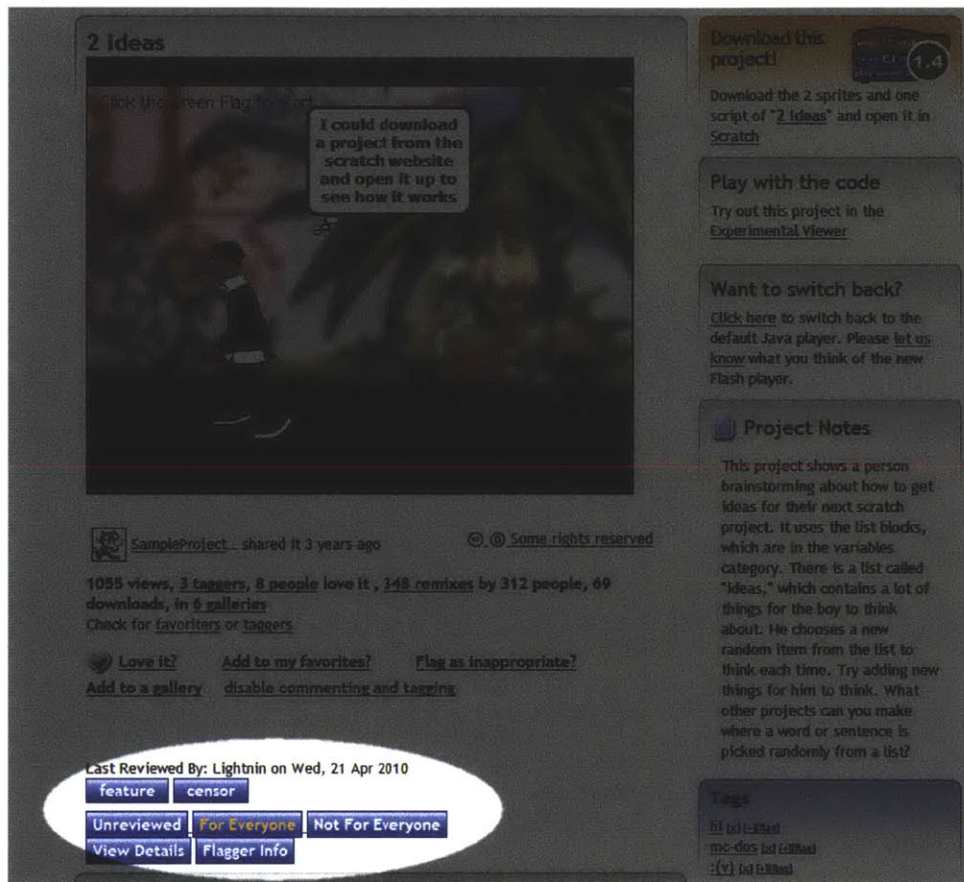


Figure 3-19: Administrators can censor, demote, protect, or feature a project.

You have [403 messages](#)

## Manage Users

[Home](#) | [User Bans](#) | [IP Bans](#) | [Announcements](#) | [Search](#) | [Admin Tags](#) | [Integraflag](#)

### Integraflag - open flags (open | [review](#) | [closed](#)) - [Search](#) - [Stats](#)

Flags	Flagged User	Action
<p>Comment (on <a href="#">Pie 1s1s</a>) removed - flagged by multiple users.                      Flagger: <a href="#">[0/2]</a> - <a href="#">Comments</a>   <a href="#">[0/0]</a> - <a href="#">Comments</a> (7/20/12, 6:05 am)</p> <p>accept credit card and so mang payment == = <a href="#">Air Jordan</a>                      (1-24) shoes \$35 UGG BOOT \$50 Nike shox (R4, NZ, OZ, TL1, TL2, TL3) \$35                      Handbags ( Coach Lv fendi D&amp;G) \$35 T-shirts (polo, ed hardy, lacoste)                      \$16 Jean (True Religion, ed hardy, coogi)\$34 Sunglasses ( Oakey, coach,                      Gucci, Armani)\$15 New era cap \$16 Bikini (Ed hardy, polo) \$18 FREE                      SHIPPING == = <a href="#">[redacted]</a></p> <p>Posted: 2012-07-19 23:18:18</p>	<p><a href="#">[redacted]</a> [0/0]  <a href="#">Comments</a>   <a href="#">IP</a>   <a href="#">Ban</a></p>	<p><a href="#">BC1</a> <a href="#">BC2</a>  <a href="#">Con</a> <a href="#">Res</a>  <a href="#">Rev</a> ✓                      Notes:0</p>
<p>This account was created on a network used by the following banned accounts in the past 30 days:  <a href="#">[redacted]</a> [3/4] - <a href="#">Comments</a> (7/20/12, 5:10 am)</p> <p>IP actions: <a href="#">78.120.117.240</a></p> <p>Registered: 2012-07-20 05:10:07</p>	<p><a href="#">[redacted]</a> [0/0]  <a href="#">Comments</a>   <a href="#">IP</a>   <a href="#">Ban</a></p>	<p><a href="#">Rev</a> ✓                      Notes:0</p>
<p>Comment (on <a href="#">[redacted]</a>) removed - flagged by the project creator.                      Flagger: <a href="#">[0/2]</a> - <a href="#">Comments</a> (7/20/12, 4:46 am)</p> <p>farts</p> <p>Posted: 2012-05-11 13:51:18</p>	<p><a href="#">[redacted]</a> [0/0]  <a href="#">Comments</a>   <a href="#">IP</a>   <a href="#">Ban</a></p>	<p><a href="#">BC1</a> <a href="#">BC2</a>  <a href="#">Con</a> <a href="#">Res</a>  <a href="#">Rev</a> ✓                      Notes:0</p>
<p>Comment (on <a href="#">[redacted]</a>) removed - flagged by multiple users.                      Flagger: <a href="#">[0/0]</a> - <a href="#">Comments</a>   <a href="#">[1/1]</a> - <a href="#">Comments</a> (7/20/12, 4:31 am)</p> <p>accept credit card and so mang payment == = <a href="#">Air Jordan</a>                      (1-24) shoes \$35 UGG BOOT \$50 Nike shox (R4, NZ, OZ, TL1, TL2, TL3) \$35                      Handbags ( Coach Lv fendi D&amp;G) \$35 T-shirts (polo, ed hardy, lacoste)                      \$16 Jean (True Religion, ed hardy, coogi)\$34 Sunglasses ( Oakey, coach,                      Gucci, Armani)\$15 New era cap \$16 Bikini (Ed hardy, polo) \$18 FREE                      SHIPPING == = <a href="#">[redacted]</a></p> <p>Posted: 2012-07-19 23:19:20</p>	<p><a href="#">[redacted]</a> [0/0]  <a href="#">Comments</a>   <a href="#">IP</a>   <a href="#">Ban</a></p>	<p><a href="#">BC1</a> <a href="#">BC2</a>  <a href="#">Con</a> <a href="#">Res</a>  <a href="#">Rev</a> ✓                      Notes:0</p>

Figure 3-20: Administrators' control panel to deal with community-generated flags.

## 3.2 Participation Patterns

Once I was ready to release the initial version of the website, I decided to first test it with users that I could meet face to face. The opportunity presented itself as I was invited to run an eleven-week Scratch workshop<sup>11</sup> for middle school students through an organization called Citizen Schools<sup>12</sup>. The workshop would help them learn how to create projects with Scratch, and share their projects on the newly created website. The workshop participants helped me identify bugs and usability problems in general. The experience also gave me insights into the social dynamics of what I hoped would be an active online community. For example, it helped me realize how the projects and people on the website were going to be even more important than the technical infrastructure behind them.

On May 15, 2007, a few weeks after this workshop ended, the website and the Scratch application were publicly announced. Several news outlets and social news websites featured the Scratch website on their front pages (Fildes, 2007; Zonk, 2007; Johnson, 2007). In a matter of hours, the server and the website could not handle the traffic and the website went down several times. This is when problems of scale became apparent, something I tackled with the techniques described in the previous section.

Visitors to Scratch website come from all over the world. Based on Google Analytics<sup>13</sup> data from March 2007 to March 2012, the typical visitor to the website spends 5 minutes 31 seconds, and visits 6.79 pages. Also, of the 56 million of visits, almost half came from the United States, 48.55%. Other visitors came primarily from the United Kingdom 11.38%, Canada 4.41%, Australia 2.87%, Germany 2.07%, Brazil 2.03%, South Korea 1.68%, Taiwan 1.39%, Mexico 1.18%, and Spain 1.15%. In terms of cities, London accounted for most visitors at 2.41%, New York at 1.12%, Sydney at 0.86%, Melbourne at 0.77%, Hong Kong at 0.63%, Seoul at 0.55%, Los Angeles at 0.54%, Chicago at 0.49%, and Minneapolis at 0.48% (see Figure 3-21). In terms of demographics from Quantcast<sup>14</sup>, 49% of visitors are female, 28% are less than 18 years old, and 21% between 18 and 24. In terms of ethnicity, 77% are Caucasian, 8% African American, 5% Asian, 8% Hispanic, and 2% other.

More specifically with regards to the 1,068,502 accounts people signed up for, 36% (392,814) self-report as female and 29.41% (314,287) shared a project on the website. 37.50% (117,857) of those project sharers self-report as female, and shared a mean of 7.24 projects—compared to 8.01 project for males.

Another interesting fact is that 77.99% of projects (1,892,778) were shared by accounts within the first 30 days of creation, 26.96% (654,331) within the first week, and 16.90% (410,264) within the first day (see Figure 3-23).

In addition, we find that the median and mean age of project sharers has remained stable through the 5 years at 12 years old, and 15 years old respectively (see Figure 3-24, Figure 3-

---

<sup>11</sup>Special thanks to Rachel Gerber for her help running this workshop.

<sup>12</sup>“Citizen Schools partners with middle schools to expand the learning day for children in low-income communities countrywide. By drawing thousands more citizens into schools each year, we’re promoting student achievement, transforming schools, and reimagining education in America.” <http://citizenschools.org/>

<sup>13</sup><http://google.com/analytics>

<sup>14</sup>Generated on Jul 13, 2012 <http://quantcast.com/scratch.mit.edu>

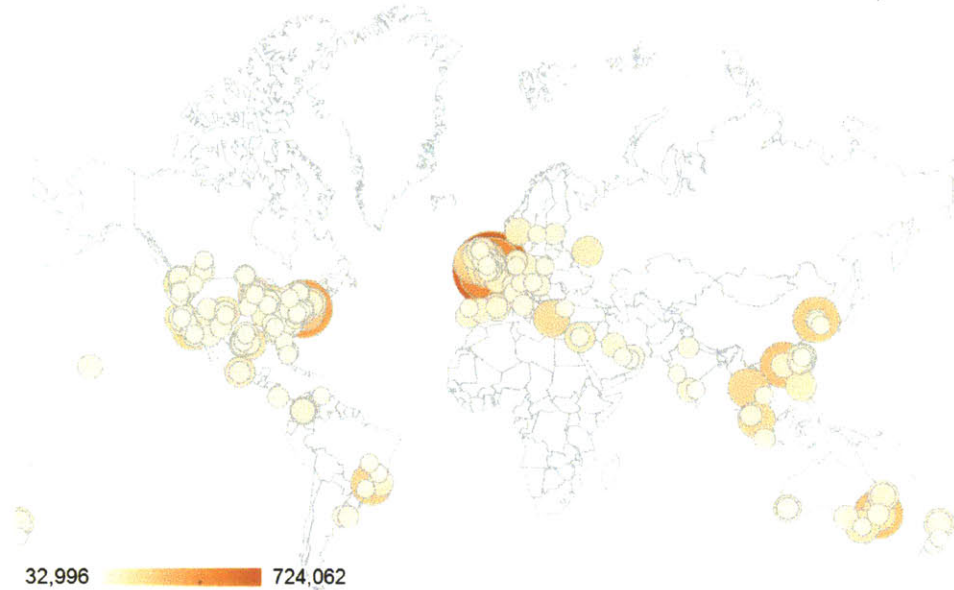


Figure 3-21: World map showing cities with most number of Scratch visitors.

25). We also find that the majority of project sharers are in elementary and middle school (Figure 3-22). Although a small percentage of projects are shared by adults (16.18% or 392,705 projects were shared by people between self-reported 22 and 60 years old), I have identified, and even met in person, several adult computer hobbyists and educators who create projects in Scratch. Though many of them know other professional programming languages, they find Scratch a fun activity or do it as part of their teaching practice. Some older members of the community have emerged as mentors who help the beginners and provide advice. One of them joined the team of people doing community moderation.

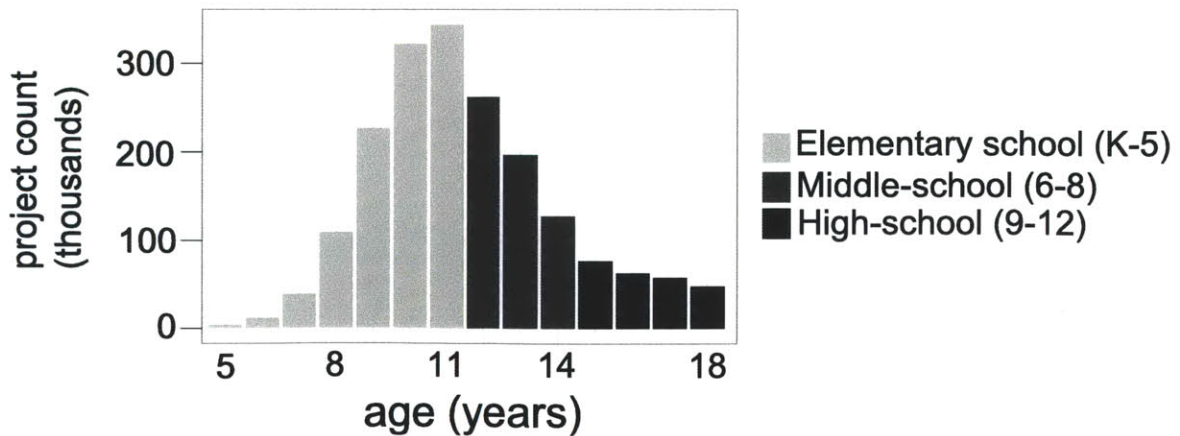


Figure 3-22: Distribution of ages of project sharers.

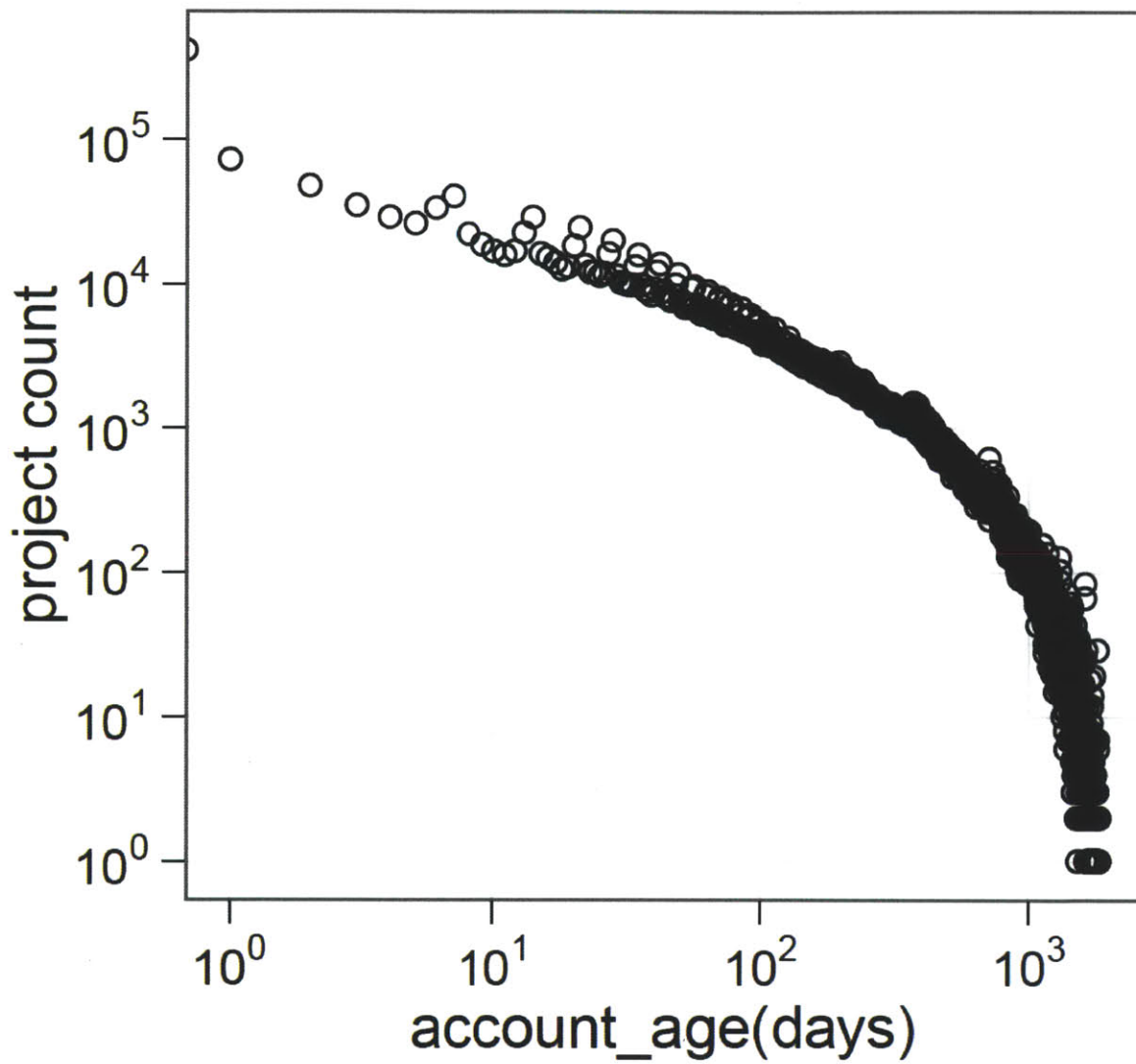


Figure 3-23: Distribution of projects by account age.

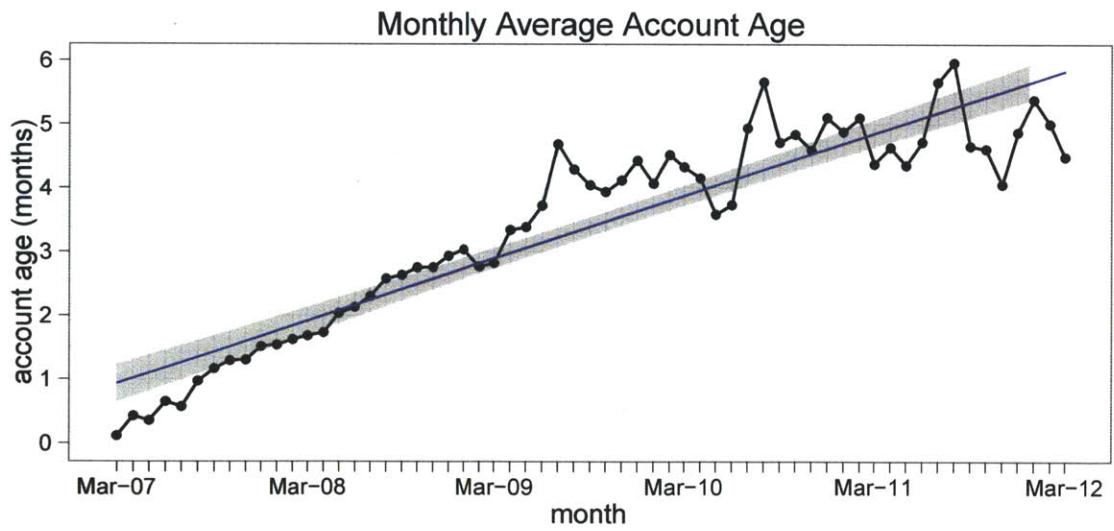
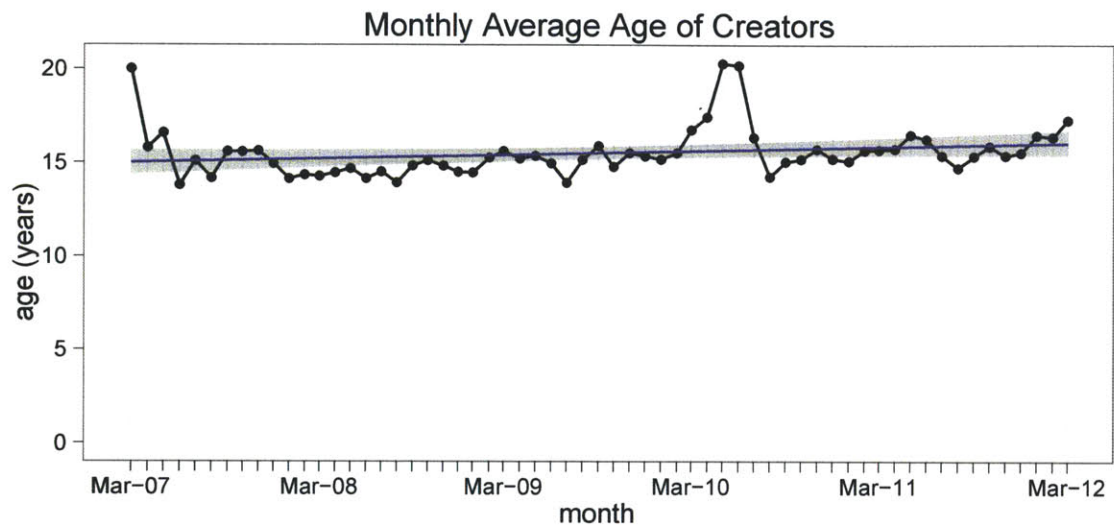


Figure 3-24: Project creator's age and account age monthly mean.

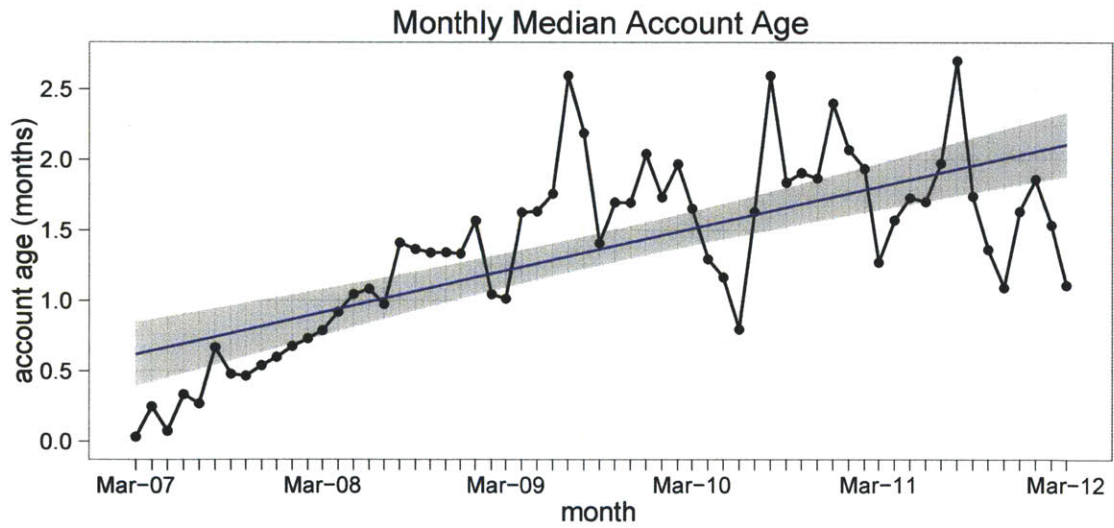
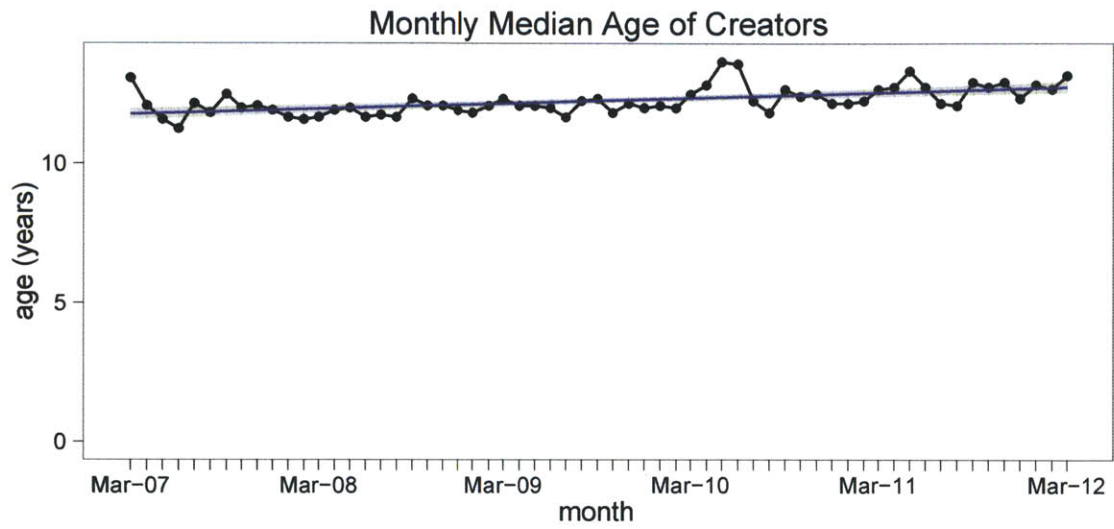


Figure 3-25: User age and account age monthly median (project creators only).



### 3.2.1 Project making

During the five years after the release of the beta version of the Scratch website, we have seen a steady increase in the number of projects shared every month and the unique creators who upload them (see Figure 3-26). Between March 5, 2007, to April 1, 2012, community members have contributed 2,426,894 projects.

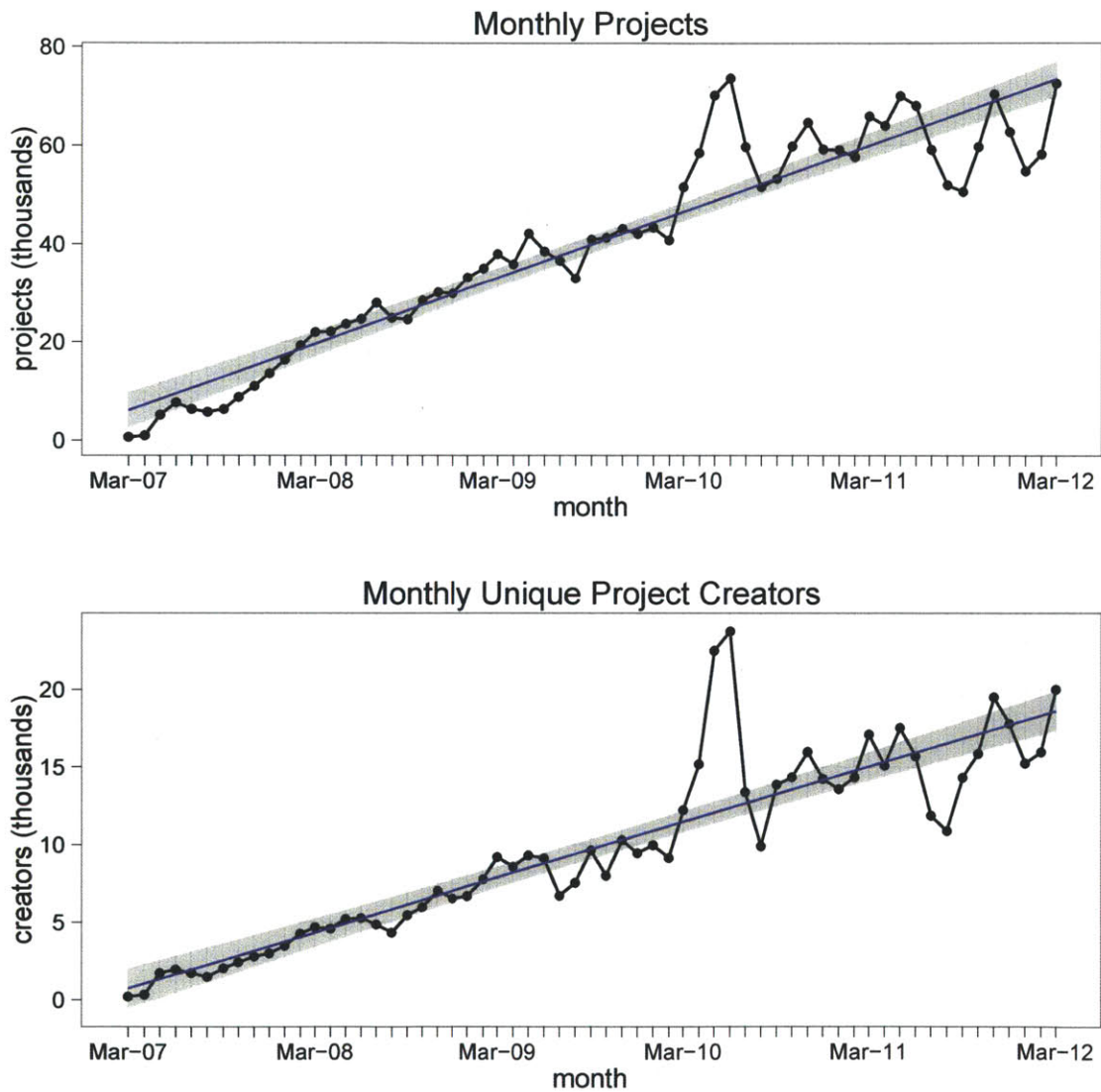


Figure 3-26: Projects and creators per month.

Projects vary from interactive greeting cards to physics simulations, animations of popular songs, and homemade video games, just to name a few. Scratch projects are organized in sprites (for example, a character in a game). Each sprite has a set of “costumes,” images that represent its various visual states; for example, a sprite of a bird flying could have many

costumes, each representing the various positions of the wings. Sprites can also have sounds associated with them; these sounds can be either recorded with the microphone or imported from the hard creator’s hard drive. Finally, “scripts,” stacks of visual programming blocks, control the sprites’ behavior.

I was interested in knowing what was inside the projects people made, so I compiled a large database with information about the most common components of each version of every project uploaded to the Scratch website. Among other things, this database has the text-based human-readable representation of the scripts for each sprite (see Figure 3-27). A sprite can have zero or more scripts, so the database was created by extracting each script associated with every sprite and created a new database table for it. This new table has a record for every script that comes with its human-readable version, the ID of the project, its version number, and the ID of the sprite, among other fields. I added a column to store a version of the script without arguments. This arrangement treats scripts possessing different arguments but the same block sequence as equals. This new database table of scripts has more than 16 million sprites, 47 million scripts (90% of projects have scripts), and 273 million lines of code—as a comparison, the Linux Kernel had 15 million lines of code in 2011 (Paul, 2012). For the sprites, most had cryptic names and there is not an easy way of representing them, so I decided to look at their costumes or images as a proxy to determine most common sprites. These misses the behavior and blocks of the sprites, but because I was already grabbing the most common scripts I decided that getting the most common images would help unveil the content of Scratch projects.

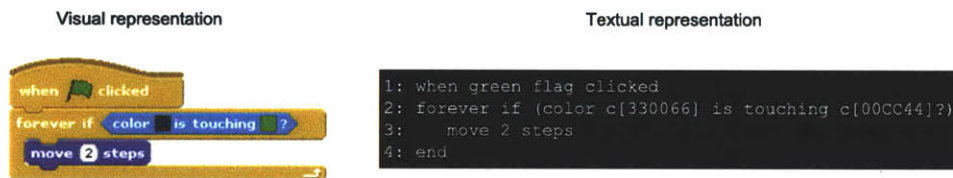


Figure 3-27: Example of a Scratch script in visual and textual form.

## Code

Scripts are groups of blocks that are triggered by some event, for example, by pressing a keyboard key or by simply starting the Scratch project. When composing scripts, users have several type of block they can use, from “control” blocks like “if-then-else” statements, to “looks” blocks like “switch to costume.” Figure 3-29 shows how often different categories of blocks are used. Similarly, the “script cloud” in Figure 3-28 shows the most common scripts.

This analysis shows that the most common *scripts* involve looks manipulation such as hiding/showing a sprite and switching its costumes. This is probably because controlling the display on the screen is useful and necessary for most types of projects, from games to animations. In addition, these scripts often come in pairs: for every “hide,” one would expect a “show.”



