Extending ScratchR:
Challenges of Deploying a Scalable User Generated Content Platform for Young People

by
Han Xu

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science at the

Massachusetts Institute of Technology

August 20, 2008

Author_____________

Department of Electrical Engineering and Computer Science
August 20, 2008

Certified by________

Mitchel Resnick
Professor of Learning Research, MIT Media Lab
Thesis Supervisor

Accepted by_

Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

Extending ScratchR: Challenges of Deploying a Scalable User Generated Content Platform for Young People
by
Han Xu

Submitted to the
Department of Electrical Engineering and Computer Science

August 20, 2008

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

ScratchR, a user generated content management platform, was developed by the Lifelong Kindergarten group at the MIT Media Lab. ScratchR lets users share projects developed with the Scratch programming language. This thesis involves technical improvements made to ScratchR from August 2007 to August 2008. Improvements include the addition of a comment reply system, the improvement of the gallery system, the addition of AJAX functionality, the creation of administration tools and the addition of a peer review system for comments and tags. These features augment the user experience of the Scratch online community by improving usability and by creating a friendlier online learning environment.

Thesis Supervisor: Mitchel Resnick
Title: Program Head of Media Arts and Sciences

Acknowledgements

First I have to thank Mitch Resnick for providing me a great opportunity to study at the Lifelong Kindergarten group and for mentoring me throughout the last academic year. I also want to thank Andres Monroy-Hernandez for his patience and support during the course of my thesis work. Finally, I would like to thank everyone at the Lifelong Kindergarten group for their brilliant ideas and contributions to both Scratch and ScratchR.

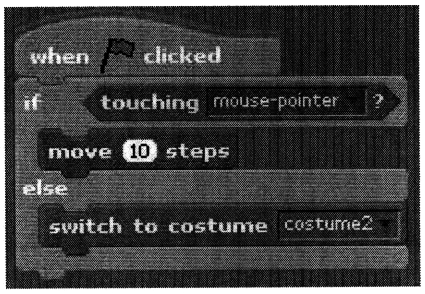
Table of Contents

1 Introduction

The Lifelong Kindergarten Group at the MIT Media Lab have developed two software packages that aim to help young people develop 21st century learning skills. The first is the Scratch programming language, which makes it easy for users to create interactive media. The second is ScratchR, which powers the Scratch Online Community. ScratchR allows users to reuse, remix and build upon other people's Scratch projects. This thesis explores my contributions to the ScratchR project.

1.1 Scratch

Scratch is a programming language developed by the Lifelong Kindergarten group at the MIT Media Lab that makes it easy for users to create interactive stories, games, music and animations. The language is designed to help young people develop 21st century learning skills. As users create and share Scratch projects, they learn important mathematical and computational ideas, while also gaining a deeper understanding of the design process [1].



```
when [flag] clicked
if touching mouse-pointer ?
  move 10 steps
else
  switch to costume costume2
```

Figure 1. Scratch blocks

Unlike traditional programming languages, Scratch is based on a drag and drop interface that utilizes predefined blocks (blocks encapsulate some programming functionality) instead of syntax-based-tokens. Users create scripts by chaining together these coding blocks instead of writing lines of code. This unique mechanism of Scratch creates a low barrier of entry for novice users of the programming language. Users who have never programmed before can pick up Scratch and create a non-trivial project in under an hour.

Although Scratch allows users to easily dive in, the language also has depth. Recently, users of the Scratch Online Community have begun publishing side scrolling games that span many levels and contain upwards of hundreds of sprites and scripts.

1.2 ScratchR

ScratchR is the engine behind the Scratch online community, which enables users to publish and share the projects that they create with in Scratch [2]. Unlike other user-generated content communities, ScratchR makes it easy to reuse, remix and build upon other people's creations in a process called creative appropriation. The goal of this online community is to create a friendly environment where users can learn from each other by viewing and analyzing each other's projects.

Although ScratchR was designed with Scratch in mind, the engine can be adapted to a broad range of content including images and videos. ScratchR currently supports projects and project galleries. The engine supports category based browsing of projects based on user generated tags. Users can also create theme based galleries that contain projects related by some user defined criteria. Since the sharing of ideas is a main theme of the online community, the ScratchR engine also features a robust comment and reply system complete with spam filtering and automatic censoring of inappropriate content.

Moderation of the Scratch online community is handled largely by the users of the website. With more than 140,000 registered users and 30,000 project contributors, it would be impossible to review all the incoming comments, projects and tags; therefore, ScratchR features peer review mechanisms that allow users to automatically censor inappropriate content on the site based on consensus. In addition to these reporting mechanisms, ScratchR also features a comprehensive administration backend that allows administrators to pinpoint problem content on the site and take appropriate action.

