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
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 chapter¹, 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.

¹Based 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, 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.”

All user names are changed to protect the privacy of the participants.
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 GreenDinosuar, 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
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 sequal 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.
Figure 4-3: Chocolate Bar episodic series.
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 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).

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

---

3 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 wondering, 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.](image)

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).

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
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\(^4\) 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 llooooott.” 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.

![First Scrolling](image)

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).

\(^4\)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 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\textsuperscript{1} “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:

\begin{quote}
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.
\end{quote}

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.

\footnote{\textsuperscript{1}Digital Millennial Copyright Act}
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, Scratch, that enables people to share and remix programmable media. Released under a GPL\(^2\) v2 license, Scratch has been reused a number of times. For example, the Portuguese telecommunications company Sapo 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 generatively, 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

---

\(^2\)General Public License
Figure 7-2: Instance of ScratchR for Portugal.
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 programmists [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.