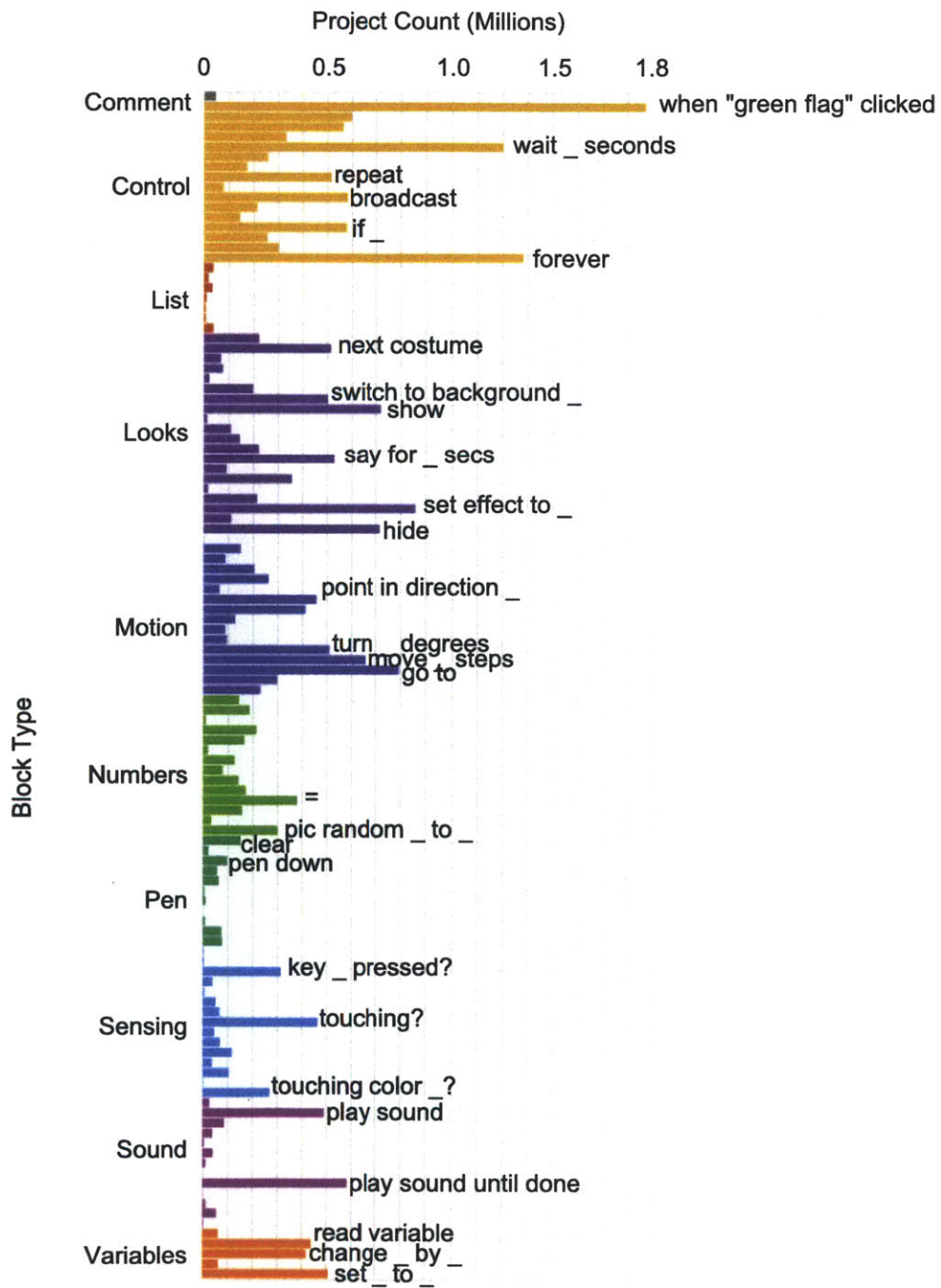


Figure 3-29: Histogram of block usage. Each color represents a block category. Each bar represents a block.

Despite not being obviously interactive, the 27<sup>th</sup> most common script (When “Green Flag” clicked; forever set  $x$  to [ ] + [ ] + [ ]) represents a form of interactivity because one of its arguments is a variable changed by pressing the arrow keys. As we can see in this project (the first to use this script), these blocks are typically used to control the horizontal position of background elements on a scrolling background game.

I was somewhat surprised to find a script related to sound ranking so highly; it is likely that animations and games often have some sound playing continuously in the background. Closer examination produced a more surprising revelation: the sounds looped more frequently are not music files imported into Scratch (namely, commercial songs) but recordings created within Scratch using a microphone. The most common sound name played with this script is “recording1” (3.82%) followed by “one1” (1.08%).

The scripts in gray in the script cloud are stand-alone “single hat” blocks. They represent a script with a trigger to start but without anything underneath to execute. The decision to include them was difficult; technically, they are scripts but because they have no blocks underneath, they do not have any effect on the project. Ultimately, their inclusion seems merited as an indication of how often people drag a hat block and leave it unused. Compared with other languages, Scratch is forgiving and lets people do this without big repercussions. These unused hat blocks possibly represent moments of tinkering and experimentation, something valued by Scratch creators and administrators.

Conversely, I decided to leave out the comment block by itself from the script cloud. Partly because scripts are groups of blocks with a “hat block” on top while the comment block is a single block that does not get executed. However, if one were to count that block as a script, it would be in the 10<sup>th</sup> position in the list (0.68%). The use of the comment block was both surprising and encouraging, in part because it was a recent addition. Many projects from 2007 and 2008 lacked the comment block as an option. But when comparing its use against other blocks, not against scripts, it is one of the least used blocks.

## Media

An “image cloud” with the most common costumes can be seen in Figure 3-30. All the images in this list are images that come with Scratch itself. However, they represent less than one percent (1%) of the total images. The reality is that the distribution of images follows a long tail distribution where many images are used only once or twice.

## Complexity

Using the five years of longitudinal user log data, we can see how users progress as they continue using Scratch for a while. For example, one can get a rough measure of the complexity of a Scratch project by looking at its number of sprites, scripts, blocks, diversity of blocks (distinct unique blocks), costumes, and sounds. Each of these measures different types of skills, from media-centric projects (i.e., those with more costumes and sounds) to more code-centric projects (i.e., those with more scripts or blocks). A simple time series of

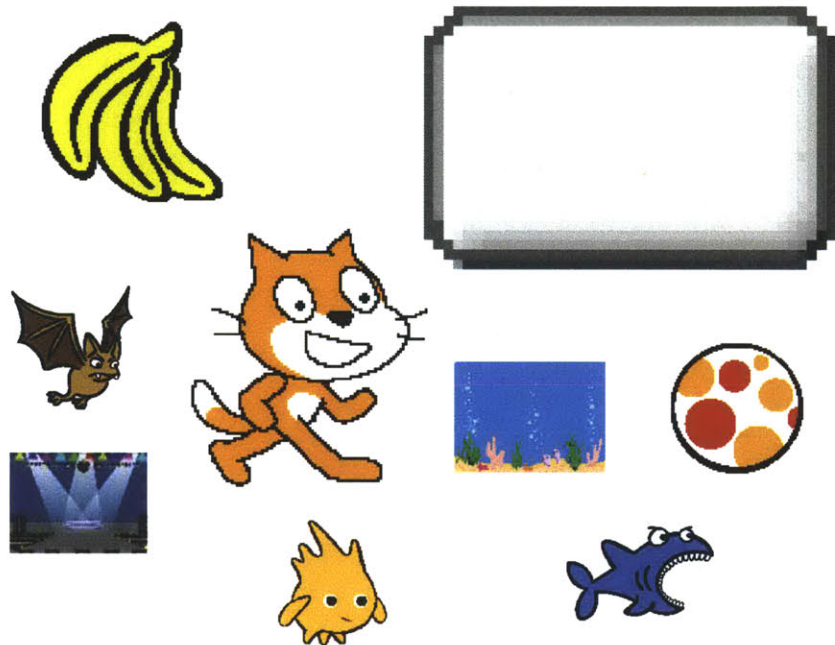


Figure 3-30: “Image cloud” showing the most frequently occurring costumes in a corpus of 47 million

each of these metrics shows that the metric for general complexity of a project, i.e., sprites, increases as the age of the account (Figure 3-31). We see the same for code complexity but different for media complexity. Because mean values might be skewed due to extreme outliers, we plotted the median values and we observed the same patterns, especially for code-centric complexity. These results do not control, however, that as users grow up, they might become more adept with Scratch or any other tool merely by virtue of being older.

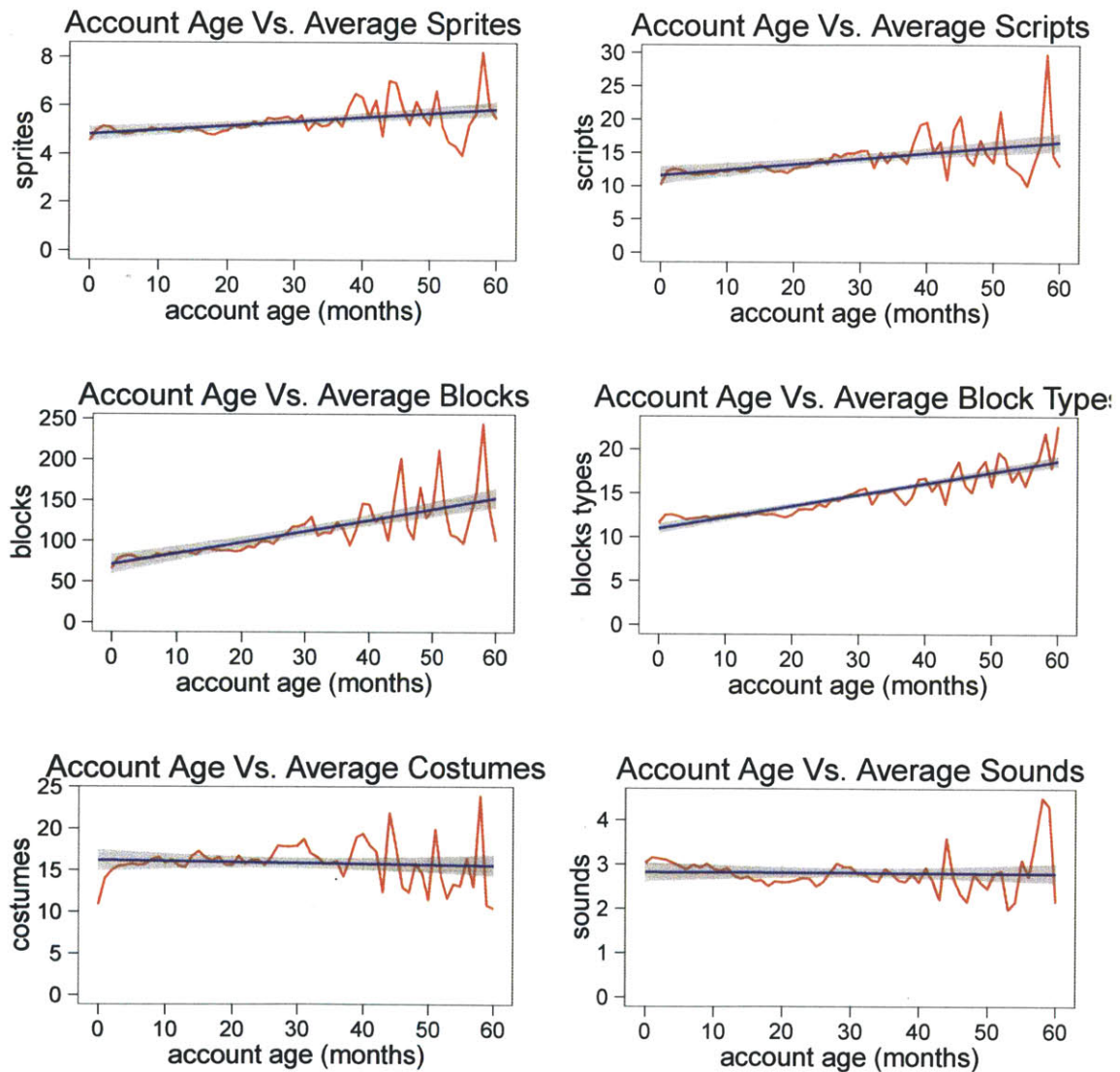


Figure 3-31: Project complexity and user's and account's age.

### 3.2.2 Interactions

Besides project making, community members come to the Scratch website to interact with others. Using the framework of youth engagement by Ito (2010), I have seen community members see the Scratch community as a place to “geek out” through creating projects, “mess around” by playing others’ projects, and “hang out” by participating in the discussion forums, often on non-programming subjects. They feel part of a community that is more than just programming (see Figure 3-33 for this model of participation).

These activities translate to 10,018,442 comments posted on projects, 57,690,455 views on projects, 1,716,538 loveits (equivalent of likes on Facebook), 1,169,248 favorites (equivalent of bookmarking), 96,525 people who have tagged a project, 4,886,289 project downloads, 134,565 galleries created, and 3,741,009 comments posted on galleries. The table below (Table 3.1) shows that these attributes are spread across projects.

As the Scratch Online Community has grown, one of the main challenges has been maintaining the initial level interaction with submitted projects. For example, the community has noted how the number of views per project has decreased. For example, the community-driven wiki notes the following:

*One could argue that more views were given [to projects] back then, as there were fewer Scratchers to choose from. Conversely, you could argue that fewer views were given back because it was a smaller community.*

A time series plot of the number of interactions per project confirms this observation (see Figure 3-32). This remains one of the challenges for future research.

## 3.3 Discussion

People often ask how the Scratch Online Community became “successful.” Of course, “success” can mean many things to various people. Assessing the success of an online community is a rich area of research explored by others (Preece, 2000; Kittur and Kraut, 2008; Kollock, 1998; Burke and Kraut, 2008a,b; Beenen et al., 2004) and slightly tangential to the goal of this document. The goal here is merely to describe how the Scratch Online Community has achieved some of the desired outcomes, and has struggled to achieve others. Broadly speaking, two measures of success, *size* and *engagement*, are considered here.

#### *Size:*

The simplest and perhaps most naive measure of success is based on the number of registered users, contributed content, page views, and visits to a website. With more than one million users, two million contributed projects, ten-million page views and one-million visits, the Scratch Online Community is big for an academic research project but relatively small compared with similar online spaces—consider YouTube’s 14 billion videos (comScore, 2010).

#### *Engagement:*

A more nuanced and difficult means to assess the success of an online community is



Table 3.1: Table of project attributes

Attribute	<i>N</i>	mean	median	sd	mode
views	16.35 6		89.771	1	
favoriters	0.485 0		5.331	0	
loveits	0.707 0		7.218	0	
versions	1.238 1		1.123	1	
flags	0.016 0		0.207	0	
sprites	6.016 3		14.345	1	
scripts	17.46 4		291.424	1	
visibility	0.795 1		0.404	1	
remixes	0.355 0		9.852	0	
remixers	0.266 0		6.638	0	
downloads	1.835 0		20.857	0	
downloaders	1.816 0		20.376	0	
comments	2.92 0		16.641	0	
commenters	1.686 0		8.921	0	
galleries	0.415 0		2.093	0	
blocks	117.19228		548.691	0	
block types	12.848 9		11.52	0	
costumes	19.334 7		71.282	3	
sounds	3.907 2		12.5	1	
strings	25.427 2		548.013	0	
saves	6.702 1		21.111	0	
days to share	39.661 0.001		352.125	0	
days to remix	4.79 0		47.375	0	
user age (years)	16.79312		15.201	11	
account age (days)	139.66844		219.625	0	

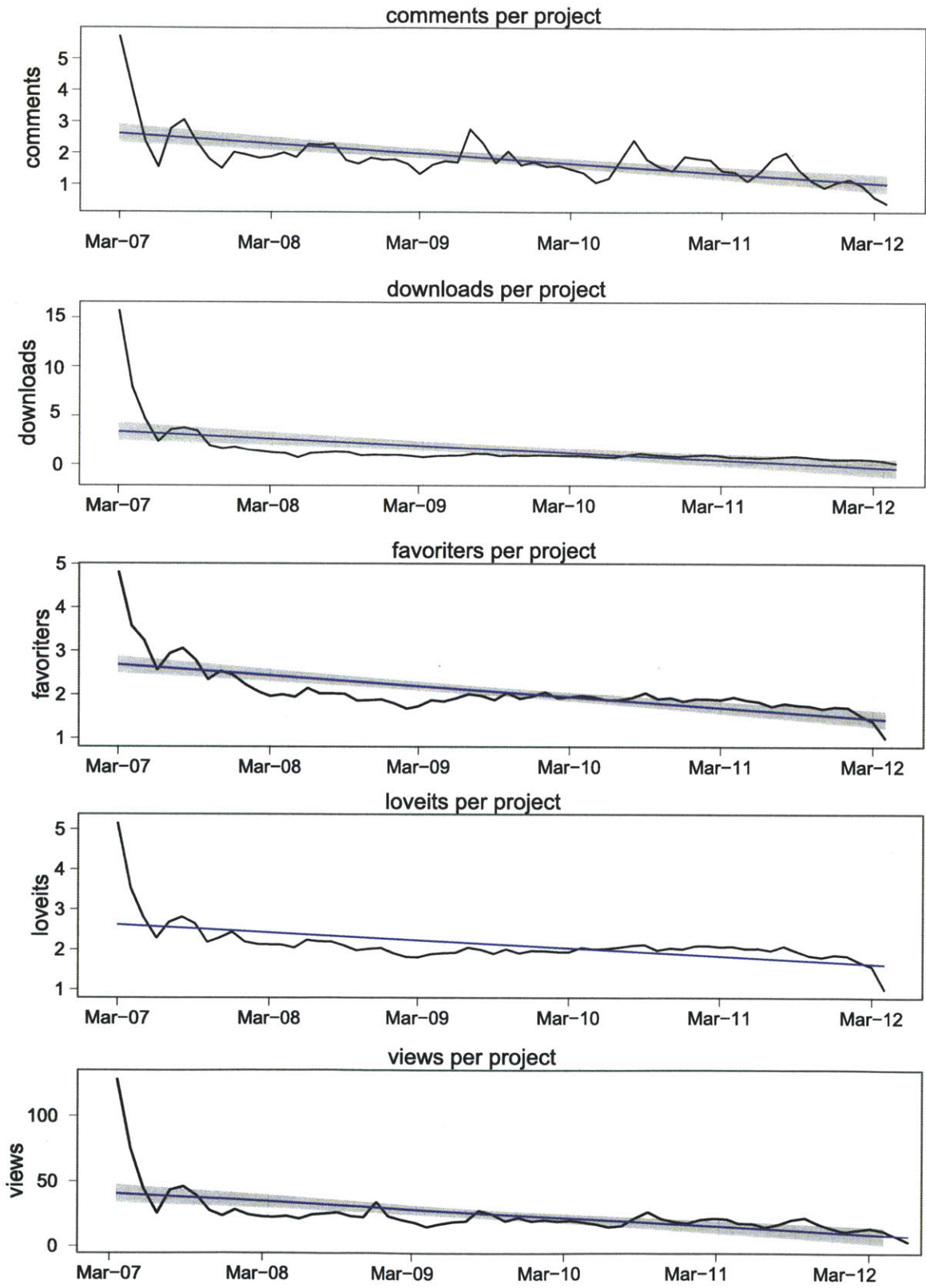


Figure 3-32: Monthly mean number of interactions per project

the diversity and richness of its members' interactions. Some rich interactions, documented in stories of members of the online community, show that many participants consider the Scratch Online Community not only a website where they upload projects but also a space where they like to hang out to meet and interact with peers.

It is difficult to know exactly what contributed to the success of an online community but below follows a description of the set of broad sociotechnical steps that possibly contributed to the success of the Scratch Online Community. Although they might be the result of post hoc rationalization, they might provide insights into others trying to work on this space.

*High-quality creation tool:*

The online community success is undeniably linked to the quality of the Scratch authoring tool. Members of the Lifelong Kindergarten Group<sup>15</sup> have spent countless hours over the years perfecting the Scratch development environment. Every pixel in the screen has probably had more hours of work than any other aspect of the Scratch experience. This attention to detail in the Scratch application has greatly contributed to the solid user experience of Scratch, even at the expense of agility.

*Bootstrapping:*

Many online social spaces face the problem of adoption. Others have already described the challenges of benefiting from the network effect (Porter and Donthu, 2008). How does one get people to participate in an online community before there are any users? This was approached by seeding the community with participants recruited as part of Scratch workshops, starting with an 11-week workshop for underserved middle-schoolers as part of an initiative called Citizen Schools.

*Authentic participation:*

From the beginning, I set myself the task of becoming an authentic and active member of the community. My daily activities would include browsing the website, playing with people's projects, and leaving comments with suggestions or merely describing the reasons for enjoying various aspects of people's work. For many children on the site, knowing that their work is visible and eliciting responses from others is a big motivation to come back. I have always tried to represent myself as an adult, but I used the language of everyday online conversations. By avoiding both complicated language and trying to be "one of the kids," I could maintain a real profile.

*Empowering active members:*

Clearly, from an early stage some were not only spending more time on the site but also contributing to it in meaningful and positive ways. Some were adults and others were children. An attempt was made to acknowledge their presence and recognize them when possible by featuring their work on the front page or even inviting some to become "moderators" of the discussion forums. This is an aspect of an online community where technological interventions can be effective.

---

<sup>15</sup>In particular John Maloney, Evelyn Eastmond, Brian Silverman, Paula Bontá and Natalie Rusk with the guidance and leadership of Mitchel Resnick, have more than a century of combined experience developing creative tools for children

*Quick iterations:*

As mentioned before, development of the Scratch desktop application occurred in a careful, slow, and almost artisanal model that led to a high-quality product. The desktop application embodied the long-held intuitions of its creators — often a blessing, but on occasions it hindered speedy innovation. The desktop nature of the application made it hard to deploy changes quickly. In addition, getting feedback from users was not as simple as checking server logs. In contrast, the Online Community development took a radically different approach. Inspired by some aspects of Agile software development (Schwaber and Beedle, 2001) the website has evolved with its users through quick iterations. The first version of the website made a few assumptions, so it was released with enough functionality for people to share their work but without complex mechanisms for moderation, to mention just one of the features that we later spent significant efforts in trying to improve. The website had clear goals but I was not too tied up in the details. I wanted to get people to use the system and to learn from them.

At first, we released changes as often as every other day, some of which were drastic changes such as the removal of the most recent projects from the front page. In the last part of the five-year period I worked on the website the changes were less frequent, about every other month.

*Emphasize user contributions:*

The layout was simple, perhaps too straightforward compared with other sites for children, but there is something to be said for simplicity in web design. Sites such as Craigslist or 4chan have proved successful despite having extremely straightforward and archaic designs.

*Software-embodied culture and policies:*

The architecture of the Scratch Online Community evolved with its users, so, as social transgressions occurred, changes to the infrastructure minimized their effect. This involved not only the implementation of more features but also the tweaking, over time, of the parameters of those policies. For example, the maximum number of times one can post the same message over a specific period, the list of approved words, or the whitelisting of URLs all changed as the community grew. Similarly, as I saw people using the website in new and desirable ways, I devoted efforts in trying to support those users. For example, the implementation of an API that would let people build companion websites was the result of one user's efforts to create a repository for clipart and code useful for Scratch projects.

*Generalizable specificity:*

On the surface, the Scratch Online Community is about sharing animations and video games; however, people come together to do more than that. People discuss their family and school lives, the games they like. Scratch is a place to hang out with like-minded people; it is a clubhouse, not just a repository. Online communities need to balance between focusing on something specific (for example, programming) and letting the conversations diverge into the general (for example, family issues). Some of the most successful (in volume and engagement) online communities have exhibited

similar patterns of helpful divergence. For example, reddit.com originally focused on technology topics, but now it has diverged into comics, politics, science, and arts. Similarly, one of the most controversial and influential communities, 4chan, was originally focused on Anime subculture, but has now diverged into a range of topics similar to reddit. Likewise, Facebook originally focused on college life and Flickr on sharing screenshots of videogames. The “seeding culture” influences the discourse and the people that it attracts, but once it happens, community designers must give some room for divergence. How much divergence depends on how much the designers want the community to grow. As it diverges, it also alienates longtime participants.

### 3.3.1 Learning through Online Community

The Scratch website offers an alternate model for how children might use the web as a platform for learning, enabling them to create and share personally meaningful projects, not merely access information. Children create and share Scratch projects to express themselves creatively, much as they would paint a picture or build a castle with LEGO bricks. In the process, they not only learn important mathematical and computer-science concepts, they also develop important learning skills: creative thinking, effective communication, critical analysis, systematic experimentation, iterative design, and continual learning. The ability to produce (not merely interact with) interactive content is believed a key ingredient to achieving digital literacy and becoming a full participant in the interactive online world.

The Scratch Online Community makes programming more engaging by turning it into a social activity. Hobbit, a 14-year old member of the community explains:

*When I think about it, recognition for my work is what really drew me into Scratch. Other things played a part, but the feeling that my work would be seen is what really motivated me.*

The website provides extensive entry points for community interactions. Children comment on projects, upload their own projects, and can become involved in existing projects. The site is also a repository of user-generated content that is a source of inspiration and appropriable objects for new ideas. Users can connect, forming a social network of creators and collaborators through “friendship” galleries (groups of projects based on a topic) and forums where users can post their questions or interests to be discussed with others. Inspired by Jenkins’ description of the states of participation in fan-fiction communities (Jenkins, 2006), it can be useful to frame the participation of members of user-generated content communities in four roles or states of participation: passive consumption, active consumption, passive production, and active production. To build a successful community it is essential for those sites to support and welcome users despite the state of participation they fall in. For example, Lave and Wenger (1991) argue that “peripheral participation” is a legitimate form of engagement. These roles/states are the core of most user-generated content sites and the Scratch Community addresses them in a relevant way for the specific audience and type of content.

*Passive consumer:*

Online communities often call these people lurkers. In this state, people assess the community to understand their values and ideas. With Scratch, this involves the act of browsing the various categories and interacting with Scratch projects others have created. Although this is the most passive state, the passive consumer alters the system merely by viewing because the number of views is for public presentation.

*Active consumer:*

An active consumer participates in the community by providing metadata. Active Scratch consumers contribute their ideas by commenting, tagging, and rating projects.

*Passive producer:*

In this state, users create projects, sometimes inspired by other projects they have seen in the community, but do not necessarily feel compelled or ready to share them to the community.

*Active producer:*

An active producer not only consumes but also contributes to the repository of projects. This person gives feedback to others' projects, becomes inspired by others, and provides inspiration to others. An analysis of website usage showed that the number of projects a user creates correlates to the level of activity by that user on projects created by others (Monroy-Hernández and Resnick, 2008). That is, there is a correlation between the number of projects a user creates and the number of comments posted on others' projects, views on others' projects, projects marked as favorites, projects marked as "I love it!", and projects downloaded. Smaller correlations were found with tags. Others often recognize the level of involvement of these active producers. Members in this state feel invested in the community and it is one of the most important assets of the Scratch Online Community.

### **3.3.2 Sharing and collaboration**

Professional programmers are familiar with the process of reuse or remixing, and base much of their work on programs and algorithms created by others. With Scratch, there was a wish to introduce children and teens to this approach, because learning in a community is more convenient, as well as more rewarding and engaging.

One of primary goals of the Scratch online community is to foster the idea of learning from one another by building on others' ideas or projects. This is one of the reasons that it is always possible for a member of the community to download the source code of any project. Additionally, users of the community often create their projects when inspired by other projects they see. In this type of creative appropriation, no code or media are reused; instead, it is the *idea* that is appropriated to create a project (see Figure 3-33). This type of appropriation often leads to the emergence of trends in the community. One of these trends was started by an interactive "dress up" project created by an 11-year-old girl from South Africa. The project was a digital version of a traditional paper doll where the viewer could choose the skin color, hair, and clothing of the doll. Projects tagged as

“dress up” were so popular that they often are in the Top Viewed section of the front page with hundreds if not thousands of views. To date, there more than 150 projects tagged as “dress up” varying from a project about dressing up a hero to dressing up a famous TV star and original characters; “dress up” projects are as diverse as their creators are.

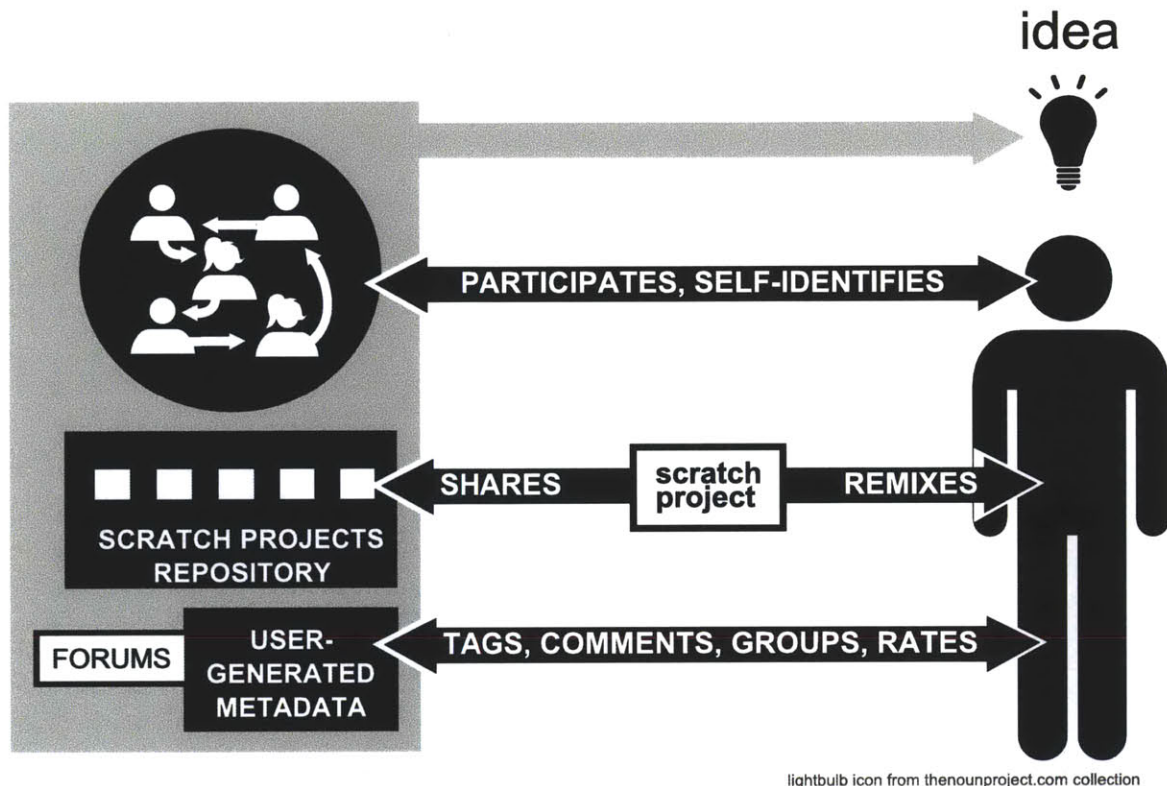


Figure 3-33: Scratch users contribute to and learn from the online community.

The Scratch website is a repository of code and ideas that can be creatively appropriated to spawn new ideas and new projects. The Scratch website and development environment make it easy for this to happen. Exactly 670,932, or 27.64%, of all projects shared were remixes of other projects. Of those, the changes made varied from simple changes to images and sounds, to changes to the actual code. With each sharing of a project on the Scratch website, the desktop application adds information about when and who shared the project. This information automatically connects projects based on other projects. When a project is a remix of another it displays a link to the original project, giving credit to the creator whose work has been remixed. Several members of the community have posted messages on the online forums expressing their concern about others “copying” their work. This controversy has provided an opportunity to discuss important ideas and differences between plagiarism and sharing. A more detailed analysis occurs in the chapter focused on attitudes toward remixing (chapter 6).

## Chapter 4.

# Process of remixing

*When I first started Scratch, I didn't know much about it or how it worked. So I gave up on it. A few years later I bought my own computer and decided to give Scratch another chance. Being a bit older I had more of an understanding of how it worked, but I still didn't really know how to use it very well. I knew from the start I was going to make games...*

*I started off with a great game idea that I'd saved over the years... I was finished, but not satisfied. The movement was choppy and in my opinion unacceptable. So I searched the site for platforming engines. I found a nice one... At that point I had no clue what remixing was so I planned to just copy the scripts block for block in another project (with credit given of course). That's when I looked at the top left corner of the scratch program and noticed the share button was still there. I gave a quick look at the scripts and began making my game...*

*I have to say if it weren't for remixing, I would have never understood velocity or scrolling. it should be used for things other than "add yourself" and coloring contests (not that I'm against those in any way) it's a tool that makes the Scratch community stand out as a friendlier and more learning based environment.*

*—14-year-old boy from the USA.*

Remixing has a significant presence on Scratch and other similar communities. In the five years of my analysis, more than a quarter (27.64% or 670,932) of all projects in Scratch Online Community were remixes of previous projects. Similar percentages of remixes have been reported on similar systems such as Kodu Game Lab and Studio Sketchpad (Bader-Natal et al., 2012).

In this chapter<sup>1</sup>, I investigate *how people engage in remixing on the Scratch Online Community*. The focus of this chapter is to show how remixing represents a wide range of practices.

---

<sup>1</sup>Based on published article: (Monroy-Hernández and Resnick, 2008).



I have defined remixing before as creating something new based on something old. While operationalizable, this definition encompasses a wide range of practices that need to be teased out. For example, remixing to fix software bugs on someone’s project is qualitatively different from the customization of a project to make it more personal, or from the kind of remixing intending to spread a “meme,” or the kind that helps members of a group collaborate.

I start by grounding this section with a set of remixing cases that exemplify some of the types of remixing that I have observed over the past five years in the Scratch Online Community. I then propose remixing taxonomy based on two dimensions: originality and generativity (as a proxy for “collaborativeness.”) I end with a discussion on the implications of different types of remixing for learning and design, and the broad motivations people might have to remix.

## 4.1 Case Studies

### 4.1.1 Mesh Inc’s collaboration

Since the Scratch website went online in May 2007, the children themselves, with the primary goal of creating projects collaboratively, have spontaneously formed many “companies.” These companies were formed using the “galleries” feature of the website which were originally designed to group projects around themes, not explicitly created for supporting collaborative groups. The website later on added better functionality for the loading of large number of comments in these galleries as a response to the usage part of the “company” phenomenon. One of the first collaborative efforts on the Scratch Online Community started when a 15-year-old girl from the UK, username BeepBop<sup>2</sup>, created a project with a series of sprites for people to remix (see Figure 4-1). The sprites included a boy and a girl walking with winter gear, and some elf-looking characters. “You can take any of these to use in your own project, or you can post a comment saying what you want and I can make it for you,” she explained.

Right after BeepBop uploaded her project, another user from the UK, a 10-year-old girl with the username SoundBubbles, wrote a comment complimenting BeepBop’s animations and asking her to create a project with “a mountain background from a bird’s view” for use in one of her own projects. SoundBubbles also asked BeepBop to submit the project to Mesh Inc., a “miniature company” that she had created to produce “top quality games” in Scratch. SoundBubbles explained “All you do is simply send in a project, I will review it back in the Mesh gallery, and, then, if it’s good enough, I will grant you a member of Mesh INC!” BeepBop accepted by saying

*I will gladly make you some mountains... I’ve actually already made some, but it’s not from a birds-eye view, but i can have a go :) Mesh Inc. sounds goooood, but I’m only really good at the drawing characters and background stuff, I might need some help with the programing.”*

---

<sup>2</sup>All user names are changed to protect the privacy of the participants.

The image is a screenshot of a Scratch project page. At the top, the Scratch logo is on the left, and a navigation menu with 'home', 'projects', 'galleries', 'support', 'forums', and 'about' is on the right. Below the logo is the tagline 'imagine • program • share'. A 'Login or Signup for an account' button is also visible. The main content area is titled 'sprites' and shows a preview window with a character and a green arrow. To the right, there is a 'Download this project!' section with a small diagram of a sprite's movement, and a 'Project Notes' section with text explaining the project. At the bottom right, there is a 'Tags' section with an 'Add Tags' input field and an 'Add' button. Project statistics are shown at the bottom left of the preview area.

**SCRATCH**  
imagine • program • share

home projects galleries support forums about Language

Login or Signup for an account

search

sprites

who's next 1

BeepBop shared it 4 years, 11 months ago Some rights reserved

1203 views, 1 tagger, 12 people love it, 122 downloads, in 8 galleries

**Download this project!**  
when clicked  
move 10 steps  
play sound 100

Download the 6 sprites and 9 scripts of "sprites" and open it in [Scratch](#)

**Project Notes**

Here are some simple walking sprites, each one only has two costumes.... They are peeerfect for platform games.... if u wanna sprite of your very own leave a comment in my gallery [\(link to gallery\)](#) say what u want and i will make it for u... be patient, coz it might take a few weeks.... i am very busy ^^

**Tags**

Add Tags

Add

Figure 4-1: BeepBop's sprites project.

SoundBubbles and BeepBop continued their exchanges and created an initial collaborative project by remixing one another's contributions. The company's headquarters was a gallery on the website, using the comments section to coordinate.

A few days later, a 14-year-old American boy who went by Hobbit, discovered the Mesh Inc. gallery and offered his services: "I'm a fairly good programmer, and I could help with debugging and stuff." SoundBubbles asked Hobbit if he could solve a problem with a particular Mesh Inc. project: "I can't make characters jump so you're up". A day later Hobbit fixed the game and posted, "this is the new updated version, so now he can jump on the snow." SoundBubbles replied, "gr8 job, Hobbit! I'll take this and carry on from here." Meanwhile, Hobbit decided to put his blogging skills to use and created a blog for Mesh Inc. where each member is listed with their corresponding role. SoundBubbles was selected as the "chairlady." Later, an 11-year-old boy from Ireland calling himself Marty was added to the Mesh staff as the expert in "scrolling backgrounds."

As others witnessed these interactions happening, Mesh Inc often received recognition in the community, and many people started to "audition" for Mesh Inc. MusicalMoon, a 12-year-old girl from Russia, started to lead the "character design" and "sound operations" with GreenDinousar, a 10-year-old boy from the US, who holds the title of "story writer." Soon after, the group could no longer handle the large number of requests and asked applicants to "audition" by submitting a project to their audition gallery. "If you want to join Mesh Inc., please put a sample project in this gallery and we will see if you are right for us," explained their gallery, which received more than 30 applications.

Despite all the interest and the frequent remixes of each other prototypes, Mesh Inc. never finished any of the projects they decided to work on. However, Mesh Inc. inspired the creation of many other "companies."

A year later, during the summer of 2008, MusicalMoon joined with three other children — aged 8, 13, and 15, respectively— to "found" a new company called "Green Bear Group." Three months later the company had a membership ranging from 12 to 18 children (Aragon et al., 2009).

Like the other collaborations described above, the participants in Green Bear Group have, for the most part, never met, live in different time zones, and do not even know one another's real names. The 8-year-old "owns" the gallery where the company is hosted, and the founders collectively make decisions on company membership. The members then vote on which games to develop. Each member has a specific skill, such as art, music, programming, or storytelling, which he or she contributes to the game through a process of iterative remixing process. The six finished GBG games from the company's first three months required an average of 17 remixes each.

#### **4.1.2 Jumping Monkey's ripple effect**

A 9-year-old American girl using the name Jessy15 joined the Scratch community soon after it was unveiled in May 2007. A month and 18 projects later, she created a video game titled "Jumping Monkey." The game featured a monkey that the player can move across

multiple floors to capture bananas. “Up arrow to jump, down arrow to move down, left and right arrows do move, too! DON’T FALL IN THE LAVA! Oh, and eat the bananas because monkey is hungry!” read the project description.

A 34-year-old Scratcher from the UK, Chaoz, made two remixes of Jessy15’s project. The first made some “simple mods” by adding “pink slippers” to the monkey’s feet so that it would be easier to detect when it touched the different floors. A month later, Chaoz went on to release a fully-fledged scrolling game based on Jessy15’s project. The game was well received by the community, getting more than two thousands visits. Chaoz credited Jessy15’s project as a necessary catalyst: “I’d never have started this if it wasn’t for her jumping monkey.”

Chaoz’ “simple mods” created a ripple effect of their own (see Figure 4-2). About a week after Chaoz’ first remix, MagicX, an 11-year-old American boy, found it and remixed it. He added couple of new obstacles. Then MagicX reused his own remix to create a new more sophisticated version of the game. In it, he gave special credit to the “pink slippers,” although misattributing them to Jessy15: “How i made this: I adapted this shoe technique from Jessy15’s Jumping Monkey.”

Two weeks later, GummyBear, a 20-year-old woman, discovered MagicX’s project and asked him for permission to remix: “Hey MagicX, I love this game. I was wondering if you wouldn’t mind me making some changes.” MagicX accepted and even invited GummyBear to collaborate: “Hey GummyBear, I need someone to help make games for my production company, Mega Software.”

GummyBear accepted the invitation, then MagicX started a project titled “Walk the Line,” a game resembling the initial Jumping Monkey but with a cat as the main character and more sophisticated gameplay. GummyBear completed it by adding a Johnny Cash’s “Walk the Line” song, and very professional-looking graphics. “This is a remake of Super Software Productions Walk On the Lines created by MagicalX,” GummyBear wrote in the description of the project. MagicX posted a comment on the remix saying: “its amazing what you’ve done with my game.” The game was posted to the Mega Software gallery and went on to receive 15,844 views, 458 loveits, 2,088 downloads, and 23 remixes. The remix ripple continued.

### 4.1.3 Galaxyman’s “media franchise”

In mid-2010, a 10-year-old child, who had joined the Scratch Online Community five months earlier, created an animated story titled “Choco Bar.” It featured his namesake, “Galaxyman,” and he described it with the headline: “You know it’s not going to end well when it involves chocolate.” By then Galaxyman had shared more than 180 projects, many of them part of a series of animated stories, such as “Mr. Pineapple Head episode 1” and others.

The “Choco Bar” received thousands of visits, more than hundred comments, and inspired more than thirty remixes, one of which reused some of Choco Bar’s components to create

6/22/2007

7/11/2007

7/18/2007

7/22/2007

8/3/2007

# SCRATCH

a journey though remixes

## TYPES OF REMIXES

- INCREMENTAL
- INSPIRATIONAL
- RESTRUCTURE
- COMPONENT

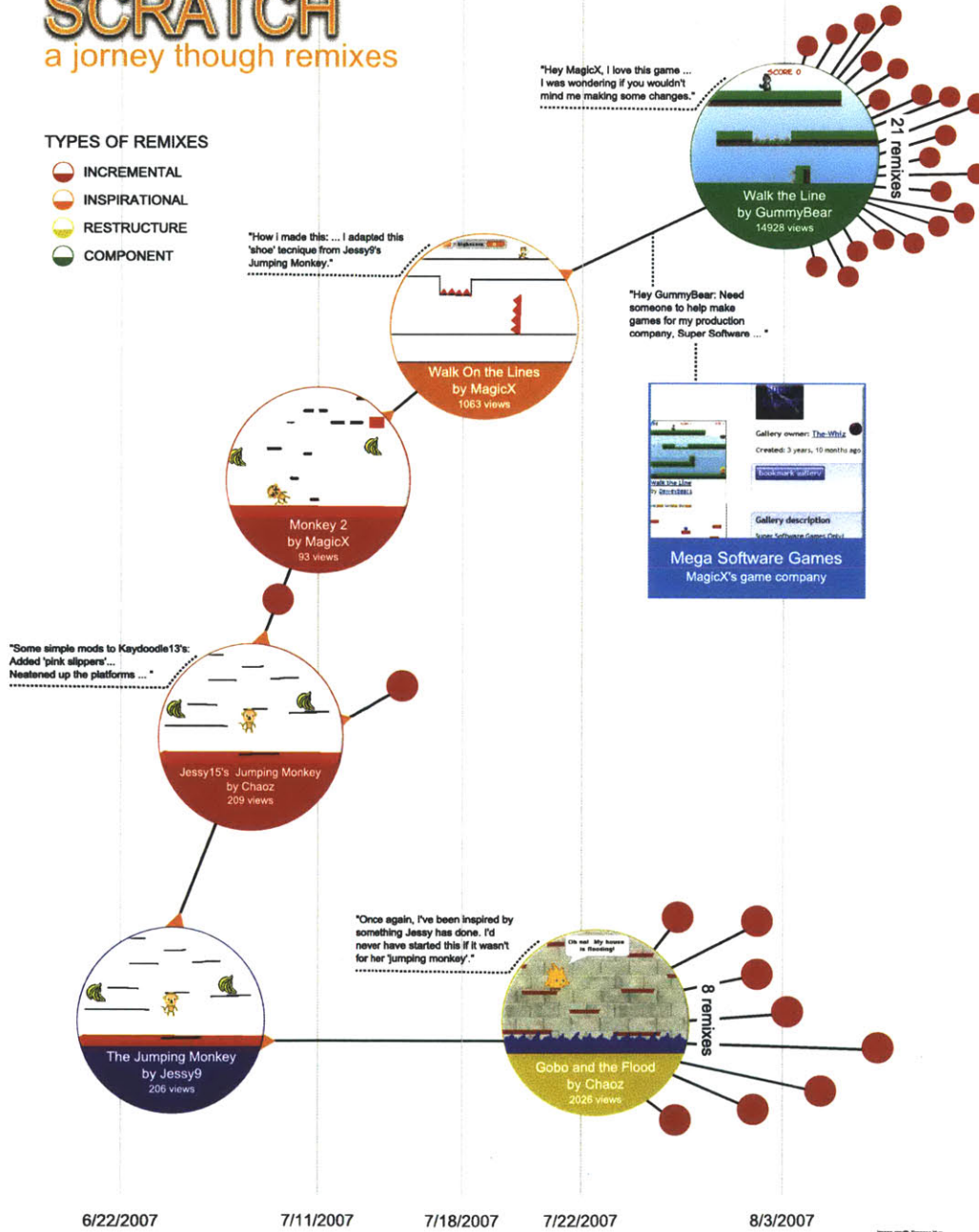


Figure 4-2: Remixing events started by "Jumping Monkey"

an animation featuring Galaxyman grabbing a chocolate bar with the word “success” in the background.

Inspired by how well-received “Choco Bar” was, the young creator produced a sequel, “Choco Bar 2,” which was equally well-received. In fact, another community member, BadumJack, created a whole series of Galaxyman-inspired remixes such as “If I were a Galaxyman character” and “Galaxyman babys.”

The third episode of “Choco Bar” was a success too as it received more than two thousand visits and hundreds of comments. Galaxyman described it thus: “Teh sequel to the candy bar 1 and 2,” and asked people to give positive feedback: “Please luv-it cause this took alot of time.” The records indicate that he worked on the project during three sessions, one of which might have lasted six hours. The project was remixed dozens of times. Many of those involved incremental changes, with of people creating projects such as “Choco vs Moon” and the “The Kitty Candy,” involved tweaks of the source project (see Figure 4-3 for a diagram of Galaxyman’s remixes).

Four months later, by mid December 2010, the fourth episode came out featuring Galaxyman running away from a big snowball. This was Galaxyman’s most generative project ever; it was remixed twice as much as the previous episode. Some of the remixes, such as “Add yourself to the x-mas snowball run away,” reused many of the sprites from “Choco Bar 4” but was created to explicitly invite others to remix by adding their character: “add your anthro with a message and then teleport them to the snow ball channel where you can add them running for their warmth and lives.”

The fifth and last episode of “Choco Bar” came out in mid-2011. Galaxyman wrote in the description “OMG that was A LOT of work.” Indeed, Galaxyman worked on the project for more than two weeks; the file saving records indicate that he labored each day for several hours. He also asked people to “please give this project as many tags as possible,” to increase its visibility. The project was tagged by 557 people, received 6,456 visits, 560 people loved-it, and it was remixed 289 times. Many of the remixes were parodies, or “spoofs” as many users call them, with different endings.

The series also triggered the creation of “Galaxyman Fan Club” where people submitted their remixes and imagined different endings. Galaxyman became the impetus for more than 1,000 remixes, inspiring several of these fan clubs and galleries, and creating a new art style with his stick figures. Many remixes are merely tweaks, customizations by other users. However, others take it much further. Many other people animated the Galaxyman characters into creations of their own design, creating projects on what it would be like to meet him, or be one of his characters.

Galaxyman is one of several characters that are part of episodic stories that create a following of dozens or even hundreds of children on the Scratch Online Community. Much like professional movie or TV producers, community members such as Galaxyman have created popular media franchises, built new cultural materials, and inspired a host of creative remixes.



## 4.2 Taxonomy

The diversity of remixing cases presented above suggests the need of a taxonomy of remixing to help tease apart the different types of remixes we might find. In this section<sup>3</sup> I propose a taxonomy for categorizing remixing based on the cases discussed above and five years of observations of the Scratch Online Community. This taxonomy is based on two dimensions: “originality” and “generativity”. Using these two dimensions, one can identify a set of remixing categories useful for the analysis of the functional roles of remixing in Scratch and beyond (see Figure 4-4).

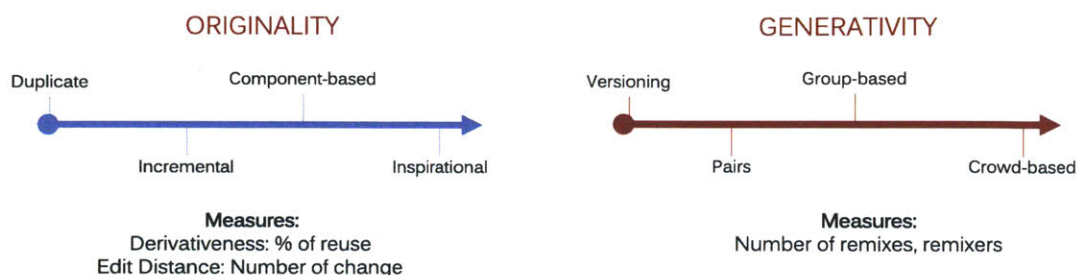


Figure 4-4: Remixing taxonomy based on originality and generativity.

### 4.2.1 Originality

The word originality is often used in a way that implies a value judgment: the more “original” a remix is, the better. However, here I use originality merely as a way to indicate how different a remix is from its source without drawing conclusions on the value of the remix.

Originality matters because it gets at the core of some tensions around remixing. For example, widespread remixing has spurred a debate around what Hemphill and Suk (2009) describe as the “distinction between close copying and remixing.” which is the theme of chapter 6.

Thinking of remixing along the originality axis helps us categorize derivative works by how transformative they are. It helps us distinguish remixes that are merely inspired by someone’s idea from those that did some tweaking of existing works and those that are replicas of previous creations.

#### Inspirational

It is often the case that people are simply inspired by the ideas of someone else’s work. In Scratch, this happens when people browse the website and find projects that motivate them to create similar ones. For example, in the story of the Jumping Monkey, one of the

<sup>3</sup>Based on joint work with Benjamin Mako Hill to be published in a special issue of American Behavioral Scientist.



remixes acknowledges using the “shoe technique” from a previous project without actually reusing the exact code from that project.

This came up during an interview with a long-term member of the Scratch community. I asked him what his most recent remix was. He replied directing me to one of his projects and another project that he credited as the inspiration. In our conversation, after some confusion, he sent me an image to explain more clearly what he meant (see Figure 4-5). It was then that I realized that he had not actually reused parts of that project but rather a particular technique.

**Andrés:** *I’m looking at your code and Whimsical’s, it looks very different.*

**Carpiz:** *let me open it up for a sec*

**Carpiz:** *yeah i didn’t directly copy code if that’s what you’re wonderng, but its based on his code*

**Andrés:** *i see, do you remember how you did this? did you have his project open on one window and yours on another?*

**Carpiz:** *yeah*

**Andrés:** *cool, so what sprite in your project has code inspired by Whimsical?*

**Andrés:** *p1?*

**LC:** *the P1 and P2 i think... the AI...*

*\* LC sent file codecomparison.png*



Figure 4-5: Inspirational remix.

The nature of inspirational remixing makes it difficult to capture through computational means. As such, it was not included in the statistics I had reported claiming that 27% of all projects in Scratch are remixes. However, a simple query in the database for projects that have not been identified as remixes but that contain the text “inspired by” yielded 2,736 projects. For example, one explained “This game was inspired by one of the mini-games in Super Mario 64 for the Nintendo,” while another stated:

*Hi everyone. This is my first project on Scratch. It’s really funny. I got the idea from the “Don’t Press the Button” games.*

## Incremental

Incremental remixes consist of making tweaks or adding something extra to a project. This type of remixing often involves downloading someone's project to customize it or to fix a bug. For example, it may involve replacing the costumes of a sprite in a game for a different one. This form of remixing often occurs when people make small modifications to the sample projects that come preinstalled with Scratch. Also anecdotal evidence suggests that this form of remixing tends to be a useful way for newcomers to get started with Scratch, modifying an existing project instead of creating something completely new.

For example, in the Galaxyman story, the project titled 'Add yourself to the x-mas snowball run away' invites people to remix by adding their own character to the chain of remixes. I did a search for projects that are identified as remixes and that have the words "I just added" or "i made it better" in their description. For example, I found projects like "boulder rash" where the remixer explained (see Figure 4-6):

*Notice that this game is originally from rpm55. I just added the sounds, nothing more (or less)!*



Figure 4-6: Example of incremental remix.

## Component-based

Remixing also occurs when people use pieces of others' projects to produce something new, rather than building on top of existing work. In these cases, often one cannot quickly tell in what way the remix and the creative work are related. This type of remixing typically involves some sample sprites that come preinstalled with Scratch, or templates, images or sounds that members of the community created for others to reuse.

Other branches of this investigation have looked into which images and programming blocks are more commonly used. For this project, it was important to learn which are the most common programming constructs or scripts created by the young Scratch programmers (see Figure 3-28).

This type of remixing occurs when people use pieces of others' projects to produce something new, rather than building on top of existing work, as is the case with incremental remixing. In these cases, often one cannot quickly tell in what way the remix and the original are related. In the Galaxyman case described above, the remix title “Add yourself to the x-mas snowball run away” represent this type of remixing as it reuses two sprites from one Galaxyman’s projects (see Figure 4-7).



Figure 4-7: Example of component-based remixing.

## 4.2.2 Generativity

Another way of categorizing remixes is based on their intended or accomplished generativity, in other words, the number of derivative works that a particular source material engenders, or tries to.

The classic example is of a highly generative piece of content called the “Amen Break,” which some argue is “the most sampled track in the history of music” (Economist, 2011). The Amen Break is a “drum sample taken from the b-side of a record released by the Winstons in 1969” (Bown et al., 2009). This six-second drum beat has now been remixed in thousands of songs and it “effectively came to define a musical style,” by “appearing on hip-hop tracks in the mid-1980s, and later, at a massively sped-up tempo, in drum and bass and jungle in the 1990s,” as well as countless number of TV commercials (Bown et al., 2009; mobius32, 2006).

Generativity is also a proxy to understand the context in which remixing happens. For example, a piece of de novo content might be remixed only within a pre-defined group of people, as it is the case of the “companies” I described above. In other cases remixing occurs as some form of agreed upon peer-to-peer interaction, as is the case in most of the remixes in the “Jumping Monkey” story. In other cases remixing occurs at a large scale and

without much coordination, as is the case in the “Add yourself to the x-mas snowball run away” project, which invites anyone to remix. These distinct types of remixes are described below.

### **Crowd-based**

Crowd remixes are often initiated when a creator explicitly invites others to remix. For example, the community member DarkSun created “5 Random Facts About Me! Meme,” a project where she asked other users to add their own facts. “Do it! Remix please! Press space. I’ll do one too,” she explained. Another member, Mozzarellagirl decided to join DarkSun: “Yeah...I can’t believe I actually did this project. Yep, jumped on the ‘5 Random Facts’ bandwagon.” Later she explained,

*I first saw DarkSun’s original project via the front page and in my RSS feeds of people’s projects [...] I ended up wanting to join the remix chain myself [...]. I thought that it would be so much more fun to look at if I tossed in some of my art and animations in there. It took nearly a week to get the graphics done, and much of DarkSun’s original programming had to be changed to accommodate for the animations (my remix still keep true to DarkSun’s original idea of pressing the space button to see the next fact). [This] eventually became one of my most commented projects, and I suspect it is also one of the more popular projects I’ve done.*

Other examples of this type of remixing are those that invite people to remix for a social cause, such as the project titled “Remix if you care about animal rights.” There are also those that invite people to participate in contests, such as the “Coloring contest” genre popular among more artistically-inclined community members consisting of downloading a still image (often with music playing in the background), coloring it and submitting it to a contest for the best-colored image. Similarly, there are those titled “add yourself to X,” where X can be any kind of collective activity such as a party or a boxing match. For example, the seventh most remixed project with 1,978 remixes is a project created by anniedoughnut, an 11-year-old girl, that shows an animation of a girl running away from a big boulder (see Figure 4-8). The project title and its description invite others to “add somebody running from the boulder!” The project was remixed early on by 11-year-old boy who remixed the girl’s project by adding an animated version of his avatar, along with an invitation for others to remix: “Come on, let’s everyone remix this and make it the biggest boulder run ever! This was started by anniedoughnut.” The project created a popular remix chain.

The home page of the Scratch website has a section called “What the Community is Remixing” that features the three top remixed projects in the past two weeks. Although these could represent any type of remixing, they are typically projects in this crowd remixing category, often referred by the community as “remix chains.”

Although typically the intention of the creator of this type of project is to create a chain, often the structure of the remix network looks more like a star, as participants may not



Figure 4-8: Sequence of iterations of a crowd-based remix.

follow the rules or check where the last element of the chain is.

Like the category above, these types of remixes are often incremental, but the relationship between the creator and intended audience is significantly different. Crowd remixing explicitly invites people to remix en masse following a specific template. The purpose of the creation is not the thing itself but the collection of many remixes created by many individuals.

### Group-based

Group-based remixing tends to involve several back and forth interactions through remixing, something that one of the Scratch members I interviewed referred to as “ping pong remixing.” This is the kind of remixing that happens in groups like Mesh Inc or the Green Bear Group. It does not involve as many people as crowd remixing and the type of remixing is also qualitatively different.

These groups traditionally are formed using “galleries” as their common space. I searched for all those galleries that had the terms used by these groups, terms like “company” and “productions,” and I found 1,705 of these groups. As in many peer-production websites, the majority of those groups did not engage in collaborative work. Only 27.97% (477) groups had one or more remixes among its members after the creation of the group.

### Versioning

About 10.25% (248,833) of all projects or 37.08% of all remixes are created by the same person who created the source project. This is typically used as some form of version control. For example, sometimes children create a Scratch project at school, upload it to the website, download it at home, continue working on it, then reupload it to the website under a name such as “My video game v2.” When this happens, the website identifies them as remixes and links them back to the previous version. Some people, mainly popular Scratch creators who care about the projects displayed on their profile page, have two accounts: one for testing and one for sharing final version of their projects. For example, there is an adult member of the community who has two user names: “Paddle2See” and “Paddle2SeeTest.” As the name implies, one is often used for sharing drafts or work in progress.

This type of remixing represents the lowest type of generativity in terms of people, but it is represented as a remix in Scratch and other online communities. For example, in the source code community GitHub forks are displayed even if they are created by the same user.

### 4.2.3 Measures

Although originality and generativity can be highly subjective, here I present a couple of ways one could measure these in a quantitative and systematic way.

#### Derivativeness

Similar to the metrics for software reuse described in the Introduction, this technique for measuring derivativeness in computational media involves calculating the *amount of content present in a remix that is derived from the antecedent work*. In the case of software, comparing lines of code is enough, but for programmable media, it is important to also consider differences in other types of content.

Although this approach allows us to consider derivativeness in code and media in aggregate, it also gives the ability to characterize the nature of remixing practice by allowing us to compare the derivativeness of projects and their components along each of the following dimensions.

For this analysis, we take apart each project’s internal elements (sprites, i.e., characters in a game or animation) and examine each of its internal *media* objects, more specifically: images or costumes, sounds, and text (the kind of text that appears in “speech balloons”). Scratch *code* elements can be described in terms of the *block counts*, such as the number of “move  $x$  steps” blocks, and *arguments*, such as the value of  $x$ . Changes in these two attributes will be the basis for calculating derivativeness for code.

To calculate derivativeness of works in Scratch, we first build a list of remix-antecedent pairs. Scratch projects, like other remixing platforms, can have multiple versions of projects, it is important to identify the version of the antecedent that was remixed by a particular project. Second, we compare each remix against its antecedent checking for added, deleted, and preserved images, sounds, texts, code blocks, and arguments. We check the size (in bytes) of the images and sounds that are preserved (i.e., present in both the remix and its antecedent) to determine if they were altered. If they changed, then we treat these media as “edited” and convert them into preserved, added or removed items based on the size difference. A common practice among Scratch users is to edit images. Since images are stored as bitmaps, these byte differences are translated to added, deleted and preserved items.

We calculate the *aggregate derivativeness* of a remix by adding all the preserved items across media and code and dividing it by the total number of items, also across media and code, in the remix. To generate additional insight into remixing practices, we separately calculate number of preserved media items by adding the number of preserved images, sounds and

text and divide it by the total number of media items in the remix in what we call *media derivativeness*. We also calculate *code derivativeness* by dividing the number of preserved blocks and arguments by the total number in the remix.

Applying this analytical technique to the almost 500,000 pairs of projects, we found that 19% are 100% derivative, that is, exact replicas of their antecedent. The distribution of aggregative derivativeness is shown in the top left corner of Figure 4-9. The median remix is 86% derivative; and half of Scratch remixes changed more than 14% of the media and code in their antecedent.

Although some users engage in replicating in the Scratch community, a large majority of remixes are at least partially transformative. The disaggregated distributions for media and code derivativeness are reported in the top right-most panels in Figure 4-9.

The distribution of media derivativeness shows a thicker “tail” than code derivativeness suggesting that, in Scratch, remixers are more likely to change larger amounts of media than code.

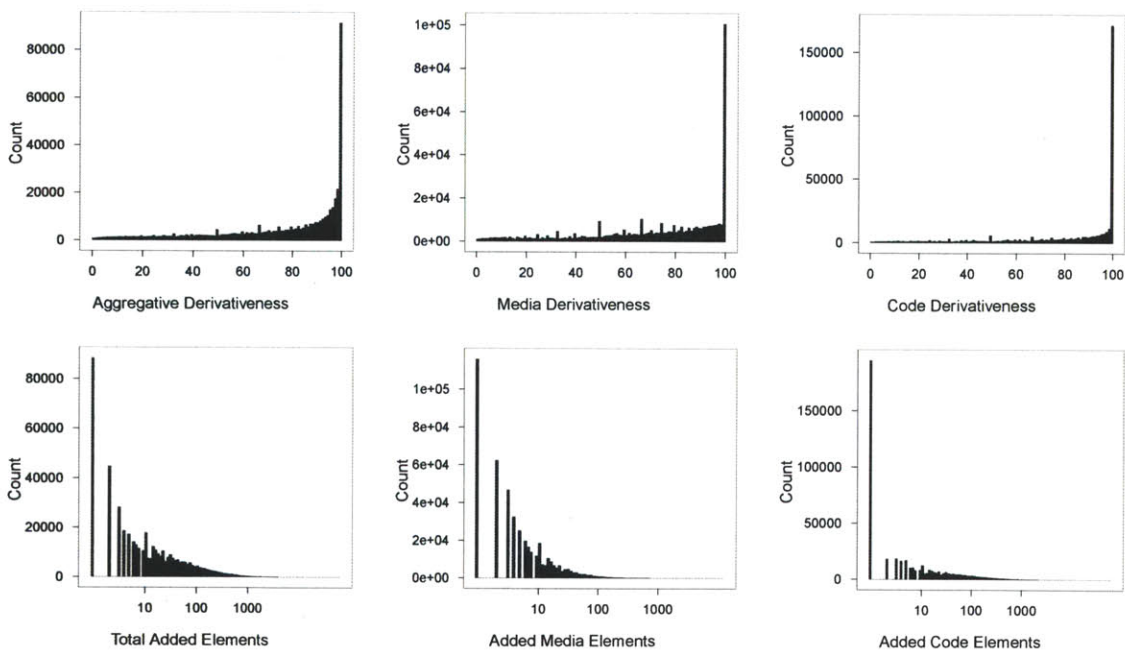


Figure 4-9: Distribution of derivativeness.

One important limitation of the metric by (Frakes and Terry, 1996) — not discussed in their work but made particularly clear in these results — is the metric’s sensitivity to the size of a project. For example, a project with one block and one image will be called 50% derivative when a single element is changed. Similarly, a 1,000 block remix with 100 blocks changed will be described as identically derivative to a 10 block remix that changes only a single block. The small spikes seen at 50% and 33% in the derivativeness histograms are caused by projects with a small number of media or code items where adding 1 block makes

a substantial difference.

Speaking to this limitation, the second row of Figure 4-9 reports the distribution of added elements only — again reported as total elements, media, and code. Because this distribution has an extremely long tail, these results are reported on a log scale. The median remix added seven elements to its antecedent. Although a large portion of remixes added only a small number of elements, the large majority of projects made at least some changes. These results suggest that the distribution of derivativeness is not entirely a function of small number of changes within small projects.

## Edit Distance

Another way to operationalize the measure of originality is using a calculation of the degree to which a project diverges from its antecedent. To calculate this divergence, we begin with the list of remix-antecedent pairs. Next, we identify and compare each component of the remix to the corresponding component in the antecedent. Our measure of originality is the Levenshtein “edit distance” between each of the scripts of each of the sprites (Levenshtein, 1966). Levenshtein distance is a widely used metric popular in software engineering to measure the divergence of code. The traditional Levenshtein analysis is a character-by-character comparison.

In the case of Scratch, we can use blocks as tokens and our measure of distance is the sum of distances across all scripts and represents the minimum number of changes to blocks that would be needed to convert an antecedent project into its remix (ignoring arguments). Applying this method to all remixes produced in 2010 we find that the mean distance for the code of a project’s remixes is 85.57 blocks (sd = 397.66, min = 0, max = 21,970).

This measure of divergence will be used in the following chapter as a way to understand what makes some content more generative than others.

It is important to note, that this edit distance metric is also applicable to images and sounds, albeit at a much coarser level. For example, I developed an edit distance metric for media based on the presence or absence of images and sounds using their file name and byte size. If the name and byte size of the media file in the remix is the same as in the original project, one can assume that the media element was not changed. If it is different, one can assume that it was changed and, since the media is stored raw, the byte size helps identify how different the two images might be. Of course, a much better approach would be to use image processing techniques, but even with this coarse metric I found that the media and code distances were somewhat useful. That said, in the subsequent uses of edit distance I take a more conservative approach and *focus only on code*. Future research should focus on improving these metrics for both images and sounds.

## Generativity

Roughly speaking one can measure generativity based on the raw number of remixes a particular project engenders. Although more than a quarter of all projects (27.65% or



670,932) are remixes, only 13.86% (313,950) are remixed; of those, the mean number of remixes per project is 2.6 and the mean number of people who remix is 1.9.

Some projects are highly generative, for example, the most<sup>4</sup> remixed project created by a community member in the 5 years of data, was an attempt to create a scrolling game (see Figure 4-10). The game, created by a 14-year-old girl and now remixed 6,041 times, had the code necessary to control the character of the game with the arrow keys, but the background did not scroll. Frustrated, she asked for help on the discussion forums: “HELP! I made my 1st scrolling project but i dont understand it very much. I also looked at what others have said but i still dont understand. if someone could tell me how to scroll step by step and very easy that would help me aaa lloooooott.” A 17-year-old from Canada, creator of a tutorial for scrolling games, found the thread, and decided to remix the girl’s game to fix it. His remix explained: “Scrolling demonstration for Goldilocks” . The girl thank him by saying, “oh thank you so much! you rock!” From then the “demonstration” was reused by several people who wanted a template for a scrolling game. For example, a 13-year-old boy reused the project and said: “this helped me create all my scrollers [...] i would have never known how to create a scroller!” The game was later cleaned up and remixed to be included in the sample projects that are part of the Scratch development environment.



Figure 4-10: The most remixed project created by a community member.

The generative nature of Scratch has changed over time. Mainly the emergence of crowd-based remixes has become more salient. Even before the introduction of the “Top Remixed” projects list on the front page (later renamed to “What the community is remixing”), this was an increasingly common phenomenon. Figure 4-11 depicts the maximum number of levels in a remix chain aggregated by month (*y*-axis) over the course of five years (*x*-axis).

<sup>4</sup>the most remixed project was actually a “Pong” game included in the Scratch development environment which engendered 10,142 remixes

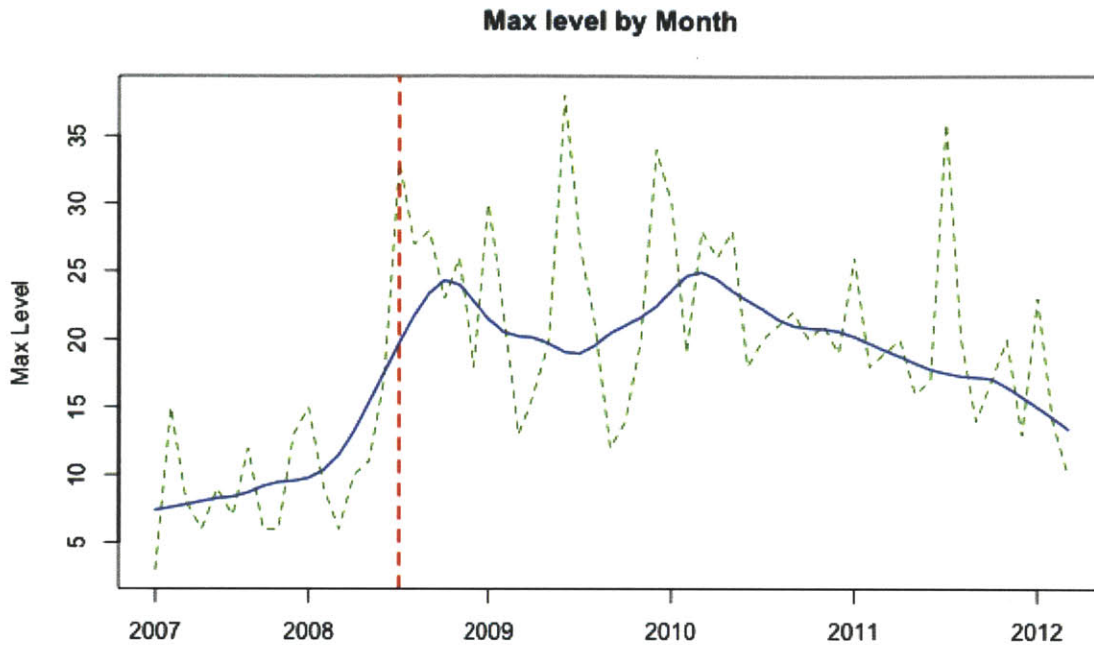


Figure 4-11: Maximum remix levels in remix chains. Blue: moving average smoothing-spline with a 20% window size. Green dashed: connects each value for max(max-level). Red dashed: time when “Top Remixed” was introduced.

Generative projects are often made by generative individuals. In order to find these individuals, I created a weighted directed graph where each node represents a member of the Scratch community and the edges represent connections via remixing. For example, if person A remixes person B, then there is an edge from node A to node B. The most generative individuals would be those with the highest in-degree. A distribution of the in-degree values shows that a few individuals are highly generative while the majority is never remixed and those who are remixed very few times. As expected, the user account with the highest in-degree is the account that “owns” the sample projects that are included in the Scratch development environment. The second highest is the Canadian teenager involved the scrolling game remix described before.

Another way of assessing generativity in Scratch is through the geographical diversity of remixing connections. In the case studies I presented, cases of remixes occur across the US, UK, and Russia; remixing connections over the course of five years have spanned many more countries, as one can see in Figure 4-11.

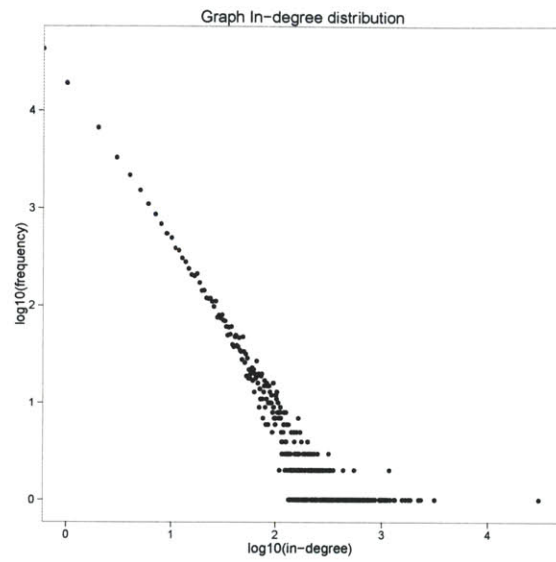


Figure 4-12: In-degree distribution. Nodes: people who have remixed. Edges: Remixing.

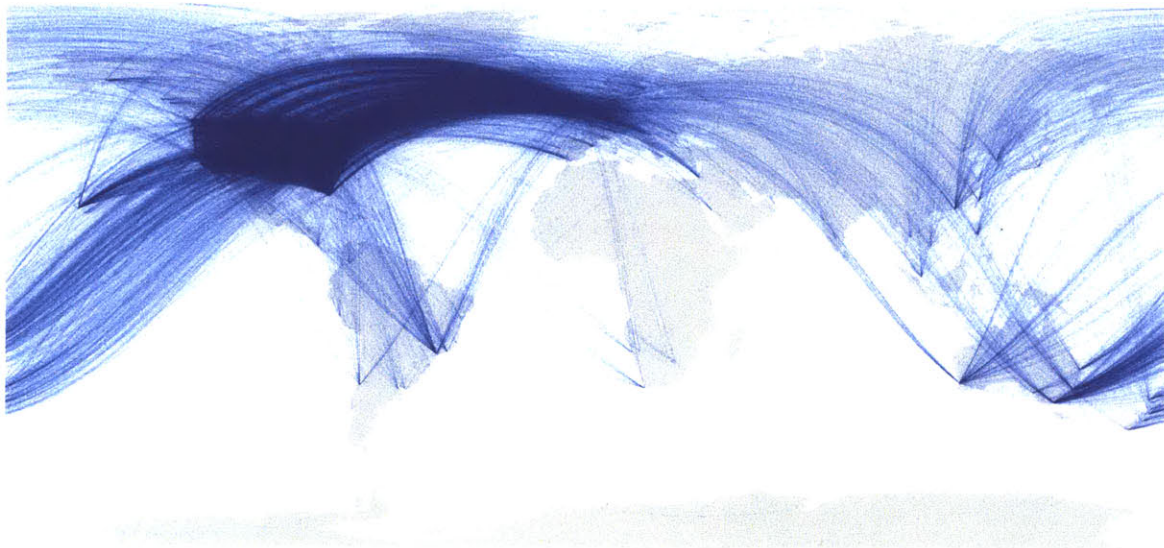


Figure 4-13: Map showing remixing connections.

## Chapter 5

# Conditions for Remixing

*I can help you. I've been on the front page once, and that's my gallery, A Better Place. I got it featured because it was a good idea, I guess. So, make projects and galleries that are unique, special, and one-of-a-kind. Getting top remixed: That's never happened to me before, but from what I've noticed it's stuff that tells about people. For example, memes and yearbooks and that sort of thing get top remixed a lot. Also, projects about good causes like Sign the Petition!! would get on the front page as would voting projects.*

*—10-year-old girl responding to the question “how to get on the front page”*

In this chapter I examine the conditions that are conducive to remixing. More specifically, I analyze *what influences remixing* in the Scratch Online Community. I propose a framework based on two qualities: the *system* attributes on which remixing may occur, and the attributes of the *content* itself.

Using as an example the “Jumping Monkey” project I described in the previous chapter, one may see how this project might have created a remix ripple effect because of the system on which it was shared—the Scratch Online Community—while the drumbeat “Amen Break” might have been highly remixed because of its intrinsic musical attributes. In this chapter I tease out these two types of attributes.

### 5.1 System Attributes

There are at least three core system attributes of the sociotechnical infrastructure of the Scratch Online Community that facilitated, and sometimes hindered, remixing. In this section, I propose a framework for what I think were the key determinants for supporting remixing in Scratch. I base this analysis on the commons-based peer production framework (Benkler, 2006) and my observations over the course of five years. The attributes are summarized in Figure 5-1.

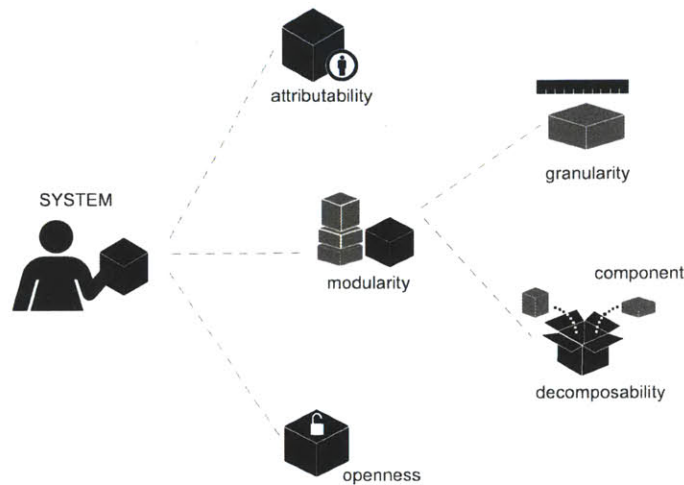


Figure 5-1: Sociotechnical system attributes that facilitate remixing system<sup>1</sup>

<sup>1</sup>icons taken from creativecommons.org, openclipart.org, and thenounproject.com.

### 5.1.1 Modularity

A sociotechnical system is modular when it allows people to create, share, and reuse objects and components. Benkler (2002) defines modularity as the “property of a project” that describes “the extent to which it can be broken into smaller components, or modules, that can be independently and asynchronously produced before they are assembled into a whole.”

This modularity is determined by the *granularity* or size of a project’s components, or, as Benkler (2006) described, “Granularity’ refers to the size of the modules, in terms of the time and effort that an individual must invest in producing them.”

Hence, there are at least three key aspects of modularity in a system: first, the ease of *integration* of components into new creations or remixes; second, the *decomposability* or ease of segmentation of an existing project into smaller components; and third, the *granularity* or size of those components. The following is an examination of modularity in Scratch using these traits.

Scratch projects are composed of multiple smaller components (see Figure 5-2) called “sprites,” such as characters in a game or elements in the user interface of an interactive project. Each sprite can have “scripts” or stacks of programming blocks that control the sprite’s behavior. Each sprite also has one or more costumes or images that represent the various visual states of a sprite, and sound that can be played programmatically.

The Scratch website’s granularity is coarse but users have found workarounds. For instance, the website enables people to share full projects but nothing smaller. Despite this limitation, people have worked around it by sharing components wrapped on full projects. For example, the project that initiated the whole Mesh Inc. company described in the previous chapter was one whose only purpose was to be a conduit for sharing its sprites.

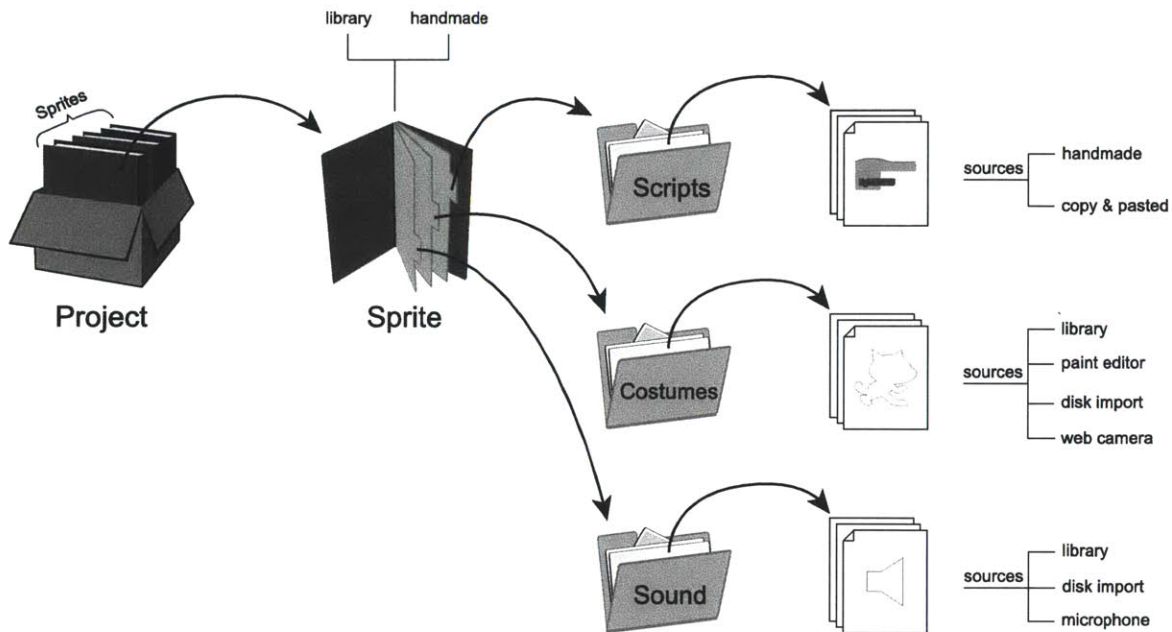


Figure 5-2: Anatomy of a Scratch project.

As a result of this limitation, a group of Scratch community members created a website called “Scratch Resources<sup>2</sup>” to address the granularity limitations of the main Scratch website. Scratch Resources encourages community members to upload more granular components to be reused in projects. The “about page” of the website reads as follows:

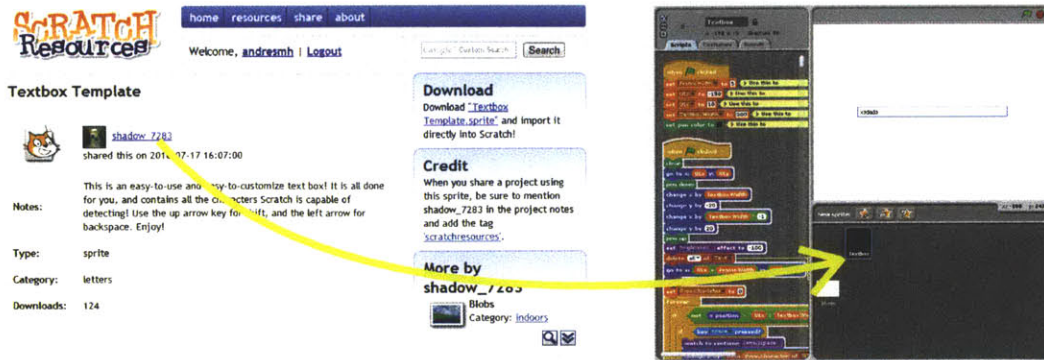
*The goal of Scratch Resources is to extend the Sharing possibilities Scratch already supports. You are now able to share and download sprites, sounds, music, backgrounds, and scripts, next to just Scratch projects. All this sharing could turn every Scratch project into a collaborative project using elements from others all over the world. This was already possible on scratch.mit.edu by putting your music in a Scratch project but we think it is important to provide a different platform, fulfilling the needs of sharing Scratch ‘resources’.*

Early on, the group behind Scratch Resources approached me to get “official” backing. Intrigued by their efforts, I decided to support them by giving them access to an API they could use to connect the authentication of their site to that of the main website. Also, I provided them with hosting space, and a `scratchr.org` subdomain—to give more validity to their efforts. Other than that, the whole system was completely built by them.

The website was reasonably successful. A couple of months past its release, people had uploaded 1,566 resources: 605 sounds, 454 backgrounds, 403 sprites, and 104 scripts. These resources were downloaded by 11,139 different people. One of the most downloaded sprites was “textbox,” an implementation of a typing widget. This sprite was used as the address bar in “Web Wizard v5.0,” a project that tried to emulate a Web browser (see Figure 5-3).

<sup>2</sup>Available at <http://resources.scratchr.org/>

### Sprite on Scratch Resources



### Sprite remixed in a project



Figure 5-3: Use of the “textbox” sprite from Scratch Resources.

Despite this interest in sharing granular components, the ease of *integration* depends on several factors. Even when the source code is provided, there are some cases where projects are “impenetrable” because of their complexity or a mismatch between the expertise of the person trying to decompose a project and the complexity of that project. For example, sometimes sprites within a project are interconnected through the use of “broadcasts” that make it harder to take apart individual sprites other than those sprites that are self-contained. To address this, some Scratch creators add instructions to their projects explaining how to reuse their components. Some of them rely on the “comment” blocks, which are present in 2.1% of all projects.

However, some Scratch creators who do not want to see their projects remixed have figured out ways to obfuscate their code. A 12-year-old community member from New Zealand used the discussion forums to share a long tutorial on how to “hack” Scratch to obfuscate a project. He started by saying:

*There have been lots and lots of people who support the Locking Downloads button. There have also been lots and lots of people who disagree and support remixing. There has been a decision that the button shall be avoided. Yes, you may hate hackers ruining your projects. Well, here's the guide to make your projects confusing! But because this ruins remixing, we have decided to split this into two parts. The first one tells you how to help your project. And the second one tells you how to unravel the scripts.*

Other people jumped in and shared their own tips such as creating a “hidden” sprite by following a unique sequence of steps, as well as steps to hack the hack of hiding scripts.

### **5.1.2 Attributability**

Three months after the public release of the website, I added a feature that would automatically generate attribution information for all remixes. This feature was added as response to several conflicts caused by what some considered “plagiarism.”

The feature relies on a “watermark” that gets added to projects when they are uploaded. Then, subsequent downloads of a project have the provenance information embedded. When a remix is uploaded, the website reads the watermark and is able to link the remix with its antecedent project. At first, the remixes were identified as “mods,” but the name was difficult to understand, so it was changed to “remixes.” In chapter 6, I present an in-depth analysis of this intervention.

The issue of attribution is not unique to Scratch. For example, at first, attribution was optional when picking Creative Commons licenses. However, after analyzing several years of usage of the licenses, the Creative Commons managers found that very few people waived the attribution clause. This led them to include attribution by default in all their licenses, and create a separate license for completely public domain works (Brown, 2004).

### **5.1.3 Openness**

The Scratch Online Community allows any of its members to download and remix any project; in that sense Scratch is more open than similar websites, such as Newgrounds, which do not provide the license, the environment, or the technical features to freely open its content.

The ethos of openness is present in Scratch through its terms of use, the license used by all projects shared, and the community moderation style enforced by administrators.

However, this openness is not always understood or embraced by its members. Also, this openness is not always compatible with other systems. For example, there are some members of the community, who even after discussions with the administrators of the site, do not see enough value of openly sharing their work and decide to stop using the website. This was the case of one of the most popular animated artists who became increasingly antagonistic toward Scratch's openness as she saw people “stealing” her art and ideas.



## 5.2 Content Attributes

In the previous section I described the structural attributes of the Scratch Online Community that were intended to support remixing. However, even in a system that allows people to share creative works that are granular, modular, attributable, and open, there will be some works that are more generative (i.e. get remixed more) or more original ways than others.

In this section, we<sup>3</sup> pose the idea that there is a trade-off between generativity and originality in creative online communities. Evidence of this is provided by testing widely cited theories that suggest that the generativity of creative works is associated with complexity, author's status, and provenance. Each of these qualities is shown to be associated with decreased originality in the resulting remixes.

To advocates of remixing such as Lessig, and peer production more generally (for example, Raymond, 1999; Benkler, 2006; von Krogh and von Hippel, 2006), the fecundity or “generativity” of remixable projects determines remixing's very existence; a remixable project only becomes “peer productive” when it is remixed. However, despite the enormous amount of remixing that occurs in some of the poster children of remixing, most articles on Wikipedia and other wikis never attract many editors (Ortega, 2009; Kittur and Kraut, 2010), most Free and Open Source Software (FLOSS) projects founder (Healy and Schussman, 2003), the majority of YouTube videos are never remixed, and most attempts at “meme spreading” on 4chan fall flat (Bernstein et al., 2011).

In addition, even when peer-production projects are generative, remixing's detractors have suggested that most remixes are uncreative and of little cultural, innovative, or economic value (for example, Keen, 2007). This section attempts to understand how designers of peer-production systems might, or might not, be able to support remixing that is both *generative* (namely, likely to engender remix projects) and *original* (namely, remix projects differ substantially from their source projects).

Proponents of remixing have argued for generatively, claiming that it leads to increased innovation and democratized production. Zittrain (2008) argues that some technologies, such as the Internet, are generative and important not because they solve problems directly but because they provide rich and unconstrained platforms on which remixing technologies can be built. Previous research has tangentially looked at ways to promote remixing (for example, Cheliotis and Yew, 2009; Luther et al., 2010). That said, little is still known about what makes some works more generative than others.

Although remixes are defined by their generative nature, the promise of remixing is also tied to the *originality* of these remixed works. The usefulness of increased generativity may be limited if the resulting remixes are not transformative. Remixing as near-perfect copying seems unlikely to achieve Benkler's goal of “making this culture our own,” or in building the transformative and empowering improvements at the heart of Zittrain's

---

<sup>3</sup>based on joint work with Benjamin Mako Hill to be published in a special issue of American Behavioral Scientist.

examples. Furthermore, issues of originality are often at the center of moral and legal discussions around remixing (for example, Aufderheide and Jaszi, 2011).

Although scholars have examined some of the most successful examples of remixing and peer production in depth, little is known about why some projects become fertile ground for rich crops of creative remixes, while a large majority does not. One might assume that generativity and originality go hand in hand. After all, the concepts are tightly coupled in the most celebrated exemplars of remixing culture. But because so much attention has been given to these unusually successful examples, little is yet known about most peer production and remixing projects that, despite their efforts, are neither generative nor the source of transformative remixes.

This section examines two related research questions. First, *what makes some pieces of content more generative than others?* Second, *what makes some content engender more transformative remixes?*

### 5.2.1 Generativity

**Hypothesis 1A:** *Works of moderate complexity are more likely to be remixed than works that are straightforward or complicated.*

In his influential essay *The Cathedral and the Bazaar*, Raymond (1999) suggests that successful FLOSS projects are those, such as the Linux kernel, that “release early and release often.” Raymond equates larger, more complicated, and more polished software projects with what he calls the “cathedral” style of software development and contrasts them with the less structured—but, in his view, more promising—“bazaar” model that he equates with FLOSS. Raymond suggests that FLOSS projects such as Linux attract participants because they publish their code earlier allowing more feedback, bug fixes, and improvements. Similarly, Zittrain’s (2008) “Principle of Procrastination” suggests that generative technologies tend to be designed in a way that leaves most details for later. Zittrain posits that:

*Generative systems are built on the notion that they are never complete, that they have many uses yet to be conceived of, and that the public can be trusted to invent and share good uses.*

For example, he suggests that the Internet was a more effective platform for innovation than corporate networks such as *Prodigy* and *CompuServe* because its relative simplicity offered fewer constraints for potential innovators.

Although both Raymond and Zittrain suggest that increased simplicity is associated with increased generativity, it can be assumed this will not hold for extremely simple works. The earliest possible release of Linux would, by definition, do nothing. It seems unlikely that a featureless or extremely broken operating system kernel would excite and elicit contributions from other programmers as Linux did. Similarly, if the designers of the Internet procrastinated *completely* and created nothing, it seems unlikely that their system would have been an even more generative platform.

**Hypothesis 1B:** *Works that are created by high status authors are more likely to be remixed than works by lower status authors.*

Exposure is a prerequisite for remixing in that a work has to be seen to be remixed. In this sense, the popularity of a work is, by definition, related to its generativity. Theorists have suggested that the relationship between popularity and remixing may run deeper. The sources of remixing projects, unlike other forms of peer production, usually have identifiable authors. Lessig's (2008) key examples include music videos based on widely popular news footage, and popular music and films. In Lessig's account, the act of remixing is often understood as a social statement — often of parody or critique. Jenkins (2006) documents how youth use fan fiction to create remixes of popular and culturally salient products and symbols. Within particular communities, research has shown that more popular individuals attract more remixers (Cheliotis and Yew, 2009).

Using surveys and interviews with musicians, Sinnreich (2010) suggests that remixing is about creating explicit connections with previous, culturally salient, works and that “mash-ups are premised on the notion of recognizability and critique of pop culture.” As much as remixing relies on cultural salience, the expectation is that works that are more salient will be generative. However, because popularity of the work itself might merely measure exposure, salience should be operationalized by looking to the status or “fame” of a work's creator while controlling for the exposure of the work in question. In other words, after having been viewed the same number of times, it is expected that a work by a higher-status creator to be more generative than a work by a lower-status author.

**Hypothesis 1C:** *Works that are remixes themselves are more likely to be remixed than de novo projects.*

A third determinant of generativity suggested by theorists is provenance cumulateness, interpreted as the ability to create a remix out of something that is itself a remix. Using the example of Wikipedia articles, Benkler's (2002) theory of peer production suggests that much value can derive from an organization's ability to aggregate the contributions of many individuals. Murray and O'Mahony (2007) suggest that peer production can be thought of as a form of “cumulative innovation” and that value is created in products such as FLOSS through organizational, institutional, and social models that can aggregate the work from many individuals.

Empirical research has pointed to power law functions as characteristic of the distribution of contributors to FLOSS projects (Healy and Schussman, 2003) and the distribution of the number of remixes that individual works receive in music mixing communities (Cheliotis and Yew, 2009). Cheliotis and Yew (2009) and Maillart et al. (2008) have each suggested that these “long tail” distributions can be explained through a theory of “preferential attachment” where new contributors chose to participate in collaborative projects when there are more previous contributors. Although characterizing the distribution of remixes is outside the scope of this analysis, it is possible to test the theoretical implication that remixing in a system of preferential attachment will tend to be cumulative and that remixed-remixes will, on average, be more generative than non-cumulative *creative* works.

### 5.2.2 Originality

Although theory on the relationships between remixed media, their creators, and the nature of their remixes is less developed, theoretical justifications can be found for three hypotheses that parallel the hypotheses about generativity. In all three cases, these hypotheses suggest that the qualities associated with higher generativity are also associated with lower originality in the remixes that result.

**Hypothesis 2A:** *Remixes of moderately complex works are less original than remixes of works that are straightforward or complicated.*

We can think of “release early, release often” and the Principle of Procrastination as referring to costs associated with participation that more fully developed systems present to potential collaborators. Zittrain suggests that the generativity of a work will be determined, in part, by how easily new contributors can master it. In other words, if Zittrain’s Principle of Procrastination is true and relatively simple works are more generative, this might be because simpler content is accessible to a relatively larger group of potential remixers. If this larger group is motivated by the relative ease of contribution, we might also expect the group to be made up of relatively less skilled creators who, on average, produce less transformative derivatives.

Although it is also possible that more complex works are closer to “completion” than relatively simpler works and are therefore subject to less intensive improvements, we suggest that originality in remixes will be driven primarily by wider participation in the act of remixing by relatively unskilled contributors and that, as a result, remixes of works of intermediate complexity will tend to be less original than very simple or very complicated works.

**Hypothesis 2B:** *Remixes of works that are created by high status authors are less original than remixes of works by low status authors.*

Sinnreich suggests that the creation of highly remixed music is one way that mash-up artists create cultural resonance. In other words, when an artist remixes a famous song, the authors must maintain recognizability of the original. Sinnreich describes how mash-up artists avoid rare vinyl samples and prefer using popular songs. For example, several musicians interviewed by Sinnreich suggest that P. Diddy’s song “I’ll be Missing You” became a cultural and commercial success in part because it consisted largely of a minimally modified version of the 1983 song “Every Breath You Take” by the band The Police.

Although remixing is a form of cultural conversation or citation, it could be expected that the products of more popular or culturally salient remixes will be remixed lightly. Highly derivative remixes of culturally salient works ensure a high degree of recognizability of the base project in the remix. As a result, qualities that lead people to remix works from more famous or well-known creators may also lead remixers to keep large pieces of the work intact. When remixing the work of less well-known creators, the choice of a particular work might be driven by use-value and recognizability may play a less important role. Finally,

a tendency toward less re-creative remixing of works by higher status creators may also be driven by the fact that charges of plagiarism may carry more weight coming from well-known community members.

**Hypothesis 2C:** *Remixes of works that are remixes themselves are less original than remixes of de novo projects.*

As much as remixing is done to improve projects toward a stated or unstated goal, it could be expected that cumulative remixes (namely, remixes of remixes) begin with a more complete or less buggy project and, as a result, merely have less work to do. In formulating “Linus’ law” that, “with enough eyeballs, all bugs are shallow,” Raymond (1999) might be interpreted as suggesting that over time, software might get closer to a state of being “done.” In other words, although more people might become involved in a particular collaboration, contributions from these people will tend to be smaller as a work matures. Benkler’s theory of peer production suggests that it is lightly motivated individuals contributing small amounts who participate in some of the most collaborative, and cumulative, works of peer production.

Suggesting an important limit to Hypothesis 1C, Cheliotis and Yew (2009) observe that when a project is cumulative and the product of many subsequent reuses, it becomes less likely to be reused in future generations. As much as the “chain” network structure becomes decreasingly likely to continue as it grows in length, it could be expected that the existence of a shared goal (stated or unstated) for such cumulative work within the collaborative community may influence its continuation. Although Cheliotis and Yew (2009) do not present data on the originality of remixes, one explanation of their observation about chain remixes is that cumulative remixing will, on average, represent a process of refining and elaborating that has limits.

These hypotheses represent a partial theory in that they attempt to highlight three of the most widely cited theoretical determinants of generativity and originality. Available knowledge suggests none of these hypotheses has been tested empirically.

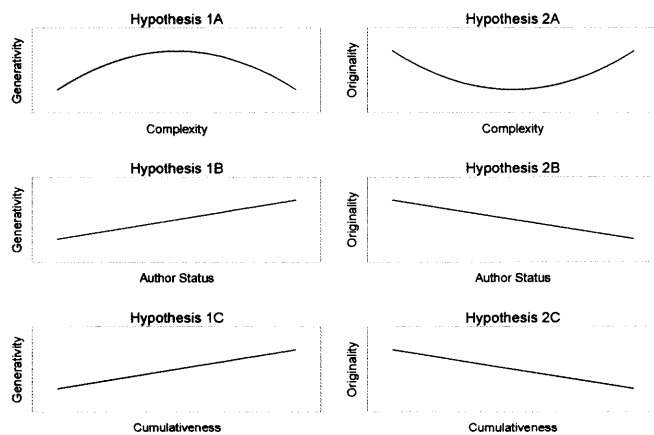


Figure 5-4: Visual representation of the hypotheses promoted in this section

## Methods

To test our hypotheses, we turn to the Scratch online community. Issues of originality and generativity play a prominent role in the Scratch community. Approximately 2% of remixes are flagged as inappropriate — often with accusations of unoriginality like, “This is MY own artwork he has uploaded without an ounce of originality.” On the other hand, Scratch creators often explicitly encourage others to remix their works, such as this user who shared a project that came with a note in the description asking for help fixing some bugs: “I need help with the following bugs in my game. 1. How to get the car to go up hills 2. How to get the character to rotate properly 3. It does not scroll [...] Please remix and help :-)” to which another responded by remixing and leaving the following note in the description of the project “i solved most of the bugs. i didn’t do the rotation for the character when in midair. [...] but i think i fixed all the other bugs.”

The originality of all the remixes<sup>4</sup> created during a one-year period from December 1, 2009 through December 1, 2010<sup>5</sup> was computed using the originality metric described in the previous chapter. This dataset included 536,245 individual projects. The projects in our dataset were shared by 105,317 unique users ( $u$ ). Because our data is longitudinal, we track each project for one year and, for time-varying measures, present measures at the end of the one year period.

To answer our first three hypotheses, we operationalize generativity in two related ways. First, we construct a count of the number of remixes of each project shared within the first year of the project’s publication ( $remixes_p$ ). Second, we construct a dummy variable indicating if a project has been remixed by another user at least once in the one year period subsequent to being shared ( $remixes_p > 0 \Rightarrow remixed_p = 1$ ). Because individuals may choose to remix a previous remixed project because others have remixed it, we suggest that our dichotomous measure ( $remixed$ ) offers a more reliable, if more conservative, measure of generativity.

We operationalize complexity of projects as blocks ( $blocks_p$ ). Blocks are analogous to tokens or symbols in source code for computer programs. Blocks are similar to, but more granular than, source lines of code which have a long history of use as a measure of both complexity and effort in software engineering (Walston and Felix, 1977; Albrecht and Gaffney, 1983).

Data on user interactions in Scratch provide a series of possible measures of a given user’s status within in the community. Possible indicators of status include the number of past expressions of admiration or “loveits” (analogous to “liking” something on other social media platforms), total past views by other users, total previous selections of a user’s work as another’s favorite, and total past downloads — each variable is measured at the level of the

---

<sup>4</sup>Due to technical errors 1,182 projects were not able to be analyzed. Also, 130,800 projects (20%) were omitted because they were removed from the site by their creators, although we find, in robustness checks not reported here, that our results are not substantially different when we include them.

<sup>5</sup>We selected data from 2010 because the website and its community were mature and stable during this period and because the Scratch website did not undergo any significant design changes that might have affected remixing behavior.

Table 5.1: Summary statistics for variables used in our analysis. Measures with the subscript  $p$  are measured at the level of the project while measures with the subscript  $u$  are measured at the level of the user.

Variable	$N$	Mean	SD	Min	Max
<b>Dependent Variables</b>					
Remixes > 0 times in 1yr. ( $remixed_p$ )	536,245	0.07	0.26	0.00	1.00
Remixes within 1yr. ( $remixes_p$ )	536,245	0.15	1.78	0.00	658.00
Edit Distance (Mean) ( $distance_p$ )	37,512	85.57	397.66	0.00	21,970.00
Edit Distance (Highest)	37,512	114.2	507.34	0.00	30,813.00
<b>Question Predictors</b>					
Number of blocks ( $blocks_p$ )	536,245	99.60	476.19	0.00	196,509.00
User's previous "loveits" ( $loveits_{up}$ )	536,245	32.47	153.19	0.00	10,526.00
Remix status ( $isremix_p$ )	536,245	0.18	0.38	0.00	1.00
<b>Controls</b>					
User age in years ( $age_{up}$ )	523,092	17.57	11.62	4.00	74.75
Account age in months ( $joined_{up}$ )	536,245	4.79	7.18	0.00	45.43
User is Female ( $female_u$ )	536,222	0.37	0.48	0.00	1.00
Blocks per sprite ( $blocks/sprites_p$ )	536,245	11.82	22.75	0.00	3111.50
Views within 1yr. ( $views_p$ )	536,245	13.57	69.90	0.00	4977.00

project or work but is aggregated for each project’s creator at the point in time that a project is shared. Because these measures are highly correlated ( $0.84 < \rho < 0.97$ ) we operationalize status as total previous “loveits” ( $loveits_p$ ) at the moment that the project in question was uploaded but our results are similar using the other indicators. We operationalize cumulativeness using a dummy variable that indicates whether a project is, itself, a remix of another project ( $isremix_p$ ).

Finally, we include a series of control variables that may also be associated with the generativity of projects and with the originality of subsequent remixes. For each user, we include self-reported measures of gender which we have coded as a dichotomous variable ( $female_u$ ), date of birth which we have coded as age in years at the moment that each project was shared ( $age_{up}$ ) which may indicate sophistication of the user, and age of each account ( $joined_{up}$ ) which may indicate the level of experience of a user with Scratch. For each project, we are primarily concerned with the effect of exposure on the likelihood of remixing so we attempt to control for views using the number of times that each project was visited in its first year on the site ( $views_p$ ). “Sprites” are the objects in Scratch projects to which code is attached. Because more modular projects may be easier to remix, we also calculate a measure of the average numbers of blocks per sprite ( $blocks/sprites_p$ ) which may act as a very coarse measure of modularity.

To answer our second set of hypotheses, we create a new dataset that includes only the subset of 37,512 projects that were shared in the community during our one-year window and were remixed at least once in the following year.<sup>6</sup> As described in the previous chapter, we operationalize the originality of a remix using a calculation of the degree to which a project diverges from its antecedent.

Of course, a given project can be remixed multiple times. In fact, 11,704 of the projects remixed in the window of time analyzed (31%) were remixed more than once within a year of being shared. The distribution of remixes was highly skewed, for example, the mean number of remixes in our sample was 2.14 but the highest value of 658. As a result, our measure of edit distance is the mean edit distance of all remixes shared in the year following a project’s publication on the website ( $distance_p$ ).

## Analysis

Our analytic strategy involves the estimation of a series of two sets of parallel models. In both cases, we include variables operationalizing project complexity, creator status, and project cumulativeness that correspond to our three sets of parallel hypotheses. Both *blocks* and *loveits* are highly skewed but a started log transformation results in an approximately normal distribution in each case. Hypothesis 1A and Hypothesis 2A predict a curvilinear relationship between the dependent variables and our measure of complexity. As a result, we include a quadratic specification for log *blocks* in each model and focus our interpretation

---

<sup>6</sup>Due to technical errors or corrupted project files, we do not include 1,217 projects that site-metadata indicates were remixed but that we were unable to analyze. We believe that these errors were due to random corruption and are unlikely due to bias our results.



on the coefficient associated with the quadratic term which will determine the direction of the curve.

Providing tests of Hypothesis 1A, Hypothesis 1B, Hypothesis 1C; our first set of models tests generativity in the full dataset of 523,069 projects shared in our window of data collection for which we have complete information<sup>7</sup>. In our first and more conservative test, Model 1, we use logistic regression to model the likelihood of a project being remixed at least once on our sets of predictors and controls.

### 5.2.3 Results

Drawing from theory and presenting detailed empirical evidence, the data suggests that the three factors associated with higher levels of generativity — moderate complexity, creator status, and cumulative provenance — are also associated with decreased originality in resulting remixes.

We see support for the hypothesis that posits that works of moderate complexity are more likely to be remixed than works that are very simple or very complicated. We see the inverted “U” shape in the relationship between complexity and generativity in the negative coefficient on the quadratic term in Models 1 and 2 (see Table 5.2). Holding other variables at their sample mean, Model 1 predicts that slightly more than 1 percent of projects will be remixed at both the minimum (0 blocks), and maximum (196,509 blocks) in the sample. That said, this support comes with a crucial qualification. For most projects, increased complexity is associated with increased generativity – in opposition to Zittrain’s Principle of Procrastination and Raymond’s call to “release early, release often.” For most complicated projects in Scratch, marginal increases in complexity are associated with an increase in generativity.

The distribution of projects by *blocks* is highly skewed toward more straightforward projects with the median project having only 26 blocks. In other words, although the most complicated projects are indeed at lower risk of being remixed, the model predicts that projects have an increasing likelihood of being remixed into the 95<sup>th</sup> percentile of projects by complexity. Even among complicated projects, the relationship is effectively flat. For example, holding all other predictors at their sample means, Model 1 estimates that 6.75% of projects with 385 blocks (the 95<sup>th</sup> percentile) would be remixed. Nearly the same proportion (6.73%) of otherwise similar projects with 1,204 blocks (the 99<sup>th</sup> percentile) would be.

As predicted, it is seen that projects that are more complex are associated with remixes that are more original. That said, there is no evidence suggesting straightforward projects are also associated with increased originality in remixes. Indeed, holding other variables at the sample mean, Model 3 estimates that a project with 3 blocks (10<sup>th</sup> percentile) will have an average edit distance of 20 blocks while an otherwise similar project with 211 blocks (90<sup>th</sup> percentile) would, on average, be associated with a mean edit distance of 80 blocks.

Support is found for Hypothesis 1B, which posits that works that are created by high status

---

<sup>7</sup>We omit 13,176 projects for which we are missing age or gender information.

	Generativity		Originality
	Model 1 (P[remixed])	Model 2 (remixes)	Model 3 (distance)
(Intercept)	-4.997* (0.030)	-5.035* (0.028)	2.288* (0.050)
log <i>blocks</i>	0.524* (0.016)	0.373* (0.016)	-0.026 (0.027)
log <i>blocks</i> <sup>2</sup>	-0.037* (0.002)	-0.035* (0.002)	0.051* (0.003)
log <i>loveits</i>	0.016* (0.004)	-0.022* (0.004)	-0.070* (0.007)
<i>loveits</i>	0.791* (0.045)	0.429* (0.045)	-1.023* (0.071)
<i>age</i>	-0.000 (0.001)	0.006* (0.001)	0.006* (0.001)
<i>joined</i>	-0.005* (0.001)	-0.003* (0.001)	0.008* (0.001)
<i>female</i>	-0.004 (0.012)	0.114* (0.012)	-0.334* (0.021)
log <i>blocks/sprites</i>	-0.519* (0.012)	-0.374* (0.012)	0.294* (0.018)
log <i>views</i>	0.850* (0.006)	1.044* (0.006)	0.169* (0.009)
log <i>blocks</i> × <i>isremix</i>	0.321* (0.025)	0.304* (0.026)	0.148* (0.040)
log <i>blocks</i> <sup>2</sup> × <i>isremix</i>	-0.045* (0.003)	-0.032* (0.003)	-0.000 (0.005)
$\theta$		0.265* (0.003)	0.302* (0.002)
<i>N</i>	523,069.	523,069.	36,722.
<i>AIC</i>	219,914.053	307,783.718	313,305.804
<i>BIC</i>	220,450.092	308,364.426	313,748.382
log <i>L</i>	-109,909.027	-153,839.859	-156,600.902

Table 5.2: A series of fitted regression models testing our six hypotheses. Models 1 and 2 test H1 on on the full dataset of projects between Dec ‘09 and ‘10. Model 1 is a logistic regression model which models the likelihood of a project being remixed at least once within 1 year after being shared. Model 2 is a negative binomial regression to model a count of the times a project will be remixed within a year of being shared. Model 3 test H2 on a count of the mean edit distance for all projects remixed within a year of being shared.

authors are more likely to be remixed than works by lower status authors. Holding other variables at their sample mean, Model 1 predicts that the odds of being remixed are slighter higher (1.02 times) for each log unit increase in the number of previous loveits the project’s creator received, and that the result is statistically significant.

Support is also found for Hypothesis 2B, which predicts that remixes of works that are created by high status authors are less original than remixes of works by low status authors. We see a negative relationship between creator status and originality of remixes. Holding other predictors at their sample mean, it is estimated that remixes of a project whose creators had received no previous “loveits” (10<sup>th</sup> percentile) would have an average edit distance of 40 blocks. An otherwise identical project whose creator had received 64 previous “loveits” (90<sup>th</sup> percentile) would be estimated to have an average edit distance of 30 blocks.

Model 2 adds important qualification to the support for the hypotheses about status and generativity. Although the likelihood of a project being remixed at least once is positively associated with author status as measured by “loveits,” generativity is negatively associated with status when operationalized as the number of remixes in one year. In other words, author status is a positive predictor of whether a project will be remixed but is associated with fewer remixes for any project.

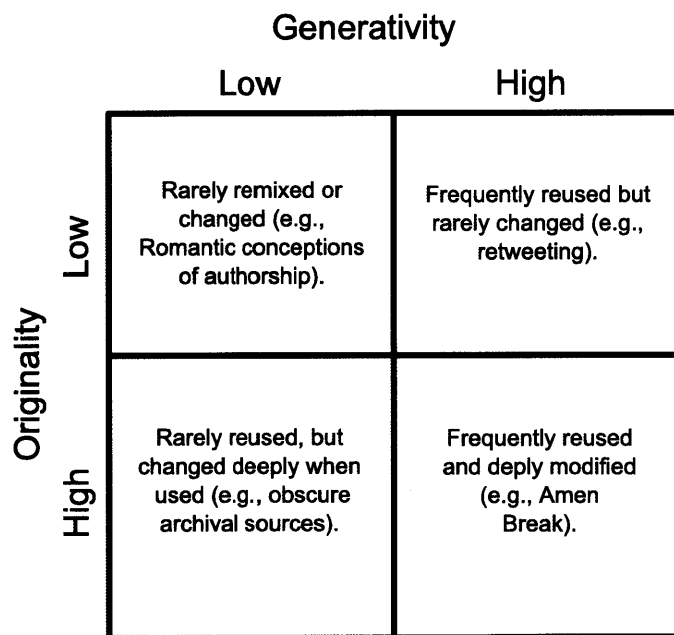


Figure 5-5: Diagram of the relationship between originality and generativity

The results provide strong support for both the prediction that works that are remixes themselves are more likely to be remixed and less original than de novo projects (Hypothesis 1C and Hypothesis 2C). Tests of the association between cumulativeness and measures of originality and generativity are captured by the parameter estimates associated with *isremix*. Model 1 suggests that the odds that a remix will be remixed is 2.21 times higher

than the odds that an otherwise similar *de novo* project will be. Model 2 suggests that remixes will also be remixed more often during the year after being shared. In strong support of Hypothesis 2C, Model 3 suggests that these remixed remixes will tend to be much less original. Holding other qualities at their sample means, Model 3 estimates that a remixed remix will have an edit distance that 17 blocks smaller (an average of 23 blocks) than a similar *de novo* project (40 blocks).

#### 5.2.4 Limitations

There are several important limitations and threats to the validity of the results presented above. First, although the data are longitudinal, the analytic strategy follows each project for one year. Model 1 treats projects as remixed only if they are remixed in one year. Projects can and are remixed after one year and there is a risk the results are biased by the fact that “late bloomers” are systematically different from other projects in ways that are correlated with the predictors.

Although the analysis does not include these late bloomers, the analysis can confidently claim to consider most of the remixing activity. We found that 30 days after being shared, 8% of projects are remixed although after a full year, 10.8% are. Following projects for an additional two years, it is estimated that only an additional 2.4% projects will have been remixed.

A second concern is that the use of average edit distance in Model 3 may lead us to conclude that projects that are generative will tend to have lower edit distances merely because all projects are susceptible to uncreative remixing and that being remixed more often puts projects at increased risk from these straightforward remixes. This threat is addressed by reestimating Model 3 using the highest edit distance of any previous remix as an alternate indicator of originality. The results of this model are largely unchanged from Model 3 reported in Table 5.2 and, indeed, are even stronger in estimates of the effect of status and cumulateness.

Third, there is an important concern with blocks as an indicator of complexity. Blocks will not capture complexity in ways that do not involve programming, such as storytelling and visual arts, and can be “cut and pasted” in a way that may not correspond to complexity through increased effort. Possibly, complicated cut-and-paste projects are skewing the results for complexity. This can be addressed with a unique measure available in Scratch. Each Scratch project records the time and date of each time a user clicks the “save” button as well as the time that the user shares the project. The time between the first “save” and the point at which the user shared the project can be used as a proxy for complexity as measured by effort.

This alternative measure is noisy in the sense that some users may not share a project for hours, days, or weeks, but not spend that entire period engaged in work on the project. Additionally, 44% of the projects in the time window were shared without ever being saved once so values on this indicator are missing. With these limitations in mind, it is possible to reestimate Models 1 and 2 on the subset of 298,926 projects for which there is data and

replace the measure of blocks with “minutes-to-share” (*MTS*). The results in modeling generativity using this alternative specification are essentially unchanged. A similar reestimation of Model 3 using the 22,048 remixed projects with *MTS* data did not find support for either the quadratic specification of *MTS* or its interaction with *isremix* but offered substantively similar predictions in its estimation of a positive linear association between edit distance and complexity, which leave the findings fundamentally unchanged.

In other robustness checks, the addition of random effects to control for possible clustering due a single user uploading several projects led to findings that the estimates and results are unchanged. Robust estimation of standard errors is used to address concerns of potential heteroscedasticity. Using robust estimates for Model 3, the interaction terms between *blocks* and *isremix*, already substantively similar, are rendered statistically insignificant. The rest of the results, and the findings for each of the hypotheses, are unchanged.

There also remain several less testable, but important, limitations. First, the originality of creative works is often subjective and open to interpretation. Edit distance, the quantitative measure of originality used, is likely correlated with this subjective sense, but will miss nuanced and subjectively important signs of originality among Scratch creators. Additionally, the lack of a measure that incorporates artistic originality means that the measure is focused solely on code. These limitations in the data and methods are acknowledged, it is hoped to explore other ways of measuring originality in future work.

### 5.3 Discussion

In this chapter, I introduced a framework to assess remixability by examining the attributes of the system and the content potentially being remixed. Then I presented an analysis of three theoretically motivated hypotheses on the determinants of generativity—moderate complexity, creator status, and cumulateness—and revealed at least some support for each.

Surprisingly, the weakest and most tentative support was for the most frequently cited theoretical relationship: Zittrain’s Principle of Procrastination or Raymond’s suggestion to “release early, release often,” which both imply that easier works will tend to be generative. Some support for this hypothesis was found, but only among the most complicated projects. Instead, largely positive relationships were found between complexity and both generativity and originality over most of the data. Perhaps Scratch’s young and amateur users are unlikely to create projects that are complex enough to trigger the effect.

Taken together, the results suggest a paradox: qualities of projects associated with increased remixing are also associated with lower originality in remixing’s outputs. These results challenge assumptions in the cultural, technical, and political discourse on remixing and content reuse, offer a nuanced model of remixing, and suggest that designers of socio-technical systems may need to trade off the possibility of more collaboration with avenues for deeper and more meaningful interaction.

For designers, the findings point to complex decisions around manipulating the visibility of

variables such as author status and project complexity. For example, designers of a new peer-production system in need of more content might want to emphasize author status and remix chains to get more content through generativity. They should be aware that this might come with a cost in terms of the originality of the remixed works. The results suggest that supporting increased complexity, at least for most projects, might have fewer drawbacks.

That said, many of the most readily available incentives for designers are status or reputation based levers, which might result in a trade-off in originality because of the incentive itself. It may also be important to avoid rewarding correlates of generativity for their own sake when it is generativity that a designer wants to encourage. Perhaps an alternative would be to highlight those rare projects that lead to generative and creative remixes.

Nothing shown in this section devalues the promise of remixing in terms of peer production, culture, and innovation. Indeed, it is believed that societies' ability to harness the power of remixing is deeply important, but requires many further analyses such as this. Though the results suggest that highly generative works that lead to highly creative remixes may be rare and difficult to support, it is not suggested that encouraging them is anything but a worthwhile and crucial goal.

Using data from Scratch, evidence is presented that supports and extends several widely held theories about the foundations of generativity and originality. The results suggest that designers of online collaborative communities may face a dilemma obscured by those celebrated exemplars of peer production communities: that system designs that encourage and support increased rates of remixing may also result in products that are more superficial.

## Chapter 6

# Attitudes toward Remixing

**Interviewer:** *How would you define remixing?*

**9-year-old boy:** *Taking somebody else's project, and then changing a lot of it, and then sharing it, and giving credit*

In this chapter<sup>1</sup> I examine *how young people react to remixing in the Scratch Online Community*. I explore this through five studies. The first three explore how people respond to remixing, especially in light of conflict. The fourth and fifth explore the success of a technical response to address those conflicts.

This analysis is driven by an interest in understanding how the system design may influence these attitudes, and these issues are analyzed from the perspective of people whose projects are remixed as well as from those creating the remixes.

### 6.1 Study 1: How do people respond to remixing?

In this study we assessed Scratch users' reactions to having their projects remixed. We identified all antecedent-remix pairs in the first 13 months of the Scratch community. These pairs were all made of projects based on another project and the corresponding created project, and all comments left by the author of the source project on each remix project.

The algorithm identified projects based on embedded metadata. As a result, no projects that were conceptually copied by a user who had seen another's work but who did not actually copy code, graphics, or sounds were counted. Additionally, as there were more than 100,000 projects, it was not feasible to watch and interact with each project and determine whether the remix projects were actually "original" or whether ideas were taken from a source outside Scratch (for example, a user might have created a Pac-Man clone).

Broadly speaking, people whose projects are remixed react either by being indifferent to it, accepting it, conditionally accepting it (specific rules or norms have to be followed), or by

---

<sup>1</sup>Based on (Monroy-Hernández et al., 2011) and a coauthored article with Hill et al. (2010).

explicitly opposing it. Similarly, remixers go about remixing by either being oblivious of the norms, exhibiting caution by asking for permission first, or even being confrontational and using remixing as a form of “trolling” (Donath, 1998).

If a user created a Pac-Man clone and it was then modified or improved by another Scratch user, the second user would be considered a “remixer” in the dataset although the first would not.

### 6.1.1 Procedure

We analyzed all 136,929 projects created and posted on the Scratch website between the community launch date in March 2007 and April 2008; 11,861 projects were deemed to be remixes. The comments left by creators on the first 3,555 projects were coded by two independent coders asked to put them into the following categories: *no comment* (projects in which the remixee did not leave any comments on the remixer’s project), *positive* (projects in which the remixee left positive comments, for example, “Love what you did with my code! Great idea!”), *hinting plagiarism* (projects in which the creator implied that the remixer had copied but did not state this explicitly, for example, “I mostly pretty much made this whole entire game”), *plagiarism* (projects in which the creator directly accused the remixer of copying, for example, “Hello Mr. plagiaries,” “Copycat!”), *negative* (projects with negative comments that were not necessarily related to copying, for example, “All right you crap eating thumb sucking baby”), and *none of the above* (projects with comments that were not positive, negative, or relevant to plagiarism, for example, “is this jarred” or “b for peanut butter jelly time!”).

The first and last categories were incompatible with all the others. The other categories were potentially overlapping (an creator could say, “you copied me but I like your addition of the flowers” and therefore count as *positive* comments and *plagiarism*), except *hinting plagiarism* and *plagiarism*, which were incompatible with each other. The coders were found reliable (absolute agreement by category: *no comment* = 100%; of those who did include a comment, coders agreed on the presence or absence of the following categories at the following rates (absolute agreement): *positive* = 87%, *hinting plagiarism* = 81%, *plagiarism* = 89%, *negative*=93%, *none of the above* = 89%). Therefore the remaining comments ( $n = 8,306$ ) were split between the two coders, with each coding approximately half the remaining projects. For the few projects that the coders disagreed on in the initial third of the projects, they met and agreed on the coding.

### 6.1.2 Results

Out of the 11,861 projects that were categorized by the algorithm as remixes, it was determined that 3,742 (31.5%) of the creators clicked and saw the remixed versions of their projects. Of those creators who saw the projects, 2,156 (58%) did not leave a comment. Of those who saw the projects and commented, 261 (7%) accused the remixer of plagiarism, 566 (15%) hinted at plagiarism concerns, 797 (21%) left positive comments, 260 (7%) left



negative comments, and 237 (6%) left comments that did not fit into these categories or were uninterpretable.

### **6.1.3 Discussion**

The results from Study 1 indicate that users on the Scratch online community have extensive responses to remixing. People who responded were just as likely to leave positive comments (21%) as they were to leave a direct or indirect complaint of plagiarism (22%). These results, however, leave open the question of why extensive comments were left by creators. Studies 2 and 3 explore several potential answers to these questions.

### **6.1.4 Implications for design**

The Scratch online community was designed explicitly as a platform for sharing and remixing media. Despite the fact the Scratch infrastructure provided the technical facilities to remix content easily, a set of explicit norms and licenses communicated to users through links to a child-friendly version of the pro-remixing license on every project, and continuous proselytizing of remixing by the administrators of the site, many users reacted negatively to remixes and expressed a sentiment that remixers had plagiarized their work.

Creative Commons has described its work, both through the creation of licenses permitting remixing and through the creation of technological systems built around RDF (Resource Description Framework) metadata, as means of reducing permission-asking (Lessig, 2008). As with Scratch, many social media and remixing communities use CC's legal and technological systems. To the degree that the results generalize, the findings suggest that the technical and normative permission to create remixes may be insufficient to supporting positive reactions to remixing in a social media remixing community.

## **6.2 Study 2: When do creators accuse remixers of plagiarism?**

The many reactions to remixing shown in Study 1 can be explained in several ways. In Study 2, the results of Study 1 were used to construct a variable measuring whether creators have accused a remixer of plagiarism. This construct was then used as the dependent variable in a series of fitted logistic regression models to provide initial tests of support for several explanations of why creators may accuse remixers of their work of plagiarism using additional data on projects and their creators.

One difference between Scratch and other online peer-production communities is that many Scratch projects are constructed at enormous individual effort. On Scratch, users share full-fledged games or animations with code, artwork, and sound. This is in contrast to many peer production communities, such as Wikipedia, where users usually contribute smaller portions of articles or small fixes. One explanation for the many complaints on

	Mean	SD	1	2	2	3	5	6	7
ACCUSE.PLAG	0.13	0.34							
SPRITES	9.05	12.24	0.08						
ORIG.REMIX	0.05	0.21	-0.01	0.00					
HAS.REMIXED	0.79	0.41	-0.01	-0.07	0.11				
FEMALE	0.25	0.43	-0.06	-0.11	0.04	0.11			
WEEKS	35.15	14.49	0.01	-0.03	0.08	0.15	0.15		
AGE	17.08	10.15	-0.07	0.08	0.01	-0.08	-0.20	-0.16	
REMIXER.AGE	15.01	9.58	-0.05	0.01	0.06	-0.09	-0.05	-0.05	0.11

Table 6.1: Means, standard deviations, and correlations among variables used in the logistic regression analysis in Study 2. The sample includes all remixed projects that had been clicked and viewed by the creators. ( $n = 3,742$ )

Scratch may be that Scratch users develop a stronger sense of ownership because of the large amount of individual time and effort creators invest in their projects. This sense of ownership may set users up to be more protective of their work and more likely to accuse remixers of plagiarism. This explanation leads to the first hypothesis (H2-1): *Creators of larger or more complicated contributions will be more likely to accuse remixers of their projects of plagiarism.*

In many peer-production communities, such as Wikipedia, the vast majority of contributions are to existing products and work is primarily cumulative in nature. In the sample of 11,861 remixes from the first year of Scratch’s activity, most remixes (11,493 or 97%) were based on *de novo* projects while the remaining were second-generation remixes. If users feel more protective of projects that are entirely the product of their own work, this may explain the many complaints in Scratch. As many projects are *created projects*, Scratch users are more likely to feel plagiarized when their work is remixed. This leads to the second hypothesis (H2-2): *Creators will be less likely to accuse remixers of their projects of plagiarism when the remixed project is itself a remix.*

A final explanation extends this reasoning from the project level to the individual. Perhaps the process of creating remixes encourages creators to be empathetic toward remixers and to integrate these users into a “remixing culture” where copied or slightly modified projects are not seen as plagiarism but as positive contributions. In the sample, most authors of remixed projects were active contributors who have, at some time, created their own remixes. Indeed, only 26% of users in the sample ( $n = 3085$ ) had never shared a remix. Possibly, many charges of plagiarism come from users who have not been integrated into Scratch’s “remix culture” and oppose remixing in general. This explanation leads to the formation of the third hypothesis (H2-3): *Creators will be less likely to accuse remixers of their projects of plagiarism if they have shared at least one remix themselves.*

Of course, other factors are likely to have an important impact on responses to remixing in Scratch that make it necessary for them to be controlled. For example, charges of

plagiarism may be due, in part, to the many projects created by males in the sample (only 25% of creators were female) who may be more likely to accuse others of plagiarism. Additionally, the proportion of projects that are remixes has increased over the life of the study. This suggests that attitudes toward remixing might have changed over time with the potential for a change in plagiarism accusation rates. Finally, younger creators may be more likely to accuse remixers of plagiarism either because they do not understand that they are permitting others to remix by sharing their work, because younger users are more likely to react negatively in general, or for many other factors that correlate with age. As a result, it became necessary to control for gender, the period when projects were shared, and age, when testing the above hypotheses.

### 6.2.1 Procedure

To explore these issues, we gathered a sample consisting of the 3,742 remix projects that had been clicked and viewed by the creators of the antecedent projects, as described in Study 1. Viewing a project is both a low bar for involvement in the community and a prerequisite for any type of response to remixing — the subject of the study — even if that response is a decision not to act.

The Scratch online community is run using a custom-built web application with data stored in a MySQL database; data collection was for each of the predictors from this source. The dependent variable is a dichotomous variable (*accuse.plag*), constructed using the results in Study 1, which measures whether creators accused remixers of plagiarism in a nonpositive manner. It is a dummy variable that takes the value of 1 when a comment was coded either *plagiarism* or *hinted plagiarism* unless the comment is also coded *positive*. The discussion explains that several alternative specifications of this outcome were explored with similar results.

Project complexity can be measured either through the amount of programming code or the total number of graphical characters (*sprites*) controlled by these scripts. Because these measures were highly correlated ( $r = 0.80$ ), the choice was to use *sprites* alone as the measure of complexity. To aid in interpretation in the models below, *sprites* are reported in standard deviation units in the fitted models. To test hypothesis H2-2 regarding the effect of cumulative contribution on responses, a dummy variable (*orig.remix*) was constructed indicating whether the created project was, itself, a remix. Similarly, a dummy variable (*has.remixed*) was constructed indicating whether the creator has ever uploaded a remix, to test H2-3.

For the controls, gender is a dummy variable (*female*) indicating whether the creator is female and was measured through self-reported data from users' registration with the Scratch website. Time was measured based on upload data in the web application database. Because there was reason to believe that the effect of time on plagiarism accusations might be nonlinear, the controls included the quadratic form of a variable measuring the number of weeks since the first project was uploaded to the live Scratch website (*weeks*). This enabled the measurement of the age of users (*age*) through a self-reported birth month and birth year fields in the Scratch registration for both the remixer and creator. Age data

was marked as missing for users with ages under 4 and more than 90 (139 observations for remixers and 124 for creators). Ages were calculated at the day the remixed project was uploaded. Both ages are skewed toward younger users with median values of 13 and 12 respectively — several years below the mean. A correlation table with means and standard deviations of all the variables included in the models is shown in Table 6.1.

## 6.2.2 Results

	Model 0	Model 1	Model 2	Model 3	Model 4
(Intercept)	-1.904*** (0.049)	-2.159*** (0.312)	-2.301*** (0.315)	-2.302*** (0.315)	-2.287*** (0.325)
FEMALE		-0.534*** (0.127)	-0.496*** (0.128)	-0.495*** (0.128)	-0.494*** (0.129)
WEEKS		0.064*** (0.019)	0.064*** (0.019)	0.064*** (0.019)	0.064*** (0.019)
WEEKS <sup>2</sup>		-0.001*** (0.000)	-0.001*** (0.000)	-0.001*** (0.000)	-0.001*** (0.000)
AGE		-0.029*** (0.006)	-0.031*** (0.006)	-0.031*** (0.006)	-0.031*** (0.006)
SPRITES (std)			0.210*** (0.042)	0.210*** (0.042)	0.209*** (0.042)
ORIG.REMIX				-0.059 (0.247)	
HAS.REMIXED					-0.022 (0.124)
REMIKER.AGE					
<i>N</i>	3742.	3615.	3615.	3615.	3615.
AIC	2888.162	2758.412	2736.849	2738.790	2738.816
BIC	2913.072	2882.269	2885.477	2912.190	2912.216
log <i>L</i>	-1440.081	-1359.206	-1344.424	-1341.395	-1341.408

Standard errors in parentheses

† significant at  $p < .10$ ; \*  $p < .05$ ; \*\*  $p < .01$ ; \*\*\*  $p < .001$

Table 6.2: Taxonomy of logistic regression models on `accuse.plag`, a dichotomous construct representing whether project creators accused the remixer of their project of plagiarism in a nonpositive manner.

Results of the fitted regression models are shown in Table 6.2. Model 0 is the unconditional model and Model 1 is the control model, which adds variables controlling for the creator’s gender, the quadratic term measuring weeks between the created project upload and Scratch’s launch, and the age of the creator. The effect of creator gender on the likelihood of plagiarism accusations is highly statistically significant, large, and stable across

later specifications. Indeed, the model estimates that, robust to the addition of all the other controls in the model, the odds of a female accusing a remixer of plagiarism is less than 0.6 times the odds of males doing so. Both parameters in the quadratic terms measuring the number of weeks since the projects were uploaded are statistically significant and robust across specifications. Finally, the measure of creator age also affects the outcome that is statistically significant and robust across later specifications. Controlling for gender, younger students are indeed more likely to accuse a remixer of plagiarism. Testing for a quadratic age term and an interaction between age and gender found no statistically significant effect of either on the outcome. Although skeptical that the effect of age on plagiarism accusation rates is linear, it is suspected that this result is a factor of the data, which is largely limited to younger users where the relationship may indeed be estimated as such.

Model 2 adds a measure of complexity, a variable measuring the number of sprites in a project in standard deviation units, an estimate of which is associated with a higher likelihood of plagiarism accusations. With the controls in the model, it is estimated in Model 2 that the odds that an creator of a project will accuse a remixer of their project of plagiarism are 1.23 times higher than the odds that the creator of a project with one standard deviation (12.2) fewer sprites will do so. Consequently, the study finds support for hypothesis H2-1 in that, even with the addition of controls for age, gender, and when the project was posted, creators are more likely to accuse remixers of plagiarism when the project is more complex.

Model 3 adds the dummy variable indicating whether the created project in question is a remix itself. This model does not find a statistically significant effect of this dummy variable on the outcome; therefore, it is impossible to reject the null hypothesis that creators are as likely to accuse remixers of their projects of plagiarism when the remixed project was itself a remix as when it was a created production. As a result, the study does not find support for hypothesis H2-2 that, controlling for gender, age, and project complexity, creators will be less likely to accuse remixers of their projects of plagiarism when the remixed project is itself a remix.

Model 4 instead adds to Model 2 the dummy variable reflecting whether creators have ever uploaded a remix themselves. Again, the study does not find a statistically significant effect of this predictor on the outcome and, as a result, also fails to find support for the final hypothesis H2-3 that, controlling for gender, age, and project complexity, creators who have uploaded remixes themselves are less likely to accuse remixers of their work of plagiarism.

As a robustness check, the study re-estimated the models on a dataset that excluded projects shared before May 15, 2007, the first day that widespread press reports of the Scratch community were broadcast. In the period before, users were a smaller subset who might have been more likely to know each other in person. The results were not substantively affected. Further models, estimated on a dataset that did not exclude implausibly high and low ages, found that the results were similar again.

The study also estimated models using slightly different specifications for the dependent variable. Because many negative reactions by creators are due to plagiarism but do not

explicitly call it out, use was made of a specification of the dependent variable also true for negative reactions, which did not specify plagiarism with similar results. The dependent variable was also reformulated so it only included explicit charges of plagiarism that were not paired with positive messages (namely, *hinting plagiarism* charges were not included). The results were again substantively unchanged.

### 6.2.3 Discussion

The study found support for the theory that creators are more likely to accuse remixers of plagiarism if the remixed project is more complex. To the degree that the results generalize to other online communities, charges of plagiarism may be of reduced concern in communities where individual contributions tend to be small.

Surprisingly, the models suggest no effect of whether the project was itself a remix on the rate of plagiarism accusations. This might indicate that Scratch users accusing remixers of plagiarism have a strong conception of “good” (creative and transformative) remixes and “bad” (plagiarizing) remixes, which are simple copies. In line with this explanation, the study did not find support for hypothesis H2-3; creators who have uploaded remixes were neither more nor less likely to accuse remixers of plagiarism than users who had never uploaded a remix.

Although included as a control, the effect of age suggests intriguing future research. Future work could be designed to address why younger children may be more likely to complain about plagiarism. For example, one possible explanation is that young remixers do not understand licensing. On the other hand, previous qualitative work suggests that, although significant, other factors may put important limits on the understanding by or desire of users to pay attention to licenses. For example, Diakopoulos et al. (2007) showed that adult users on an online video sharing site asked for permission before reusing media, despite licensing considerations that made it clear that such use was legally permissible. Of course, other factors associated with age may also play an important role in the relationship observed.

Finally, although the framing and the variables in the model attempt to capture aspects of creators and their projects, which may affect the likelihood of creators accusing remixers of plagiarism, aspects of remixers and their remixes almost certainly play an important role in setting up projects for negative feedback by the author of a created project.

The high correlation between qualities of created projects and their remixes makes exploring this comparison difficult in the dataset. By controlling for creators’ gender, the date, creators’ age, and the complexity of the remixed project, it is estimated that remixes by younger users are more likely to result in accusations of plagiarism. Of course, as discussed above, the effect of age on the outcome is difficult to interpret reliably alone. However, even as a tentative result, this model provides support for the argument that accusations of plagiarism are influenced by what each remix consists of, and by who the remixer is, as well as by aspects of the person leaving the feedback. A further attempt to unpack these results occurs in Study 3.

## 6.2.4 Implications for design

In Study 1, it was shown that a technical ability to remix and normative statements in support of remixing do not guarantee either positive reactions or an elimination of charges of plagiarism. Study 2 unpacked the initial results, and evaluated several explanations of the difficulties that designers may encounter when attempting to address these problems.

The findings support the theory that the importance of systems to address charges of plagiarism may be higher in communities where contributions are smaller. Even within Scratch, where every contribution is a stand-alone project, differences in project complexity are associated with large differences in the likelihood that an author will accuse a remixer of plagiarism. Although it is impossible to speak of causal effects to the degree that the results generalize to other communities, the findings imply that encouraging cumulative contribution may not result in a lower rate of plagiarism accusations. Although designers may be encouraged to involve more users in remixing to increase positive attitudes toward remixing, the relationship might be more complex or less tightly associated than some designers might assume. Scratch's example suggests that increased participation in remixing alone may not correspond to a decreased likelihood of plagiarism accusations.

## 6.3 Study 3: Are plagiarism complaints more common when remixes are more similar?

Although the technical and legal ability to remix is constant across the Scratch online community, the nature and content of remixes vary extensively. Some remixes are near or even perfect copies of the project they are based on while others bear little similarity.

Study 2 explored several explanations for the many complaints by focusing on qualities of creators and of the remixed project. Of course, as alluded to in the discussion of Study 2, creators' reactions to remixes are also likely to be influenced by the nature of the remix and the remixer. Perhaps the most obvious remixer-side explanation for the extensive responses to plagiarism in Study 1 is that the extent to which remixers rely on the created project varies. That is, users may not mind remixing when the remix is merely inspired by or loosely based on their work but object when the remixed project is nearly identical to their own. Study 3 makes a first attempt to investigate this hypothesis (H3): *Creators are more likely to accuse remixers of plagiarism when the created project and its remixed project are more similar.*

Because qualities of remixes are highly correlated with qualities of remixed projects, adding remix-level variables to the logistic regression model in Study 2 was untenable with the dataset and methods. Similarly, the available automatic methods of measuring differences between created projects and their remixes were found unreliable. Hand coding is possible but requires viewing and interacting with each pair of projects and is extremely time intensive. As a result, Study 3 represents a first attempt to explore project similarity by offering a bivariate comparison between creator reactions and project similarity using a reduced, nonrepresentative, sample.

### 6.3.1 Procedure

A random selection of 40 creator-remixer project pairs from each of the 6 categories of comments (plagiarism, hinting plagiarism, no comment, and so on; total  $n=240$ ) were put in a random order and given to a new pair of coders, unaware of how these projects were selected, that these projects represented six categories of projects, or that their selection had anything to do with the comments left on these projects. These coders were asked to watch or play each of the projects in each pair and to make a judgment of similarity on a 5-point scale (from 1 = *can't tell they are related* to 5 = *can't tell they are different*). Their responses were highly correlated, ( $r = 0.79, p < 0.001$ ; Cronbach's  $\alpha = 0.88$ ), they rated them within one point of each other in 95% of cases, and these ratings were averaged for a final similarity score for each project pair.

### 6.3.2 Results

A one-way ANOVA was conducted on similarity ratings as a function of the type of comment left. Similarity influenced the type of comment left ( $F_{(5,234)} = 4.78, p < 0.001$ ). Because the specific interest was in assessing whether more similar projects were more likely to lead to plagiarism concerns, planned contrasts were conducted, comparing the similarity scores of the *plagiarism* ( $\mu = 4.40, \sigma = 0.65$ ) group with scores in the other groups — *doesn't fit* ( $\mu = 3.53, \sigma = 0.85$ ), *negative* ( $\mu = 3.93, \sigma = 1.02$ ), *positive* ( $\mu = 3.75, \sigma = 0.85$ ), *hinting plagiarism* ( $\mu = 3.46, \sigma = 1.30$ ) and *no comment* ( $\mu = 3.65, \sigma = 1.14$ ) groups. This analysis revealed that accusations of *plagiarism* were associated with more similar remixes than the *hinting plagiarism* projects ( $t_{234} = 4.22, p < 0.001, d = 0.91$ ), the *doesn't fit* projects ( $t_{234} = 3.94, p < 0.001, d = 1.15$ ), *no comment* projects ( $t_{234} = 3.38, p = 0.001, d = 0.81$ ), *positive* projects ( $t_{234} = 2.93, p = 0.004, d = 0.85$ ), and *negative* projects ( $t_{234} = 2.14, p = 0.033, d = 0.55$ ).

### 6.3.3 Discussion

This study indicated plagiarism accusations were influenced by the similarity between the created project and the remix and these findings give tentative support for H3. When remixes were highly similar to the created projects, they were much more likely to elicit an accusation of plagiarism.

### 6.3.4 Implication for design

Designers of social media remixing systems may be able to decrease charges of plagiarism against remixes by promoting differentiation between created projects and their remixes. In particular, users might react more positively if a system either created technical affordances to create dissimilar remixes or to highlight differences among apparently similar projects.



For example, in Scratch, remixers begin with an unmodified version of the full source of the project to be remixed. An example of technical affordances to facilitate differentiated project might be a remixing interface that begins with a blank project and treats remixed projects as sources of code and media. However, such design affordances may present negative consequences in other areas of the site by increasing the cost to users of making simple improvements or engaging in more direct forms of collaboration. Another suggestion for Scratch may include a “changelog” facility that allows users to explain substantive differences between a remix and an apparently similar created project. For example, a user who fixes a bug or changes a set of sprites could explain initially unnoticeable changes. By emphasizing differences, apparent similarity and charges of plagiarism might both be decreased.

## 6.4 Study 4: Human and Machine Attribution

This study presents an analysis that evaluates the effectiveness of a feature designed to address some of the tensions described in the previous section. More specifically, a feature that consisted of detecting remixes and automatically adding attribution to them. Here I present why feature did not have the intended consequences and what we can learn from it.

This study builds on a dataset used in the previous section and it includes remix-pairs determined by an algorithm using detailed project metadata tracked by the Scratch online community. The dataset is limited in that it does not include projects whose concepts were copied by a user who had seen another’s work but who did not actually copy code, graphics, or sound. Similarly, the dataset contains no measure of the “originality” of projects or an indicator based on ideas that were taken from a source outside Scratch (for example, a user might have created a Pac-Man clone, which would not be considered a remix in the analysis).

The data presented here includes each coded reaction of the author of created projects (named “creators”) on remixes of their projects shared by other users in the site during a twelve-week period after Scratch’s launch, from May 15 through October 28, 2007. Although 2,543 remixes were shared in this period, the analysis is limited to the 932 remixed projects (37% of the total) that had been viewed by the original project’s creator at the time of data collection — a prerequisite for any response. Of these 932 remixes that were viewed by a project creator, 388 creators (42%) left comments on the remixes in question. The remaining were coded as “silence.” Two coders blind to the hypotheses of the study evaluated the comments left by creators of projects being remixed as positive, neutral, or negative. They were also coded as containing accusations of plagiarism or as hinting plagiarism.

Unless it also contained an explicitly negative reaction, an accusation of plagiarism was not coded as “negative.” However, because plagiarism tends to be viewed as negative within Scratch (as suggested by the quotations in the previous section) and more broadly in society (Posner, 2007), accusations of plagiarism (both direct and hinting) were recoded as “negative” except, as was the case in several comments of “hinting plagiarism,” when

these accusations were included with comments that were also coded as positive. Previous published work using this dataset, and later robustness checks, show that the results are substantively unchanged if these explicit charges of plagiarism are excluded from the “negative” category or only the weaker “hinting plagiarism” accusations are excluded.

### 6.4.1 Study 4a: Automatic Attribution

Even before the Scratch website was officially announced, several early adopters became upset at finding remixes of their projects. One of the first complaints occurred on the discussion forums where a 13-year-old asked:

*Is it allowed if someone takes your game, changes the theme, and then calls it “their creation”? Because I created a game called “Pong 2.1” and a week later, a user called “julie” redid the background, and called “her creation” and I am really annoyed with her for taking credit for MY game [...]*

Several people responded showing support for the creator and some, as this 16 year-old, proposed a couple of solutions to the administrators of the site:

*Make it so you can only download a view of how your game/story/animation works. [...] - Or make it so downloadable Scratch files have [read] only protection. [...] - Maybe downloaded Scratch files, after being uploaded, are marked with the creator’s name at the bottom, then any OTHER person who edits it afterward are put on the list.*

Four months after that incident, new functionality was implemented to automatically identify remixes and to point to their created project and its author. About the same time, functionality was added to link to a comprehensive list of remixed works as part of the information of each project from the pages of created projects.

To test the effectiveness of automatic attribution, the effect of the design intervention described in the previous section is considered. The design change took place six weeks after the public launch of the Scratch community and at the precise midpoint in the data collection window. The intervention affected all projects hosted on the Scratch online community including projects shared before the automatic attribution functionality was activated. As a result, the creators’ reactions are classified as occurring outside a technological regime of automatic attribution when a project was both uploaded and viewed by a project’s creator before automatic attribution functionality was activated.

A comparison of the distribution of coded comments among positive, neutral, negative, and silent in the periods before and after the intervention suggests that the introduction of automatic attribution had little effect on the distribution of reaction types (See Figure 6-1). Although the period after the intervention saw more users remaining silent and fewer both positive and negative comments,  $\chi^2$  tests suggest that there is no statistically significant difference in creator reactions between remixes viewed before and after the introduction of automatic attribution ( $\chi^2 = 3.94; df = 3; p = 0.27$ ). As a result, the investigators cannot

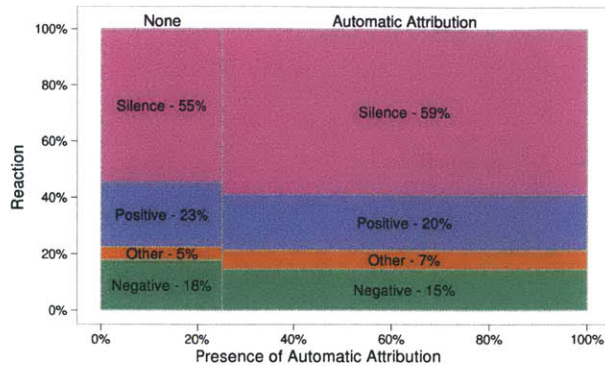


Figure 6-1: Mosaic plot showing the distribution of reactions of creators who had viewed remixes of their project during the six-week periods before and after the introduction of automatic attribution. The proportion of response types is shown along the  $y$ -axis. The proportion of projects viewed with, and without, automatic attribution is shown along the  $x$ -axis. ( $n = 932$ )

conclude that there is any relationship between the presence, or absence, of an automatic attribution system in Scratch and the distribution of various types of reactions.

These results suggest that automatic attribution systems might have limited effectiveness in communities such as Scratch. Of course, the analysis is not without important limitations. For example, the existence of an automatic attribution regime may also affect the behavior of users preparing remixes. A remixer might avoid making perfect copies of projects if they know that their copies will be attributed and are more likely to be discovered.

#### 6.4.2 Study 4b: Manual Crediting

Although the introduction of an automatic attribution feature to Scratch seems to have had a limited effect on creators' responses to remixes of their projects, the presence or absence of credit was a theme in discussions on Scratch online forums — as shown in the quotes in the previous section — and in many coded reactions from the periods both before and after the introduction of automatic attribution. Indeed, in project descriptions or notes from the periods both before and after the change, remixers frequently “manually” gave credit to the creators of their work. Even after remixes were automatically attributed to creators, remixers who did not also give credit manually – essentially producing information redundant to what was already being displayed by the system — were criticized.

For example, after the introduction of automatic attribution functionality, a user left the following comment on a remix of their project:

*Bryan, you need to give me Pumaboy credit for this wonderful game that I mostly pretty much kinda totally made this whole entire game . . . and that you need to give me some credit for it*

For this user, automatic attribution by the system did not represent a sufficient or valid form of credit giving. In the following study, tests are performed for this effect of “manual” credit giving by remixers on coded response types using a method that parallels the analysis in Study 1a and that uses the same dataset.

Manual crediting can happen in several ways. Exploratory coding of 133 randomly selected projects showed that 35 (36%) of each remix pair gave credit. Of these 35 projects, 34 gave credit in the project description field, and 1 project gave credit in a “credits” screen inside the game. As a result, the authors of this study split the sample of projects used in the Study 1a and coded each of the user-created descriptions for the presence or absence of explicit or manual credit giving.

First, to establish that distinct behaviors are examined, the chapter attempts to establish that automatic and manual attributions are not substitutes for each other. As suggested by the qualitative findings and the results in Study 1a, little difference was found in the rate of explicit credit giving among projects created in the presence or absence of automatic attribution. Overall, 276 (about 30%) of the 932 projects in the sample offered explicit credit in the description field of the project. Manual crediting-giving was a widespread practice both before automatic attribution, when 31% of projects in the sample offered explicit credit, and after, when 27% did so. The difference between these two periods was not statistically significant ( $\chi^2 = 1.41$ ;  $df = 1$ ;  $p = 0.24$ ). Previous work studying *Jumpcut*, a video remixing website, supports the idea that automatic and manual credit giving are not interchangeable phenomena. One *Jumpcut* user with permission to create remixed works commented that they, “still feel a moral obligation to people as creators who have a moral right to be attributed (and notified) despite the physical design, which accomplishes this automatically” (Diakopoulos et al., 2007).

Measurements of the effectiveness of manual credit giving used a parallel analysis to Study 1a, with a comparison of the distribution of creator reactions in the presence, and absence, of manual credit giving by remixers. The results show that negative reactions are less common with manual credit but that this difference is minute (from 16% without manual credit to 14% with it). However, the proportion of users who react positively almost doubles with credit giving (from 16% with no crediting to 31% in its presence). A graph of these results is shown in Figure 6-2. Tests show the null hypothesis that these differences in the distribution of reactions are because of random variation ( $\chi^2 = 27.60$ ;  $df = 3$ ;  $p < 0.001$ ), and can be confidently rejected.

Also important to note is a difference in the number of users who are silent after viewing a project (62% lacking manual credit compared with 49% in its presence). This larger proportion of commenting in general might have an important substantive effect on the discourse and behaviors on the site because silent creators might, obviously, have a more limited effect on attitudes toward remixing and user experience than vocal users do. As a robustness check, the study considers the reaction of only creators who left comments ( $n = 388$ ) and found that even with a smaller sample, the result was stronger. In the restricted sample, 41% reacted negatively when they were not given credit. However, only 27% did so when they were credited. Similarly, 42% of users who left comments on projects that did not give credit manually left positive messages. Nearly two-thirds of comments

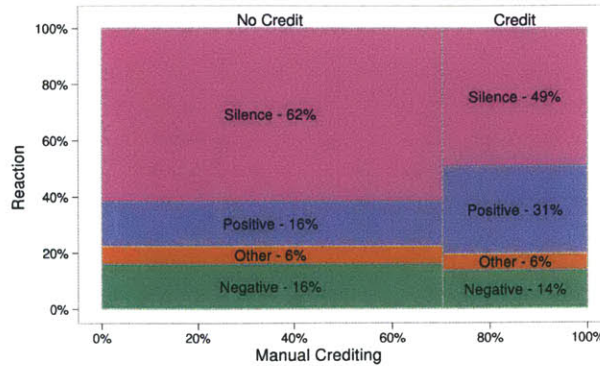


Figure 6-2: Mosaic plot showing the distribution of reactions of creators who had viewed remixes of their project's when manual credit was given.

(61%) were positive when credit was given. These differences within the reduced sample that includes only explicit reactions were also statistically significantly different ( $\chi^2 = 14.09$ ;  $df = 2$ ;  $p < 0.001$ ). Many silent participants are included in the belief that nonresponse is an important type of reaction with real effects on the community. Understanding the reasons for nonresponse and the effect of silence in response to various types of credit giving remains an opportunity for further research.

Both before and after the intervention, manual crediting resulted in more positive comments from the creators of remixed projects. Of course, the results presented here are uncontrolled, bivariate relationships and these results, although provocative, should still be viewed as largely tentative and with caution. As shown in the later qualitative analysis, attribution and credit giving are complex social processes, and there is no claim that the preceding analyses capture it fully.

Especially when credit is given explicitly in the project description, as it was in the project described above, this type of remixing is likely to be welcomed by the creator. For example, user bajooliehaa, posted a project based on Jacque's project, noting the following in the description of the project: "i kinda copied Jacque's 'jetpackcat' game. i used the cat, the scripts (i changed and added some), and the fuel thingy" to which Jacque replied, "I like what you changed about my project."

Other times, however, remixing is poorly received with the consequence of conflicts and animosity. Sometimes, initially negative reaction can change to positive through conversation that facilitates the organic process of humanization and empathy (Benkler, 2009). For example, user robymin29 remixed kooli39's project, something that kooli39 did not appreciate and expressed by creating a response project that consisted of the animation of frowning cat saying:

*Hi i'm kooli39 THE ORIGAL CREATOR MARIO DISCO ROBYMIN29 COPIED ME!! AND DIDN'T EVEN AKNOWLAGE ME HE DIDN'T CHANGE ANYTHING!!I WROTE OR DREW!! AND RAYMAN ...IF YOUR READING THIS THINK ABOUT OTHER PEOPLE!!!!!!*



<b>Name (pseudonym)</b>	<b>Age</b>	<b>Gender</b>	<b>Relationship to Scratch</b>
Nicole	10	F	She has created with hundreds of Scratch projects, primarily animations and art ones.
Kyle	14	M	Casual user of Scratch, interested in math/science simulations and video games.
Amy	15	F	Ardent photographer, has never used Scratch.
Charles	9	M	Active member of a subgroup of Scratch interested in simulation of operating systems.
Ryan	12	F	Longtime member of the Scratch community. Creates complex video games.
Jon	9	M	Casual user of Scratch, collaborates with Scratch friends in person.
Jake	11	M	Casual user, likes making video games.
Cody	16	M	Creates hip-hop accessories, not active in Scratch.
Paul	9	M	Creates Scratch projects with a focus on engineering and video games.
Jimena	17	F	Highly technical teen with programming experience but no experience with Scratch.
Madeline	14	F	Popular animator in the Scratch community.
Susie	10	F	Has created hundreds of projects including games, animations and art, but preferring art.

Table 6.3: Table listing details of interviewees used in Study 2. ( $n = 12$ )

attribution, and credit. No substantive difference was found between the Scratch users and non-users in their answers to questions related to the hypothetical automatic and manual mechanism for attribution.

Before each interview, subjects completed a survey, which elicited demographic information and posed questions about their familiarity with other technologies, and was primarily designed to get a sense of the interviewees' social and technical background. Interviews were structured around a protocol that included a set of nine fictional remixing cases intended to elicit conversations about remixing. The cases were inspired by Sinnreich et al. (2009) theoretical work and from three years of experience moderating the Scratch community. They were designed to present cases where remixing could be controversial but where there is no clear "correct" answer. The goal of the cases was to offer a concrete, and common, set of quandaries to stimulate broad conversations about attitudes toward remixing.

The cases were presented as printed screenshots of various project pages from the Scratch website (anonymized to avoid referring to real cases that users might have seen). The printouts were shown to the interviewees (or discussed over the phone) while explaining each case. All the cases included a remix and its corresponding created project. The cases varied with automatic attribution, manual credit, and the degree of similarity between created project and its remix. For example, the first three cases were:

1. A created project and its remix are identical. The project notes only describe how to play the video game. The remix shows the automatic attribution but no manual credit on the notes.
2. A created project and its remix are different (as shown visually and in project meta-data) but one can clearly see the influence of its created project. The project notes of the remix show manual credit but no automatic attribution. The interviewee was told to imagine the site had a glitch that prevented it from connecting it to its created project.
3. The same set of remix and created projects as in (2) but this time automatic attribution is displayed but manual credit is not.

Each of the interview logs was coded using inductive codes and grounded theory (Charmaz, 2006). The coded responses were analyzed based on categories related to how interviewees answered specific questions about the distinction between automatic attribution and manual credit.

## Results

Confirming the results of Study 4, for users of Scratch, automatic attribution was usually seen as insincere and insufficient. Throughout the interviews, it was found that for most children, getting explicit credit from another person was preferred to attribution given automatically by the system. When asked why, children often responded that knowing that another person had cared enough to give credit was valued more than the computer system would do on its own. The fact that it takes some work, although minimal, to write an



acknowledgment statement, sends a signal of empathy, authenticity, and good intentions (Donath, 2008). Amy articulated this when explaining why she preferred getting credit from another person:

*I would like it even more if the person did it [gave credit] on their own accord, because it would mean that [...] they weren't trying to copy it, pirate it.*

Similarly, Jon explained, “No [the “Based on” is not enough], because he [the remixer] didn't put that, it always says that.” For Jon, automatic attribution is not authentic because it is always there and, as a result, clearly is not coming from the person doing the remix.

Most interviewees seemed to have a clear notion of what they think a moral remix should be. For some, it is about making something different. Jake for example, defines a “good” remix as, “if it has a bunch of differences then it's a good remix. If it has like two, then it's bad.” Besides the differences between the created project and its remix, for some, manual credit is part of what makes it moral. Charles said, “[remixing] is taking somebody else's project and then changing a lot of it and sharing it and giving credit.” Continuing, Charles explained:

*If Green had actually said in the project notes, “This is a remix of Red's project, full credit goes to him,” then I would consider it a remix. However, this [pointing at a remix without manual credit] is definitely a copy.*

Likewise, Ryan mentions that a fictional remix was, “perfectly fine because they gave credit in the project notes.”

Interviewees suggested that manual credit also allows users to be more expressive. For example, Susie explained that expressiveness is the reason that she prefers manual credit through the project notes saying, “I think the manual one is better because you can say ‘thank you’ and things like that. The automatic one just says ‘it's based on.’” Susie also notes that for her, the project notes are a space where a creator can express her wishes about her intellectual property independent of, and even in contradiction to, the license of the projects:

*If I do a project that has music that I really like, I often download the project, take the music. Unless it says in the project notes, “Do not take the music.”*

For Susie and other users of Scratch, the project notes are a space for more than just instructions on how to interact with one's project; they are an expressive space where one can communicate with an audience without having to encumber the creative piece of work with it.

Others point to the fact that people do not pay as much attention to automatic attribution statements as much they do to the manual credit left in project descriptions. Jake, for example, explains that, although he agrees there is some usefulness to having both, project notes still are more important, “because, you know, sometimes people just like skim through a project and you don't see it ‘til the end.” Jake continued to say that creators that do not have both should get a “warning.”

Even though interviewees value manual credit, they still see the usefulness of the automatic mechanism as some sort of community-building prosthetic device — an explanation for the positive reactions to the feature’s introduction. For example, Nicole argues that although manual credit on the notes has more value for her, the automatic attribution is useful as a backup and because it provides a link:

*Well, I think that they should probably write in the notes that – then it should also say “Based on blank’s project,” just in case they forget, and because it gives a link to the created project and it gives a link to the user so you don’t have to search for it.*

A similar explanation was articulated on a comment exchange on one the website’s galleries. A teenage girl who actively participates in Scratch explained the pragmatic value of automatic attribution saying, “the ‘based on’ thingy, it gives a link, and we all luv links, less typing,” before reiterating that manual credit is more valuable:

*At the beginning I thought that you don’t have to give credit when the “based on” thingy is in there, but I realized a lot of people don’t look at that, and I noticed people confused the remix with the created project.*

Creating a Scratch project is a complicated task. A project’s sources can be diverse and the creator can easily forget to acknowledge some, as Paul explains, when asked to choose between a system of manual credit and automatic attribution:

*The thing is, it would be a lot better if they had both. Because, sometimes people probably just forget to do that. And then people would not know.*

Sometimes interviewees recognize what Posner calls the “awkwardness of acknowledgment,” that is, situations where credit is not really needed and can be an unnecessary burden or go against the aesthetics of the work (Posner, 2007). For example, Paul mentioned that sometimes there are projects in Scratch that are remixed so much — as the sample projects that come with Scratch or some “remix chains”<sup>2</sup> — where credit is not necessary:

*There’s this one called “perfect platformer base,” which many people remix. So I don’t think that needs any credit. It’s not actually a real game. It’s all the levels and stuff are just demonstrations.*

Because manual crediting has a higher emotional value, some children mentioned that conflicts over remixing could be addressed by the administrators of the site by editing the project of the remix in question, to enforce credit without transforming it into attribution. Doing so would make it appear that a remixer had credited a created project when they had not. Susie offers a suggestion along these lines when asked about how the administrators of the website should deal with a case of a complaint over a remix that is a parody of someone else’s project. Susie suggested that, “I might remove the project but I might not, you know, maybe I would edit the notes to give credit.” Similarly, Charles described his approach for solving conflicts if he were the administrator of the website suggesting that, “I probably

---

<sup>2</sup>Remix chains typically start with someone sharing a project inviting others to remix (namely, “add your animated avatar to the park.”)

just would stay out of the argument. I probably wouldn't remove it [the remix], I'd just add something in the project notes [like] 'based on Gray's project.'"

This phenomenon of giving less value to technologically simplified social signals is experienced in other social platforms. For example, Amy expressed how on the social network site Facebook, she loves to get comments on her photographs but dislikes those who do not leave comments or opt instead to press the "I like it" button:

*I love when people comment on my pictures. Everybody sees them, because they tell me they have. I'm like, "Oh really? That's great. Why didn't you comment?" I don't like it when people just "like it", because you know they have something to say about it; they just don't. It's like, if they like it, then [they should] take the time to say something.*

Although not designed to be a random sample, these interviews support the proposition that both Scratch participants and other young people share a set of norms about characteristics that determine what a "good" or moral remix is. Among these norms, acknowledging one's sources seems to play a central role. However, participants also seem to share the opinion that this norm is not satisfied through an automated process. They clearly understand the pragmatic value of automated acknowledgment, but they do not see it as a substitute for adherence to the social norm of credit giving. They also see it as void of emotion and expressiveness. For Scratch users, normative constraints are separate from architectural constraints and one cannot replace the other. These findings support and enrich the results from the first study and help provide a better understanding of how Scratch participants, and perhaps children in general, experience authorship norms and automation in online spaces.

## 6.6 Discussion

These studies explore attitudes toward remixing that are believed important. Study 1 shows that users react to remixing in diverse ways. Although every project on Scratch is shared under a license that permits remixing, as many authors of created projects accuse remixers of plagiarism as react positively. Study 2 tested three hypotheses about aspects of remix projects and their creators that might be related to reactions to remixing. Although the analysis cannot offer causal explanations, the findings support the theory that the authors of more complex projects tend to accuse others of plagiarism at a higher rate. On the other hand, no support is found for the hypotheses that authors of created projects that are themselves remixed, or authors who have never published remixes accuse remixers of plagiarism at a higher rate. Study 3 presents tentative findings that support the explanation that users are more likely to make accusations of plagiarism when projects are more similar.

In this analysis, several crucial assumptions are made. In general, the framing tends to treat charges of plagiarism as negative and to be avoided. This interpretation is roughly supported in the dataset: 32% of comments coded as negative were also coded as explicitly calling out plagiarism while only 2% of positive comments did so. Of course, this does not mean that charges of "copycat" are necessarily associated with either bad feelings by

users or, more important, behaviors that social media designers find problematic. Although the understanding of the coded comments and the experience with the community gives confidence in the framing, further work should unpack these assumptions.

The results from Study 1a called in to the question the effectiveness of automatic attribution functionality in encouraging more positive user reactions in Scratch. Later, building on these results in Study 1b suggests that manual crediting may do the work that Scratch's designers had hoped automatic attribution would. Results from the analysis of user interviews presented in Study 2 help answer the question of "why?" and suggest that users find manual credit authentic and more meaningful to users because it takes more time and effort. Usually, UI improvements are designed to help reduce the time and effort involved in using a system. However, in trying to help users by attributing automatically, Scratch's designers misunderstood the way that attribution as a social mechanism worked for Scratch's users. The fundamental insight is that although both attribution and credit may be important, they are distinct concepts and that credit is, socially, worth more. A system can *attribute* the work of a user but *credit*, seen as much more important by users and as having a greater effect on user behavior, cannot be done automatically. Computers can attribute. Crediting, however, takes a human.

As suggested at the beginning of this chapter, this fundamental result leads to two distinct contributions. First, and more specifically, the analysis offers an improved understanding of the way that attribution and credit works in user-generated content communities over what has been available in previous work. The two studies suggest that scholars are correct to argue that credit plays an important role in social media communities and offer empirical confirmation for the important role that authenticity plays in how users conceptualize credit. The in-depth interviews explain some reasons for this being so. Second, through the evaluation of an unsuccessful technological design, the work offers a broader, if more preliminary, contribution in suggesting an important limit of designers' ability to support community norms in social media systems. As the literature on design and social media grows, the importance of good support for communities with healthy norms promoting positive interactions is likely to increase. In attempting to design for these norms, it is suspected that researchers will increasingly encounter similar challenges.

It is argued that designers should approach interventions iteratively. This design approach can be understood through the theoretical lens of the social construction of technology (Pinch and Bijker, 1984): designers cannot control technological outcomes, which must be built through a close relationship between designers and users. Designers must move away from seeing their profession as providing solutions. They must channel users, work closely with them, and iterate together, to negotiate and achieve a set of shared goals.

The prevalence of user-generated content sites stresses the importance of how online social spaces should deal with issues of attribution and the results are likely to be immediately relevant to designers. For example, the Semantic Clipboard is a tool built as a system of automatic attribution for content reuse (Seneviratne et al., 2009). Developed by researchers who found a high degree of Creative Commons license violations around the reuse of Flickr images, the tool is a Firefox plugin that provides, "license awareness of web media," and enables people to automatically, "copy [media] along with the appropriate license metadata."

The results suggest one way that this approach may fall short.

However, automatic attribution is not the only way that technologists can design to acknowledge others' contributions. Indeed, the results suggest that there may be gains from design changes, which encourage credit giving without simply automating attribution. For example, Scratch's designers might present users with a metadata field that prompts users to credit others and suggests creators whose work the system has determined might have played a role. This affordance might remind users to credit others, and might increase the amount of crediting, while maintaining a human role in the process. The research has suggested that extra effort instills manual credit giving with its value, which suggests that in other social media communities similar affordances, which help prompt or remind users to do things that a system might do automatically, represent a class of increasingly important design patterns and a template for successful design interventions in support of community norms.

Indeed, promising future work might use attitudes toward and responses to remixing as an independent variable. For example, designers of remixing communities may want to look at the effect reactions to remixing have on the rate or nature of contributions. It seems unlikely that a community hostile toward remixing or actively involved in calling one another "copycats" would be a solid foundation on which to build such a culture. Future work will be able to build on these findings to establish how these attitudes help frame a social environment. Similarly, such work should look at the effect of positive reactions. Although Study 2 focused on charges of plagiarism, positive responses seem as likely to affect remixing rates as negative reactions and accusations of plagiarism. Future work should build on the work carried out here to do so.

## Chapter 7

# Conclusions

I became interested in remixing because of the controversies surrounding intellectual property. I was particularly motivated by the cause that Lessig, Benkler, Jenkins, and other scholars had inspired: the stopping of what is seen as an attack on amateur creativity by corporate interests. My resolve strengthened after receiving “cease and desist” notices from lawyers representing companies who felt that some of the projects in the Scratch website were in violation of their intellectual property rights. For example, we received an official DMCA<sup>1</sup> “take down notice” from the owners of the popular video game Pac-Man, demanding the removal of a Pac-Man project created by a young community member (see Figure 7-1)—an inspirational remix using the terminology from chapter 4. The letter included a commentary on how young programmers should also learn to respect the intellectual property of others:

*While we appreciate the educational nature of your enterprise and look forward to the contributions of the future programmers you are training, part of their education should include concern for the intellectual property of others.*

Although I still strongly believe the current copyright system is broken, I found remixing to be a much more nuanced phenomenon. I was surprised how often young creators—at least the ones on the Scratch Online Community and a few other similar websites—did not universally favor remixing when it came to their creations being reused by others. Many of them asked us to give them the ability to “lock” their projects to prevent others from remixing them or even downloading them. At the same time, these young members were perfectly content, and rightly so, I believe, to remix video games like Nintendo’s Mario Bros. or Namco’s Pac-Man, or to grab images from search engine results to produce creative programmable media with them.

---

<sup>1</sup>Digital Millennium Copyright Act

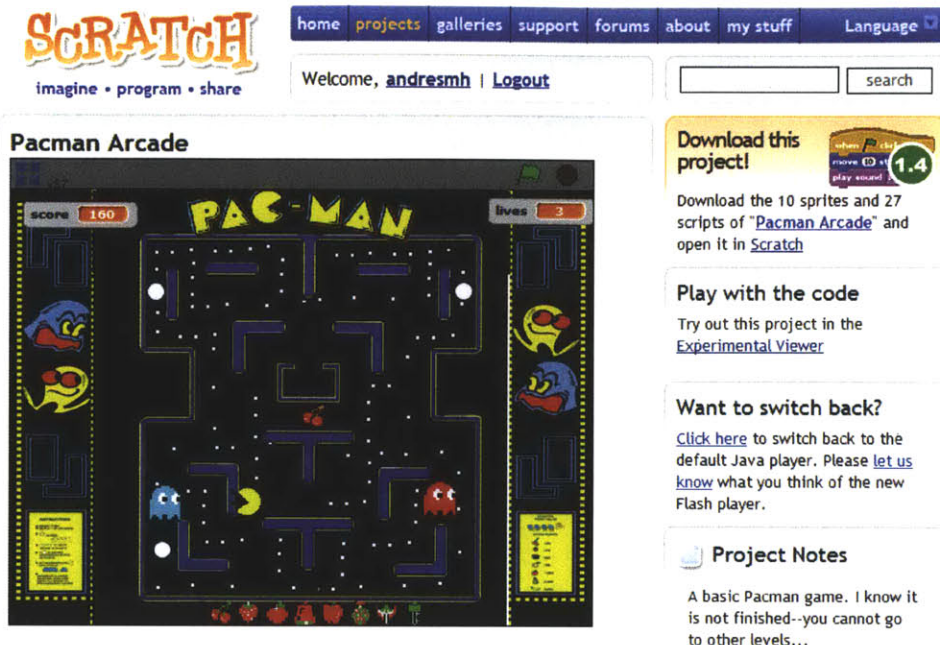


Figure 7-1: Pac-Man remix removed after to DMCA take down notice.

## 7.1 Summary and Contributions

In chapter 4, I argue that the reason why amateur creators in Scratch exhibit seemingly conflicting attitudes is because remixing represents a several different types of *processes*. To tease this apart, I developed a *remixing taxonomy* along two dimensions: *originality and generativity*. Originality helps us understand the effort involved and how different a remix is from its source project. For example, it allows us to distinguish perfect copies from modular reuse or merely inspirational remixes. Generativity, on the other hand, helps us assess how prolific a particular source project is, or intends to be. For example, it helps us distinguish between those ad hoc remixes among two individuals, those happening in a small collaborative group, and those part of a crowd.

While some people want to “lock” their projects, others are actively trying to encourage people to remix their work or to engage in remixing as part of collaborative groups. In chapter 5, I present the conditions for remixing, more specifically the *system and content attributes* that are associated with different degrees of generativity and originality. The core finding here is that there is a paradoxical inverse relationship between generativity and originality: the attributes associated with an increase in one-author status, complexity, and cumulativeness are also associated with a decrease in the other.

As I mentioned before, remixing emerged as one of the main tensions in the community. At the same time, remixing was mechanism for interaction and collaboration. In chapter 6, I examine the attitudes young people have toward remixing. Perfect copies tend to receive more complaints than those that are highly generative. On the other hand, those people who

are very positive toward remixing often used it as part of small or large scale collaborative practices. Furthermore, the presence or absence of manual credit also determines how people react to remixing, regardless of the presence of automatic attribution.

In this work, I have presented the design and study of a remixing system. Also, I use this system to investigate the process, conditions, and attitudes toward remixing. The main *contributions* of this work are:

- The implementation of a scalable social computing system, ScratchR, that enables people to share and remix programmable media. Released under a GPL<sup>2</sup> v2 license, ScratchR has been reused a number of times. For example, the Portuguese telecommunications company Sapó created an instance of the website for their local market (see Figure 7-2).
- The development of a large and international online community of more than one million amateur creators. The website expanded Scratch from being “just” a creation tool to becoming a place where people collaborate, interact, and hang out (see Figure 7-3 for an illustration of how a young member sees the community).
- The collection of a large corpus of research data that includes more than two million interactive media projects, and activity logs of more than a million accounts.
- A new theoretical framework to understand remixing based on a set of mixed methods studies. This framework provides a remixing taxonomy based on two dimensions: originality and generativity, and evidence to suggest that those two dimensions are at odds with one another. Then I expand our understandings of how young people perceive remixing, finding evidence for the need for manual credit regardless of automatic attribution.

## 7.2 Design Implications

Throughout this work, I present different implications for system design. In chapter 3, I explain that part of the success of the Scratch website is that it provides opportunities for “creative socialization.” I argue for the value of using sociability to support creative engagement through basic mechanisms like remixing, which often facilitate more complex collaborative practices such as group work. Additionally, I argue for “participatory diversity,” that is the support for both “making” and “listening” or lurking. This allows people to engage and disengage as they see fit, decreasing the chances of turning people away because they might feel forced to participate in a specific way. That said, although multiple forms of engagement are supported, the aesthetic design of the Scratch website emphasized the content generated by the community rather than any kind of sophisticated interface elements on the website itself.

In terms of scaling community moderation, I present a hybrid model that relies on input from the community and administrators’ decisions. Part of the reason this works is because

---

<sup>2</sup>General Public License





Figure 7-2: Instance of ScratchR for Portugal.



Figure 7-3: Scratch Online Community as seen by a 14-year-old community member.

all the communication channels are public, preventing conflicts emerging in hidden places. That said, dealing with off-site communications continues to be a challenge.

At the end of chapter 3, I argue for the value of designer engagement in the community, first in the form of bootstrapping and then as a continued practice. Bootstrapping a community works well through in-person gatherings, such as workshops, that help identify potential issues with the system and, more importantly, seed the community with the kind of content that is desired for the system. Continuous and authentic engagement in the community helps designers maintain a mental model of the different types of user that ultimately help inform the system design and policies for the community.

Using the findings from chapter 4, the take away is that supporting remixing actually means supporting a diverse range of practices. System designers might need to emphasize the most desirable forms of remixing, but acknowledge that others will emerge. For example, this could be implemented by detecting and celebrating on the front page of the website any type of remixing that is in particular need of recognition and encouragement. The emphasis on specific types of remixes could be automated and manually curated. The former scales but is likely to be gamed (as any other metric that increases attention), the later might give more control over what is displayed but with a higher cost in terms of time.

The implications from chapter 5 are two-fold. First, remixing systems need to be modular, open, and support mechanisms for attribution. These three system attributes, however, come with a cost in terms of system complexity which can be addressed given the right user interface. Second, remixing systems need to decide whether quantity (i.e., generativity) or quality (i.e., originality) of remixing is desired. This decision can change, but the levers needed to incentivize one or the other are the same: author status, content com-

plexity, and cumulative provenance. For example, when starting a community one might want to favor quantity over quality. Later on, system designers might want to promote more original remixing by highlighting works of medium complexity and by less popular contributors.

Finally, chapter 6 provides a warning about two reasons why conflict might emerge as a result of remixing: lack of originality, and manual credit. System designers might address this by reducing the visibility of exact copies, or like Twitter’s development of a retweet button: highlighting content re-use while still letting people remix as a form of spreading a particular piece of content. Also, system designers might want to create the mechanisms for encouraging contributors to acknowledge their sources. Often people do not know acknowledge because they forget where they got the source materials, so automation can still be useful to identify these scenarios and remind people of the value of giving credit as a form of prosociality.

### 7.3 Future Research

Through this work, I hope to have opened the path to future empirical studies on remixing in Scratch and beyond. In chapter 3 I present an overview of five years in the history of the Scratch Online Community. This overview, however, does not undertake a detailed analysis of the different types of members that have participated in the community by remixing. Nor does it investigate the people who are often at odds with one another in their perspectives on remixing. For example, generally speaking, *visual artists* on Scratch tend to be more protective of their work than *programmers*. A close study of these individual differences might help understand how different forms of creative participation might impact collaboration.

As mentioned before, amateur creators in Scratch have different attitudes toward remixing due to issues of originality and attribution. However, this model of user behavior can be extended. Future research should analyze two types of motivations for remixing: *relational* and *functional*. Relational remixing serves as a way to *connect with others*, while functional remixing plays a purely pragmatic role, is a mechanism to “get the job done.” For example, by participating in remixing chains, such as coloring contests or “add your character to the party,” people got to be part of a small movement. This type of remixing often led to reciprocity networks where people remix one another as form of socialization. On the other hand, remixing the code to create a “scrolling background” game like I mentioned in chapter 3, permits reuse without necessarily trying to connect with others.

Generally speaking, I noticed that when people engage in remixing for relational reasons, it tends to be in the context of aesthetic creations frequently—though not always—involving people who self-identify as artists. On the other hand, people remixing for functional reasons tend to involve both the reuse of code and media; however, remixing code—often but not always by people who self-identify as programmers—tends to be less contentious than remixing art. This poses an area for future research: to investigate the extent of the validity of this stereotype that artists are less amenable to remixing than coders.

Similarly, in the chapter about the conditions for remixing (chapter 5), I presented a rhetorical framework to understand the system attributes conducive to remixing. However, more work is needed to operationalize and find quantitative metrics for assessing the impact of each of those traits. For example, experimenting with different levels of openness, modularity, and attribution would help clarify the degree to which each of those characteristics impacts remixing behavior. Also, a comparative analysis of the role of remixing in each of the collaborative groups or companies would illuminate how to better design future online communities for creative collaboration.

Additionally, one of the richest areas for future work on remixing is an in-depth analysis of the implications of remixing for learning. For example, a worthwhile study could investigate how to best leverage remixing as scaffolding in people’s learning of programming, and how to help young people understand when one needs to go beyond “copying and pasting.”

Last, one of the main concerns common to studies of peer production is generalizability. Though I cannot speak for the generalizability of these results to other remixing communities or peer production projects, I believe that studying remixing in Scratch gives good insight into the behavior of young creators. How much these results will generalize to adults, to other communities, or to activities beyond the creation of animations and games, remain largely open questions for future research.

## 7.4 Epilogue: MusicalMoon

Four years after the creation of “Mesh Inc.” and “Green Bear Group” I had the opportunity to chat with MusicalMoon —one of the most active members of those two groups.

MusicalMoon was able to articulate in detail the reasons why she thought both groups did not accomplish as much as she would have expected:

*There are several reasons. . . . one of them is that I couldn’t organize the development of projects well. We got into the details right away. Without first creating a general idea of a sort. Because of that, everybody had their own idea about the project inside their head, and we had a lot of random, completely different suggestions. In the end I just took control and decided to do everything like I saw it. :P Also, I couldn’t spread the work between people very well. While our artists had something to do, our programmers [sic] had to sit and do nothing. Or vice-versa. My goal with Mesh Inc was a bit too ambitious for my knowledge at that time, if we went with more simple projects like most of the companies, we could have done moderately well, I think.”*

As we talked about her plans for the future — such as her goals for college, which she will attend in a year or so — she also described how her experience with Scratch influenced her interest in studying topics in “engineering and social studies”:

**Andrés:** *what year of school are you now?*

**MusicalMoon:** *10th grade or 11th grade in the US*

**MusicalMoon:** Two more years to go : )

**Andrés:** *wow! and then college?*

**MusicalMoon:** *Yeah*

**Andrés:** *do you have an idea of what you would like to study in college?*

**MusicalMoon:** *I've had a lot of stuff in mind...I personally think social engineering would be good but I have a feeling that is a bit of an undeveloped field of study right now.*

**Andrés:** *what do you mean by social engineering?*

**MusicalMoon:** I got the idea from a russian guy whose lectures I've seen. He studies the mechanics of progress and development of projects and works to optimise it. It's called systems engineering, not social engineering : P The guy's name is Anatoly Levenchuk. What do you think?

**Andrés:** *it's super interesting!*

**MusicalMoon:** *I thought so as well... I had a pretty long search before stopping on this one. I actually started from wanting to be an economist and a sociologist later. This looks like it involves a knowledge of engineering and social studies.*

**Andrés:** *do you think the Internet influenced your interests?*

**MusicalMoon:** *Yes, very much. Scratch, too.*

**Andrés:** *how?*

**MusicalMoon:** *Well, I gained an interest in organizations when I had to manage Mesh Inc. Mesh Inc sort of fell apart. I tried to find out what was wrong with it and how it could have been prevented. I think I've gained a lot of knowledge about it since then.*

## Appendix A

# Entity Relationship Diagrams

The ScratchR database has grown in complexity, but at its core remains the same as when the project started. In this section I present the full list of MySQL tables along with their fields, indices, and relationships. The diagrams presented from the perspective of three different entities: project (see Figure A-2), user (see Figure A-1), and gallery (see Figure A-3).

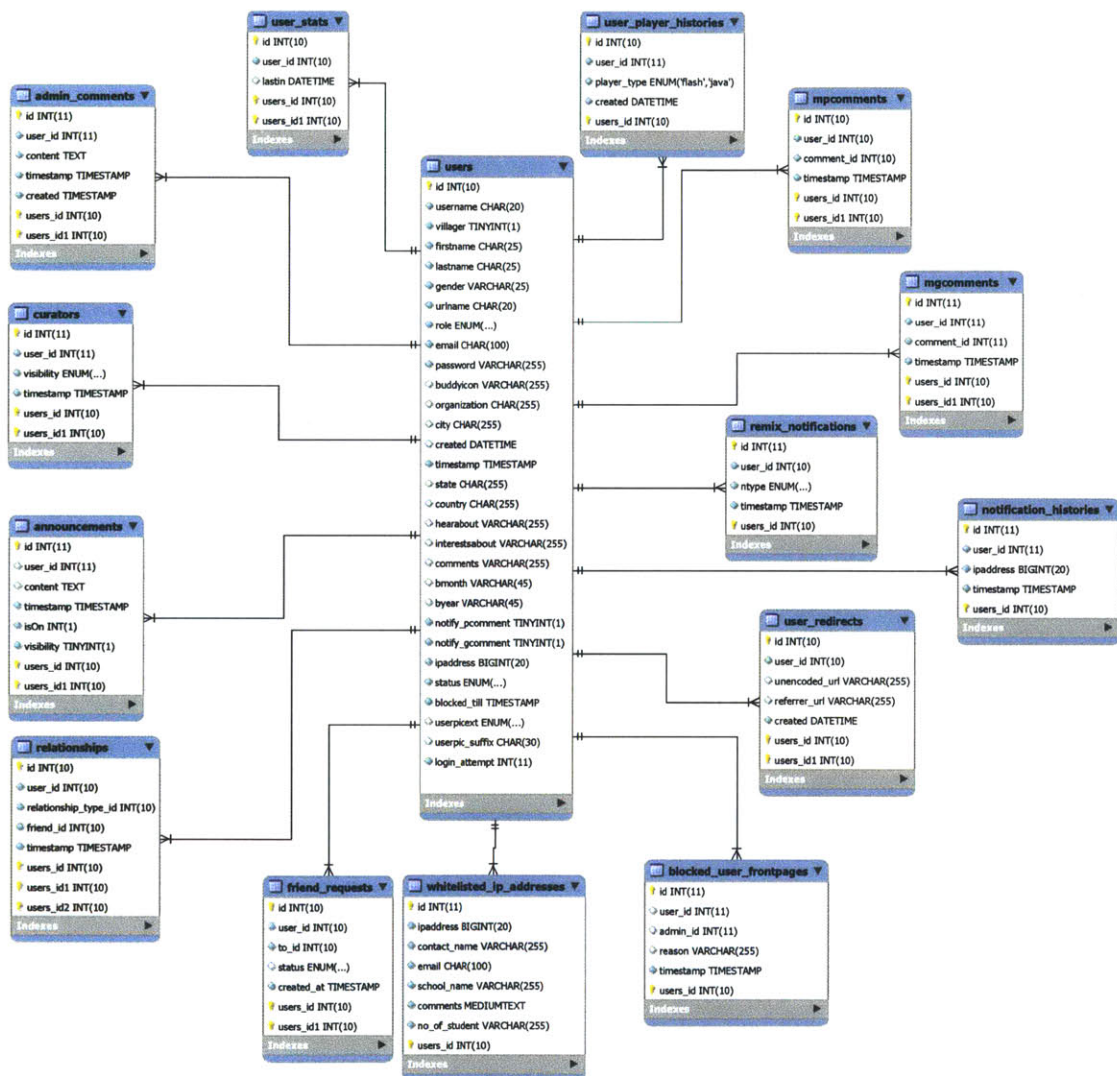


Figure A-1: User tables

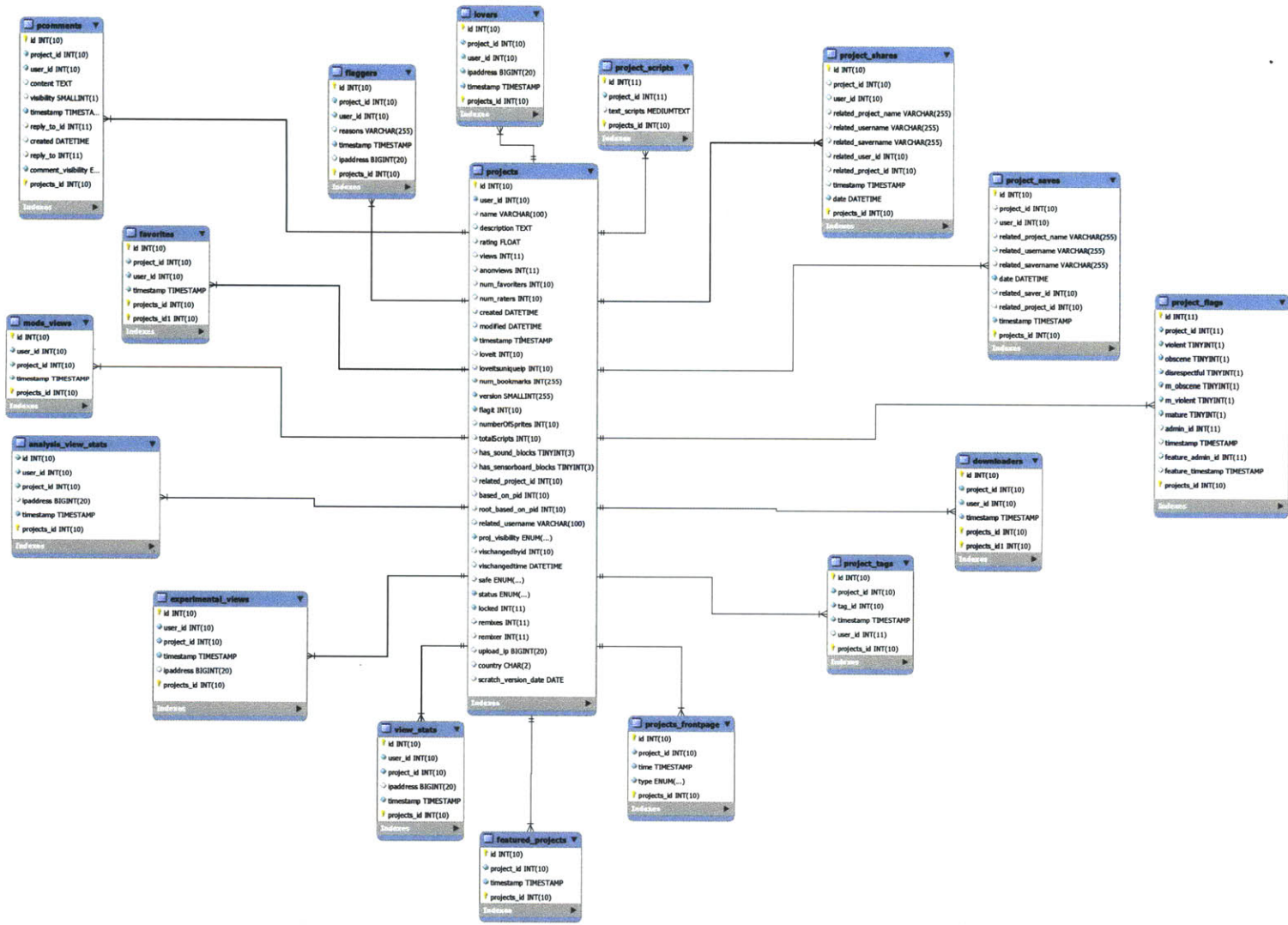
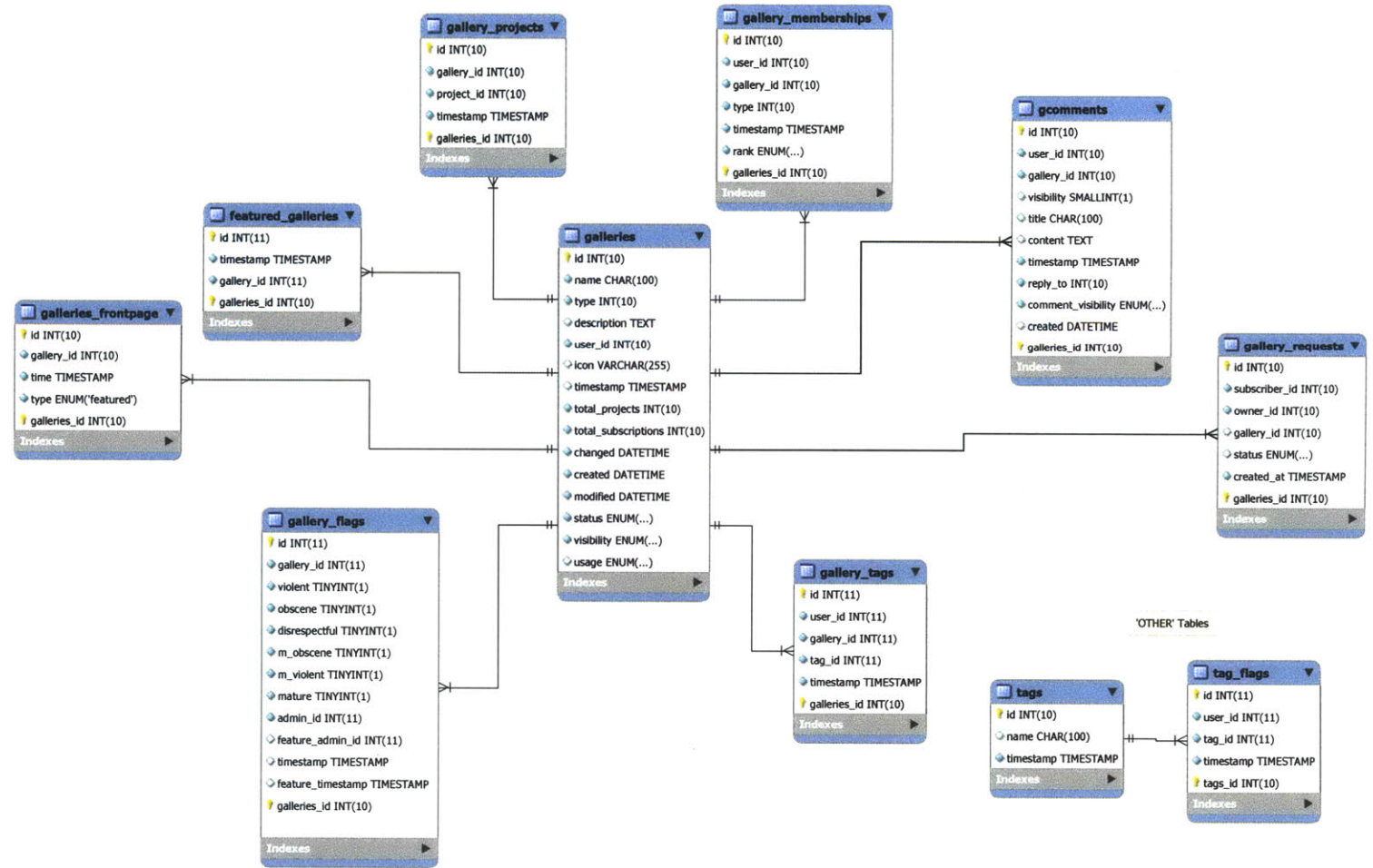


Figure A-2: Project tables



Figure A-3: Project tables



## Appendix B

# Project Attribute Tables

The following tables present the aggregate values of the attributes of 2,426,894 projects shared in the first five years of activity on the Scratch Online Community.

Table B.1 and Table B.2 show the attributes for all 2,426,894 projects. Table B.3, Table B.4, Table B.5 show the attributes for those projects that are remixes. Table B.6, Table B.7, and Table B.8 show the attributes only for those *de novo* projects (non-remixes). Table B.9, Table B.10, and Table B.11 show the attributes for remixes whose source project is created by someone other than the creator of the remix. Table B.12, Table B.13, Table B.14 show the remixes that function as version control since the creator of the remix is the same as the creator of the source project.

Table B.1: All Projects: *Male*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	1,573,607	32.481	11	223.8	1
views	1,573,607	17.797	6	103.417	1
anonviews	1,573,607	1.127	0	8.6	0
favoriters	1,573,607	0.52	0	6.14	0
loveits	1,573,607	0.763	0	8.35	0
versions	1,573,607	1.267	1	1.238	1
flags	22,117	1.36	1	1.384	1
sprites	1,573,607	6.762	3	16.258	1
scripts	1,573,607	21.15	6	359.227	1
is_visible	1,573,607	0.8	1	0.4	1
remixes	1,573,607	0.376	0	10.748	0
remixers	1,573,607	0.273	0	7.32	0
downloads	1,573,607	2.197	0	24.737	0
downloaders	1,573,607	2.172	0	24.15	0
comments	1,573,607	2.659	0	17.51	0
commenters	1,573,607	1.633	0	9.823	0
galleries	1,573,607	0.463	0	2.438	0
blocks	1,573,607	146.475	37	645.532	0
block_types	1,573,607	14.722	11	12.435	0
costumes	1,573,607	22.687	8	83.05	3
sounds	1,573,607	4.359	2	14.622	1
ugstrings	1,573,607	29.334	3	562.869	0
saves	1,573,607	7.766	1	23.766	0
seconds_to_share(DAY)	1,573,607	42.127	0.002	341.155	0
seconds_to_remix(DAY)	243,776	36.251	0.27	127.906	0
user age(years)	1,566,297	16.588	12	15.562	11
account age(days)	1,571,326	140.64	44	221.651	0

Table B.2: All Projects: *Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	853,287	28.239	10	115.365	1
views	853,287	13.683	5	56.447	1
anonviews	853,287	0.881	0	5.156	0
favoriters	853,287	0.419	0	3.364	0
loveits	853,287	0.605	0	4.426	0
versions	853,287	1.185	1	0.872	1
flags	7402	1.299	1	1.115	1
sprites	853,287	4.641	2	9.742	1
scripts	853,287	10.655	3	59.154	1
is_visible	853,287	0.786	1	0.41	1
remixes	853,287	0.317	0	7.936	0
remixers	853,287	0.252	0	5.148	0
downloads	853,287	1.169	0	10.393	0
downloaders	853,287	1.161	0	10.23	0
comments	853,287	3.4	0	14.895	0
commenters	853,287	1.782	0	6.957	0
galleries	853,287	0.327	0	1.217	0
blocks	853,287	63.188	17	288.595	0
block_types	853,287	9.393	7	8.596	0
costumes	853,287	13.15	6	40.901	3
sounds	853,287	3.072	2	7.004	1
ugstrings	853,287	18.22	0	519.424	0
saves	853,287	4.74	1	14.835	0
seconds_to_share(DAY)	853,287	35.114	0	371.464	0
seconds_to_remix(DAY)	116,901	23.852	0.086	98.216	0
user age(years)	850,165	17.171	12	14.503	11
account age(days)	853,136	137.877	43	215.834	0

Table B.3: Remixes: *Male and Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	670,932	32.949	13	195.477	1
views	670,932	17.891	7	93.258	1
anonviews	670,932	1.075	0	8.332	0
favoriters	670,932	0.518	0	5.479	0
loveits	670,932	0.731	0	7.128	0
versions	670,932	1.268	1	1.401	1
flags	9456	1.326	1	1.262	1
sprites	670,932	8.766	4	16.105	1
scripts	670,932	28.077	8	220.301	1
is_visible	670,932	0.756	1	0.43	1
remixes	670,932	0.312	0	6.008	0
remixers	670,932	0.254	0	3.921	0
downloads	670,932	2.188	0	22.698	0
downloaders	670,932	2.168	0	22.129	0
comments	670,932	2.702	0	15.864	0
commenters	670,932	1.621	0	8.743	0
galleries	670,932	0.452	0	2.248	0
blocks	670,932	206.894	53	604.502	0
block_types	670,932	17.258	13	14.118	6
costumes	670,932	30.791	11	93.734	3
sounds	670,932	5.353	2	12.363	1
ugstrings	670,932	50.006	4	785.406	0
saves	670,932	12.857	3	32.184	0
seconds_to_share(DAY)	670,932	94.077	0.031	391.484	0
seconds_to_remix(DAY)	115,802	32.386	0.192	116.813	0
user age(years)	668,401	15.996	12	14.871	11
account age(days)	670,494	147.587	57	215.896	0

Table B.4: Remixes: *Male*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	456,714	35.641	13	226.341	1
views	456,714	19.703	8	107.007	1
anonviews	456,714	1.16	0	9.627	0
favoriters	456,714	0.59	0	6.363	0
loveits	456,714	0.827	0	8.304	0
versions	456,714	1.302	1	1.512	1
flags	7179	1.321	1	1.297	1
sprites	456,714	9.884	5	17.702	1
scripts	456,714	33.463	11	261.288	1
is_visible	456,714	0.762	1	0.426	1
remixes	456,714	0.327	0	7.176	0
remixers	456,714	0.261	0	4.637	0
downloads	456,714	2.625	0	26.496	0
downloaders	456,714	2.6	0	25.81	0
comments	456,714	2.654	0	17.664	0
commenters	456,714	1.654	0	9.906	0
galleries	456,714	0.504	0	2.587	0
blocks	456,714	254.876	72	692.117	0
block_types	456,714	19.773	17	14.784	6
costumes	456,714	35.831	13	106.428	3
sounds	456,714	6.073	2	13.477	1
ugstrings	456,714	51.567	6	692.094	0
saves	456,714	15.006	4	35.85	0
seconds_to_share(DAY)	456,714	100.652	0.071	401.749	0
seconds_to_remix(DAY)	79,483	36.253	0.209	125.6	0
user age(years)	454,892	16.166	12	15.715	11
account age(days)	456,296	148.031	56	218.329	0

Table B.5: Remixes: *Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	214,218	27.208	13	102.015	1
views	214,218	14.027	7	52.956	1
anonviews	214,218	0.893	0	4.449	0
favoriters	214,218	0.365	0	2.768	0
loveits	214,218	0.526	0	3.471	0
versions	214,218	1.197	1	1.126	1
flags	2277	1.344	1	1.148	1
sprites	214,218	6.382	3	11.658	1
scripts	214,218	16.595	4	79.096	1
is_visible	214,218	0.742	1	0.438	1
remixes	214,218	0.278	0	1.81	0
remixers	214,218	0.239	0	1.525	0
downloads	214,218	1.254	0	10.753	0
downloaders	214,218	1.247	0	10.591	0
comments	214,218	2.804	0	11.089	0
commenters	214,218	1.549	0	5.494	0
galleries	214,218	0.343	0	1.242	0
blocks	214,218	104.596	25	328.404	0
block_types	214,218	11.896	8	10.77	6
costumes	214,218	20.045	8	56.564	2
sounds	214,218	3.819	2	9.38	1
ugstrings	214,218	46.679	0	954.353	0
saves	214,218	8.276	2	21.757	0
seconds_to_share(DAY)	214,218	80.058	0.004	368.256	0
seconds_to_remix(DAY)	36,319	23.923	0.168	94.234	0
user age(years)	213,509	15.635	12	12.88	10
account age(days)	214,198	146.641	59	210.617	0

Table B.6: De Novo projects: *Male and Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	1,755,962	30.241	10	191.718	1
views	1,755,962	15.762	5	88.395	1
anonviews	1,755,962	1.028	0	7.259	0
favoriters	1,755,962	0.472	0	5.274	0
loveits	1,755,962	0.698	0	7.252	0
versions	1,755,962	1.227	1	0.997	1
flags	20,063	1.354	1	1.349	1
sprites	1,755,962	4.966	2	13.465	1
scripts	1,755,962	13.404	4	314.284	1
is_visible	1,755,962	0.81	1	0.392	1
remixes	1,755,962	0.372	0	10.97	0
remixers	1,755,962	0.271	0	7.418	0
downloads	1,755,962	1.701	0	20.107	0
downloaders	1,755,962	1.682	0	19.664	0
comments	1,755,962	3.003	0	16.928	0
commenters	1,755,962	1.71	0	8.988	0
galleries	1,755,962	0.401	0	2.03	0
blocks	1,755,962	82.918	23	521.749	0
block_types	1,755,962	11.163	8	9.848	0
costumes	1,755,962	14.957	6	59.968	3
sounds	1,755,962	3.354	2	12.508	1
ugstrings	1,755,962	16.035	1	423.144	0
saves	1,755,962	4.35	1	14.148	0
seconds_to_share(DAY)	1,755,962	18.87	0	333.535	0
seconds_to_remix(DAY)	244,875	32.159	0.173	120.367	0
user age(years)	1,748,061	17.098	12	15.314	11
account age(days)	1,753,968	136.64	39	220.959	0



Table B.7: De Novo projects: *Male*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	1,116,893	31.189	10	222.739	1
views	1,116,893	17.017	6	101.902	1
anonviews	1,116,893	1.114	0	8.143	0
favoriters	1,116,893	0.492	0	6.046	0
loveits	1,116,893	0.736	0	8.368	0
versions	1,116,893	1.253	1	1.106	1
flags	14,938	1.379	1	1.423	1
sprites	1,116,893	5.485	3	15.448	1
scripts	1,116,893	16.115	5	392.183	1
is_visible	1,116,893	0.816	1	0.388	1
remixes	1,116,893	0.396	0	11.904	0
remixers	1,116,893	0.279	0	8.168	0
downloads	1,116,893	2.021	0	23.979	0
downloaders	1,116,893	1.997	0	23.435	0
comments	1,116,893	2.661	0	17.447	0
commenters	1,116,893	1.625	0	9.789	0
galleries	1,116,893	0.447	0	2.374	0
blocks	1,116,893	102.149	29	620.049	0
block_types	1,116,893	12.656	10	10.666	0
costumes	1,116,893	17.313	7	70.614	3
sounds	1,116,893	3.658	2	15.009	1
ugstrings	1,116,893	20.243	2	500.223	0
saves	1,116,893	4.805	1	15.492	0
seconds_to_share(DAY)	1,116,893	18.195	0.001	309.849	0
seconds_to_remix(DAY)	164,293	36.249	0.311	129.007	0
user age(years)	1,111,405	16.76	12	15.496	11
account age(days)	1,115,030	137.615	39	222.925	0

Table B.8: De Novo projects: *Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	639,069	28.584	9	119.505	0
views	639,069	13.567	5	57.569	1
anonviews	639,069	0.877	0	5.372	0
favoriters	639,069	0.437	0	3.541	0
loveits	639,069	0.632	0	4.703	0
versions	639,069	1.181	1	0.769	1
flags	5125	1.278	1	1.1	1
sprites	639,069	4.057	2	8.934	1
scripts	639,069	8.664	3	50.589	1
is_visible	639,069	0.801	1	0.4	1
remixes	639,069	0.329	0	9.11	0
remixers	639,069	0.257	0	5.883	0
downloads	639,069	1.14	0	10.269	0
downloaders	639,069	1.132	0	10.106	0
comments	639,069	3.6	0	15.964	0
commenters	639,069	1.861	0	7.381	0
galleries	639,069	0.321	0	1.209	0
blocks	639,069	49.307	16	272.555	0
block_types	639,069	8.555	7	7.548	0
costumes	639,069	10.839	5	33.763	3
sounds	639,069	2.822	2	5.98	1
ugstrings	639,069	8.681	0	233.622	0
saves	639,069	3.555	1	11.383	0
seconds_to_share(DAY)	639,069	20.048	0	371.318	0
seconds_to_remix(DAY)	80,582	23.82	0.064	99.96	0
user age(years)	636,656	17.686	12	14.973	11
account age(days)	638,938	134.939	38	217.475	0

Table B.9: Only Collaborative Remixes: *Male and Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	422,099	31.638	16	147.343	1
views	422,099	17.362	9	73.701	1
anonviews	422,099	1.017	0	7.88	0
favoriters	422,099	0.445	0	3.923	0
loveits	422,099	0.616	0	5.347	0
versions	422,099	1.184	1	0.928	1
flags	6896	1.286	1	1.262	1
sprites	422,099	8.856	4	14.768	1
scripts	422,099	26.117	8	193.196	2
is_visible	422,099	0.78	1	0.414	1
remixes	422,099	0.35	0	7.384	0
remixers	422,099	0.298	0	4.807	0
downloads	422,099	1.908	0	17.114	0
downloaders	422,099	1.896	0	16.787	0
comments	422,099	2.614	1	12.629	0
commenters	422,099	1.56	1	6.738	0
galleries	422,099	0.427	0	1.716	0
blocks	422,099	204.133	50	582.972	0
block_types	422,099	17.262	13	14.365	6
costumes	422,099	33.846	12	95.219	3
sounds	422,099	5.605	2	12.423	1
ugstrings	422,099	51.925	3	817.661	0
saves	422,099	13.681	3	32.254	0
seconds_to_share(DAY)	422,099	139.929	0.689	444.081	0
seconds_to_remix(DAY)	81,486	33.821	0.703	114.377	0
user age(years)	420,367	15.14	11	14.179	10
account age(days)	421,759	149.813	64	210.763	0

Table B.10: Only Collaborative Remixes: *Male*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	276,633	32.997	15	167.532	1
views	276,633	18.553	9	83.416	1
anonviews	276,633	1.089	0	9.456	0
favoriters	276,633	0.49	0	4.477	0
loveits	276,633	0.668	0	6.186	0
versions	276,633	1.208	1	1.044	1
flags	5120	1.281	1	1.289	1
sprites	276,633	10.117	5	16.207	1
scripts	276,633	31.676	10	232.029	1
is_visible	276,633	0.792	1	0.406	1
remixes	276,633	0.37	0	9.031	0
remixers	276,633	0.309	0	5.831	0
downloads	276,633	2.233	1	19.637	0
downloaders	276,633	2.218	1	19.262	0
comments	276,633	2.419	0	13.812	0
commenters	276,633	1.518	0	7.523	0
galleries	276,633	0.465	0	1.932	0
blocks	276,633	257.578	69	677.035	0
block_types	276,633	20.046	16	15.137	6
costumes	276,633	40.042	15	108.412	3
sounds	276,633	6.522	2	13.362	1
ugstrings	276,633	51.161	6	644.92	0
saves	276,633	16.394	4	36.199	0
seconds_to_share(DAY)	276,633	155.437	1.085	462.554	0
seconds_to_remix(DAY)	53,868	37.853	0.787	123.409	0
user age(years)	275,377	15.455	11	15.255	11
account age(days)	276,304	145.253	60	209.979	0

Table B.11: Only Collaborative Remixes: *Female*

num_views	145,466	29.051	16	98.033	1
views	145,466	15.098	9	50.214	1
anonviews	145,466	0.879	0	3.182	0
favoriters	145,466	0.358	0	2.557	0
loveits	145,466	0.517	0	3.187	0
versions	145,466	1.137	1	0.651	1
flags	1776	1.3	1	1.179	1
sprites	145,466	6.456	3	11.161	1
scripts	145,466	15.546	4	75.841	2
is_visible	145,466	0.756	1	0.429	1
remixes	145,466	0.311	0	1.753	0
remixers	145,466	0.278	0	1.545	0
downloads	145,466	1.29	0	10.77	0
downloaders	145,466	1.285	0	10.565	0
comments	145,466	2.985	1	9.991	0
commenters	145,466	1.641	1	4.909	0
galleries	145,466	0.355	0	1.202	0
blocks	145,466	102.496	24	314.175	0
block_types	145,466	11.966	8	10.968	6
costumes	145,466	22.064	9	61.204	2
sounds	145,466	3.862	2	10.181	1
ugstrings	145,466	53.377	0	1071.93	0
saves	145,466	8.523	2	22.048	0
seconds_to_share(DAY)	145,466	110.439	0.042	405.007	0
seconds_to_remix(DAY)	27,618	25.956	0.561	93.809	0
user age(years)	144,990	14.541	11	11.847	10
account age(days)	145,455	158.476	74	211.975	0

Table B.12: Only Versioning Remixes: *Male and Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	248,833	35.173	9	257.285	1
views	248,833	18.787	5	119.308	1
anonviews	248,833	1.173	0	9.047	0
favoriters	248,833	0.643	0	7.403	0
loveits	248,833	0.926	0	9.404	0
versions	248,833	1.412	1	1.949	1
flags	2560	1.436	1	1.258	1
sprites	248,833	8.613	4	18.147	1
scripts	248,833	31.402	8	259.862	1
is_visible	248,833	0.714	1	0.452	1
remixes	248,833	0.247	0	2.203	0
remixers	248,833	0.179	0	1.503	0
downloads	248,833	2.662	0	29.866	0
downloaders	248,833	2.629	0	29.017	0
comments	248,833	2.851	0	20.198	0
commenters	248,833	1.724	0	11.361	0
galleries	248,833	0.496	0	2.937	0
blocks	248,833	211.577	58	639.343	0
block_types	248,833	17.252	14	13.687	0
costumes	248,833	25.607	10	90.926	3
sounds	248,833	4.926	2	12.248	1
ugstrings	248,833	46.752	5	727.419	0
saves	248,833	11.46	3	32.018	0
seconds_to_share(DAY)	248,833	16.296	0.001	262.859	0
seconds_to_remix(DAY)	34,316	28.979	0	122.338	0
user age(years)	248,034	17.447	13	15.869	11
account age(days)	248,735	143.811	46	224.283	0

Table B.13: Only Versioning Remixes: *Male*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	180,081	39.702	10	294.594	1
views	180,081	21.47	5	135.449	1
anonviews	180,081	1.269	0	9.884	0
favoriters	180,081	0.743	0	8.477	0
loveits	180,081	1.072	0	10.77	0
versions	180,081	1.446	1	2.022	1
flags	2059	1.42	1	1.31	1
sprites	180,081	9.525	5	19.774	1
scripts	180,081	36.207	11	300.72	1
is_visible	180,081	0.715	1	0.451	1
remixes	180,081	0.262	0	2.301	0
remixers	180,081	0.187	0	1.512	0
downloads	180,081	3.229	0	34.46	0
downloaders	180,081	3.186	0	33.452	0
comments	180,081	3.015	0	22.316	0
commenters	180,081	1.864	0	12.722	0
galleries	180,081	0.564	0	3.352	0
blocks	180,081	250.724	76	714.647	0
block_types	180,081	19.354	17	14.215	0
costumes	180,081	29.362	11	102.971	3
sounds	180,081	5.383	2	13.624	1
ugstrings	180,081	52.191	7	758.871	0
saves	180,081	12.875	3	35.2	0
seconds_to_share(DAY)	180,081	16.495	0.003	262.636	0
seconds_to_remix(DAY)	25,615	32.889	0	130.025	0
user age(years)	179,515	17.256	12	16.337	11
account age(days)	179,992	152.295	51	230.496	0

Table B.14: Only Versioning Remixes: *Female*

Attribute	<i>N</i>	mean	median	sd	mode
num_views	68,752	23.309	7	109.864	1
views	68,752	11.76	4	58.27	1
anonviews	68,752	0.922	0	6.345	0
favoriters	68,752	0.38	0	3.169	0
loveits	68,752	0.544	0	4.005	0
versions	68,752	1.323	1	1.74	1
flags	501	1.503	1	1.015	1
sprites	68,752	6.227	3	12.644	1
scripts	68,752	18.815	4	85.532	1
is_visible	68,752	0.711	1	0.453	1
remixes	68,752	0.208	0	1.922	0
remixers	68,752	0.158	0	1.477	0
downloads	68,752	1.177	0	10.715	0
downloaders	68,752	1.168	0	10.646	0
comments	68,752	2.422	0	13.106	0
commenters	68,752	1.356	0	6.557	0
galleries	68,752	0.315	0	1.321	0
blocks	68,752	109.039	27	356.606	0
block_types	68,752	11.747	9	10.339	0
costumes	68,752	15.773	7	44.907	3
sounds	68,752	3.728	2	7.402	1
ugstrings	68,752	32.506	2	637.514	0
saves	68,752	7.754	1	21.119	0
seconds_to_share(DAY)	68,752	15.778	0	263.443	0
seconds_to_remix(DAY)	8701	17.471	0	95.291	0
user age(years)	68,519	17.949	13	14.562	11
account age(days)	68,743	121.598	35	205.481	0



# Bibliography

- Albrecht, A. J. and Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, SE-9(6):639– 648.
- Aragon, C., Poon, S., Monroy-Hernández, A., and Aragon, D. (2009). A tale of two on-line communities: Fostering collaboration and creativity in scientists and children. In *Everyday creativity: shared languages and collective action*, Berkley, CA. ACM.
- Arikan, H. B. (2006). *Collective systems for creative expression*. Thesis, Massachusetts Institute of Technology. Thesis (S.M.)—Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2006.
- Aufderheide, P. and Jaszi, P. (2011). *Reclaiming Fair Use: How to Put Balance Back in Copyright*. University Of Chicago Press.
- Bader-Natal, A., Monroy-Hernández, A., Zamfirescu-Pereira, J., and Farnham, S. (2012). Meta-remix: Reflecting on four communities built for learning, tinkering, and remixing with code.
- Batts, D. A. (2011). Patrick cariou vs richard prince, gagosian gallery, inc., lawrence gagosian, and rizzoli international publications, inc.
- Beenen, G., Ling, K., Wang, X., Chang, K., Frankowski, D., Resnick, P., and Kraut, R. E. (2004). Using social psychology to motivate contributions to online communities. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 212–221, Chicago, Illinois, USA. ACM.
- Benkler, Y. (2002). Coase’s penguin, or, linux and the nature of the firm. *Yale Law Journal*, 112(3):369.
- Benkler, Y. (2006). *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press, New Haven, CT. Some chapters.
- Benkler, Y. (2009). Law, policy, and cooperation. In Balleisen, E. and Moss, D., editors, *Government and Markets: Toward a New Theory of Regulation*. Cambridge University Press.

- Bernstein, M. S., Monroy-Hernández, A., Harry, D., Andre, P., Panovich, K., and Vargas, G. (2011). 4chan and /b/: An analysis of anonymity and ephemerality in a large online community. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, Barcelona, Spain. AAAI Press.
- Bolin, M., Webber, M., Rha, P., Wilson, T., and Miller, R. C. (2005). Automation and customization of rendered web pages. *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 163–172. ACM ID: 1095062.
- Bown, O., Eldridge, A., and McCormack, J. (2009). Understanding interaction in contemporary digital music: from instruments to behavioural objects. *Organised Sound*, 14(02):188–196.
- boyd, d., Golder, S., and Lotan, G. (2010). Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, HICSS '10, pages 1–10, Washington, DC, USA. IEEE Computer Society.
- Brandt, J., Guo, P. J., Lewenstein, J., and Klemmer, S. R. (2008). Opportunistic programming: how rapid ideation and prototyping occur in practice. In *Proceedings of the 4th international workshop on End-user software engineering*, WEUSE '08, pages 1–5, New York, NY, USA. ACM.
- Brown, G. O. (2004). Announcing (and explaining) our new 2.0 licenses. <http://creativecommons.org/weblog/entry/4216>.
- Bruckman, A. (1998). Community support for constructionist learning. *Computer Supported Cooperative Work (CSCW)*, 7(1):47–86.
- Bruns, A. (2007). Probusage. In *Proceedings of the 6th ACM SIGCHI conference on Creativity and cognition*, pages 99–106, Washington, DC, USA. ACM.
- Burke, M. and Kraut, R. (2008a). Mind your ps and qs: the impact of politeness and rudeness in online communities. In *Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 281–284, San Diego, CA, USA. ACM.
- Burke, M. and Kraut, R. (2008b). Mopping up: modeling wikipedia promotion decisions. In *Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 27–36, San Diego, CA, USA. ACM.
- Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S. (2007). I tube, you tube, everybody tubes: analyzing the world’s largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 1–14, New York, NY, USA. ACM.
- Charmaz, K. (2006). *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. Sage Publications, London.

- Cheliotis, G. and Yew, J. (2009). An analysis of the social structure of remix culture. In *Proceedings of the fourth international conference on Communities and technologies*, pages 165–174, University Park, PA, USA. ACM.
- Chen, S. (2007). The state of our video ID tools.
- comScore (2010). comScore releases may 2010 U.S. online video rankings. [http://www.comscore.com/Press\\_Events/Press\\_Releases/2010/6/comScore\\_Releases\\_May\\_2010\\_U.S.\\_Online\\_Video\\_Rankings](http://www.comscore.com/Press_Events/Press_Releases/2010/6/comScore_Releases_May_2010_U.S._Online_Video_Rankings).
- Davis, Z. (1966). The reviews. *HiFi/Stereo Review*, 17:146–147.
- Diakopoulos, N., Luther, K., Medynskiy, Y. E., and Essa, I. (2007). The evolution of authorship in a remix society. In *Proc. HT 2007*, pages 133–136, Manchester, UK. ACM.
- Donath, J. (2008). Signals in social supernets. *Journal of Computer-Mediated Communication*, 13(1):231–251.
- Donath, J. S. (1998). Identity and deception in the virtual community. In Smith, M. A. and Kollock, P., editors, *Communities in Cyberspace*, pages 27–57. Routledge.
- Economist, T. (2011). Musical history: Seven seconds of fire. *The Economist*.
- EvanitaEWM (2010). deviantART 10th birthday bash at house of blues - angelo sotira’s closing speech PT 2.
- Ferguson, K. (2010). Everything is a remix part 1.
- Fildes, J. (2007). Free tool offers ‘easy’ coding. *BBC*.
- Frakes, W. and Terry, C. (1996). Software reuse: metrics and models. *ACM Computing Surveys*, 28:415–435.
- Gaffney Jr, J. and Durek, T. (1989). Software reuse—key to enhanced productivity: some quantitative models. *Information and Software Technology*, 31(5):258–267.
- Healy, K. and Schussman, A. (2003). The ecology of open-source software development.
- Hemphill, S. C. and Suk, J. (2009). Remix and cultural production. *Stanford Law Review*, 61(1227).
- Hill, B. M. H., Monroy-Hernández, A., and Olson, K. (2010). Responses to remixing on a social media sharing website. In *Proc. ICWSM 2010*, pages 74–81, Washington, D.C. AAAI.
- Inc., T. (2012). Twitter / twitter terms of service. [https://twitter.com/tos/previous/version\\_6](https://twitter.com/tos/previous/version_6).
- Ito, M. (2007). Technologies of the childhood imagination: Yu-gi-oh!, media mixes, and everyday cultural production. In Karaganis, J., editor, *Structures of Participation in Digital Culture*, pages 86–111. The Social Science Research Council, New York.

- Ito, M. (2010). *Hanging out, messing around, and geeking out : kids living and learning with new media*. John D. and Catherine T. MacArthur Foundation series on digital media and learning. MIT Press, Cambridge Mass.
- Jenkins, H. (2006). *Convergence Culture: Where Old and New Media Collide*. NYU Press, revised edition.
- Jenkins, H. (2007). Nine propositions towards a cultural theory of YouTube.
- Jenkins, H., Purushotma, R., Clinton, K., Weigel, M., and Robinson, A. (2009). *Confronting the Challenges of Participatory Culture: Media Education for the 21st Century*. John D. and Catherine T. MacArthur Foundation Reports on Digital Media and Learning. The MIT Press, Cambridge, MA.
- Johnson, C. Y. (2007). With simplified code, programming becomes child's play. *Boston.com*.
- Keen, A. (2007). *The Cult of the Amateur: How Today's Internet is Killing Our Culture*. Crown Business, 3rd printing edition.
- Kim, M., Bergman, L., Lau, T., and Notkin, D. (2004). An ethnographic study of copy and paste programming practices in OOPL. In *Proceedings of the 2004 International Symposium on Empirical Software Engineering, ISESE '04*, pages 83–92, Washington, DC, USA. IEEE Computer Society.
- Kittur, A. and Kraut, R. E. (2008). Harnessing the wisdom of crowds in wikipedia: quality through coordination. In *Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 37–46, San Diego, CA, USA. ACM.
- Kittur, A. and Kraut, R. E. (2010). Beyond wikipedia: coordination and conflict in online production groups. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 215–224, Savannah, Georgia, USA. ACM.
- Kollock, P. (1998). Design principles for online communities. *PC Update*, 15(5):60, 58.
- Lave, J. and Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, 1 edition.
- Lenhart, A. and Madden, M. (2005). Teen content creators and consumers. Technical report, Pew Internet and American Life Project.
- Lessig, L. (2000). *Code and Other Laws of Cyberspace*. Basic Books.
- Lessig, L. (2008). *Remix: Making Art and Commerce Thrive in the Hybrid Economy*. Penguin Press, New York, NY.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

- Livingstone, S. (2008). Taking risky opportunities in youthful content creation: teenagers' use of social networking sites for intimacy, privacy and self-expression. *New Media & Society*, 10(3):393–411.
- Loudon, J. C. (1839). *The Gardener's magazine and register of rural & domestic improvement*. Longman, Rees, Orome, Brown and Green.
- Luther, K. and Bruckman, A. (2008). Leadership in online creative collaboration. In *Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 343–352, San Diego, CA, USA. ACM.
- Luther, K., Caine, K., Zeigler, K., and Bruckman, A. (2010). Why it works (when it works): Success factors in online creative collaboration. In *Proceedings of the ACM Conference on Supporting Group Work.*, Sanibel Island, Florida, USA. ACM.
- Maillart, T., Sornette, D., Spaeth, S., and von Krogh, G. (2008). Empirical tests of zipf's law mechanism in open source linux distribution. *Physical Review Letters*, 101(21):218701.
- Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. (2004). Scratch: a sneak preview. In *Second International Conference on Creating, Connecting and Collaborating through Computing*, pages 104–109. IEEE.
- Manovich, L. (2005). Remix and remixability.
- Media, N. B. (1979). *Billboard*. Nielsen Business Media, Inc.
- Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M. A., and Aiden, E. L. (2011). Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.
- Minsky, M. (1984). LogoWorks: challenging programs in logo. In Solomon, C., Minsky, M., and Harvey, B., editors, *Introduction to LogoWorks*. McGraw-Hill.
- mobius32 (2006). Video explains the world's most important 6-sec drum loop.
- Monroy-Hernández, A. (2006). TWiki - MAS712 web - ClassReflections - GroupThree - AndresMonroyHernandez. <http://wiki.media.mit.edu/view/MAS712/AndresMonroyHernandez?rev=11>.
- Monroy-Hernández, A. (2007). ScratchR: sharing user-generated programmable media. *Proceedings of the 6th international conference on Interaction design and children*, pages 167–168. ACM ID: 1297315.
- Monroy-Hernández, A. (2011). Here's what 44 million scratch scripts look like.
- Monroy-Hernández, A., Hill, B. M., Gonzalez-Rivero, J., and boyd, d. (2011). Computers can't give credit: How automatic attribution falls short in an online remixing community. In *Proceedings of the 29th international conference on Human factors in computing systems*, Vancouver, British Columbia, Canada. ACM Press.

- Monroy-Hernández, A. and Resnick, M. (2008). Empowering kids to create and share programmable media. *interactions*, 15(2):50–53.
- Murray, F. and O’Mahony, S. (2007). Exploring the foundations of cumulative innovation: Implications for organization science. *Organization Science*, 18(6):1006–1021.
- Nielsen, J. (2005). Usability of websites for teenagers (Jakob nielsen’s alertbox). <http://www.useit.com/alertbox/teenagers.html>.
- O’Reilly, T. (2005). What is web 2.0 - design patterns and business models for the next generation of software. <http://oreilly.com/web2/archive/what-is-web-20.html>.
- Ortega, F. (2009). *Wikipedia: A Quantitative Analysis*. PhD dissertation, Universidad Rey Juan Carlos.
- Papazoglou, M. P. and Georgakopoulos, D. (2003). Introduction: Service-oriented computing. *Commun. ACM*, 46(10):24–28.
- Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books, New York.
- Paul, R. (2012). Linux kernel in 2011: 15 million total lines of code and microsoft is a top contributor. <http://arstechnica.com/business/news/2012/04/linux-kernel-in-2011-15-million-total-lines-of-code-and-microsoft-is-a-top-contributor.ars>.
- Perkel, D. (2008). Copy and paste literacy: Literacy practices in the production of a MySpace profile. In Drotner, K., Jensen, H. S., and Schroder, K. C., editors, *Informal Learning and Digital Media*, pages 203–224. Cambridge Scholars Press, Newcastle, UK.
- Philip, K., Umarji, M., Agarwala, M., Sim, S. E., Gallardo-Valencia, R., Lopes, C. V., and Ratanotayanon, S. (2012). Software reuse through methodical component reuse and amethodical snippet remixing. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, CSCW ’12*, pages 1361–1370, New York, NY, USA. ACM.
- Pietrolungo, S. (2001). Singles minded. *Billboard*.
- Pinch, T. J. and Bijker, W. E. (1984). The social construction of facts and artefacts: Or how the sociology of science and the sociology of technology might benefit each other. *Social Studies of Science*, 14(3):399–441.
- Porter, C. E. and Donthu, N. (2008). Cultivating trust and harvesting value in virtual communities. *Management Science*, 54(1):113–128.
- Posner, R. A. (2007). *The little book of plagiarism*. Pantheon Books.
- Preece, J. (2000). *Online Communities: Designing Usability and Supporting Sociability*. Wiley, 1 edition.
- Quickmeme (2011). Yo dawg i heard you like remixes so i put a remix in your remix. <http://www.quickmeme.com/meme/35eehi/>.

- Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11):60–67.
- Schwaber, K. and Beedle, M. (2001). *Agile Software Development with Scrum*. Prentice Hall, 1 edition.
- Seneviratne, O., Kagal, L., and Berners-Lee, T. (2009). Policy-aware content reuse on the web. In Bernstein, A., Karger, D., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., and Thirunarayan, K., editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 553–568. Springer Berlin / Heidelberg.
- Shaw, R. and Schmitz, P. (2006). Community annotation and remix: a research platform and pilot deployment. *Proceedings of the 1st ACM international workshop on Human-centered multimedia*, pages 89–98. ACM ID: 1178761.
- Sinnreich, A. (2010). *Mashed Up: Music, Technology, and the Rise of Configurable Culture*. University of Massachusetts Press.
- Sinnreich, A., Latonero, M., and Gluck, M. (2009). Ethics reconfigured: How today’s media consumers evaluate the role of creative reappropriation. *Information, Communication & Society*, 12(8):1242.
- Stallman, R. (1985). The GNU manifesto. *Dr. Dobb’s Journal of Software Tools*, 10(3):30.
- Stone, V. (2009). ccMixer: a memoir.
- Thorne, M. (2009). Analysis of 100M CC-Licensed images on flickr.
- Tran, T. (2009). YouTube goes offline.
- Turkle, S. and Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. *Signs*, 16(1):128–157.
- Viegas, F. B., Wattenberg, M., and Dave, K. (2004). Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 575–582, Vienna, Austria. ACM.
- von Hippel, E. (2005). *Democratizing innovation*. The MIT Press, Cambridge, Massachusetts.
- von Krogh, G. and von Hippel, E. (2006). The promise of research on open source software. *Management Science*, 52(7):975–983.
- Walston, C. E. and Felix, C. P. (1977). A method of programming measurement and estimation. *IBM Systems Journal*, 16(1):54–73.
- Wenger, E. (1998). Communities of practice learning as a social system. *1998*, 9(5).

- Williams, E. (2009). Why retweet works the way it does.
- Wong, J. and Hong, J. I. (2007). Making mashups with marmite: towards end-user programming for the web. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444. ACM ID: 1240842.
- YouTube, L. (2010). What are the criteria for partnership? <http://support.google.com/youtube/bin/answer.py?hl=en&answer=82839#US>.
- Zang, N. and Rosson, M. B. (2008). What’s in a mashup? and why? studying the perceptions of web-active end users. *Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 31–38. ACM ID: 1550043.
- Zittrain, J. (2008). *The Future of the Internet—And How to Stop It*. Yale University Press.
- Zonk (2007). MIT media lab making programming fun for kids. <http://tech.slashdot.org/story/07/05/15/1420238/mit-media-lab-making-programming-fun-for-kids>.