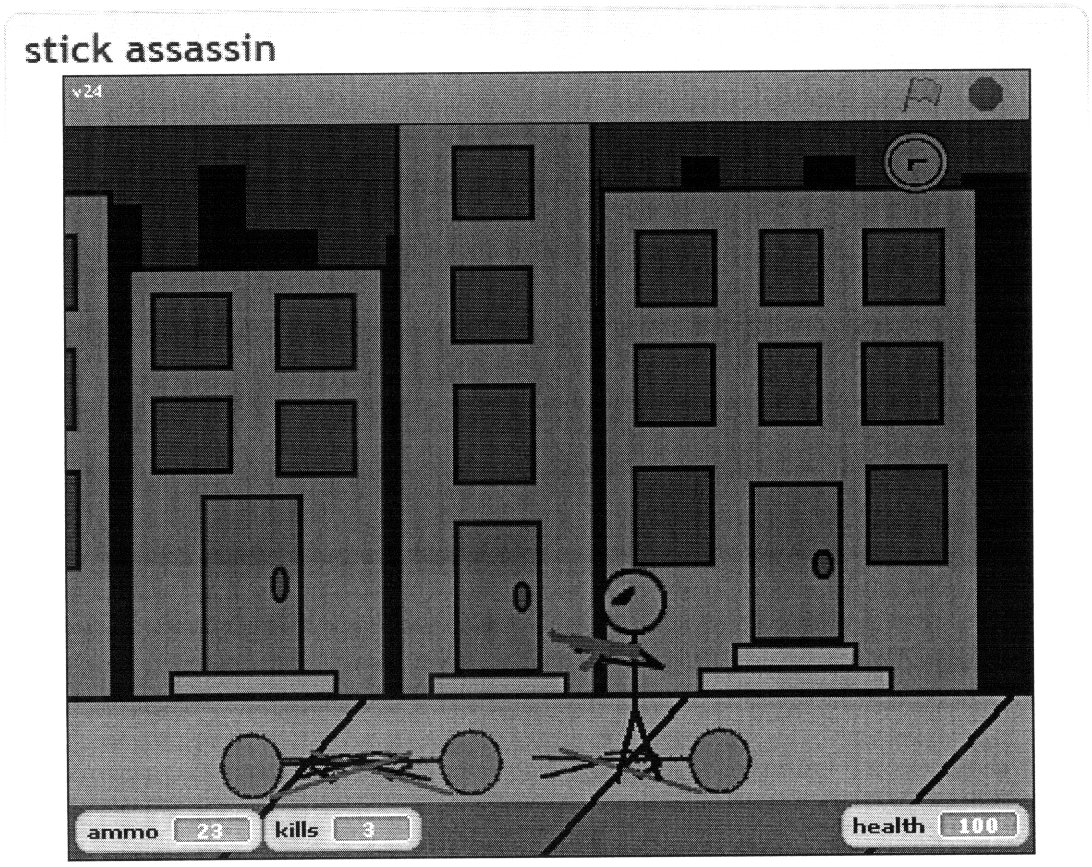


Figure 2. Side scrolling game made using Scratch posted on ScratchR

1.3 Guide to Thesis

This thesis begins with an overview of the technical and human environment surrounding the development and deployment of the ScratchR content management engine. The following section examines the state of ScratchR in August 2007. Then, the next section describes the related work and methodologies that have inspired the design of ScratchR. The next section describes my contributions to the ScratchR engine and the improvements I have made to the website. In this section, I explore design alternatives and give justification for the implemented design. The next section examines the effectiveness of the site improvements. I continue by analyzing the software development principles that I learned from working on ScratchR. Then, I conclude by discussing future work.

ScratchR will be used to denote both the engine behind the Scratch online community and the current version of the Scratch website. The Scratch online community will be used to refer to the Scratch website located at scratch.mit.edu and to the community of the users on the website.

2 Human and Technical Environment

2.1 Scratch Software Team

The core software development team behind ScratchR is led by the founder of the project, Andres Monroy-Hernández. Between August 2007 and August 2008, several undergraduate research assistants of the Lifelong Kindergarten Group contributed modules to ScratchR including one for comment spam detection. The translation engine was also integrated during this period by an independent contributor from Europe. My focus during this period was on the development of front end features and the removal of bugs on the site. The software team also has one member dedicated to optimizing the Apache server in order to handle the increasing server load.

2.2 Scratch Design Team

In addition to the software development team, everyone in the Lifelong Kindergarten Group contributes to the ScratchR design process. The bulk of the design work on ScratchR occurs during the weekly meetings where members of the Scratch Team propose new features to implement.

2.3 Design Constraints

The majority of the ScratchR user base being young people eight and up poses unique challenges in the design and deployment of the website. One major constraint in designing new features for ScratchR lies in our inability to collect personal identifying information from children under the age of 13. The Scratch Team cannot ask for email addresses or location information.

The average age of the user base also creates an online environment that is extremely sensitive to inappropriate content. Inappropriate content on ScratchR may lead to parents or schools censoring the site. Since the cost of showing inappropriate content is so high, dealing with inappropriate content in a timely manner is critical to the success of the Scratch Online Community. Furthermore, determining what is inappropriate on the

ScratchR is a much greater challenge. This is especially true for borderline cases where a project displays overt violence but is also very well programmed. Often, the Scratch Team has to make a decision balancing the negative aspects of a project with the positive aspects in order to determine whether a project is appropriate for the site. After a year of reviewing numerous projects, it is still difficult to develop a comprehensive set of rules for dealing with all the content on the site and often times project appropriateness will be determined by discussion and group consensus.

2.4 Technical Constraints

ScratchR is built using PHP and MySQL on an Apache server. We utilize a centralized LAMP development environment hosted by the MIT Media Lab.

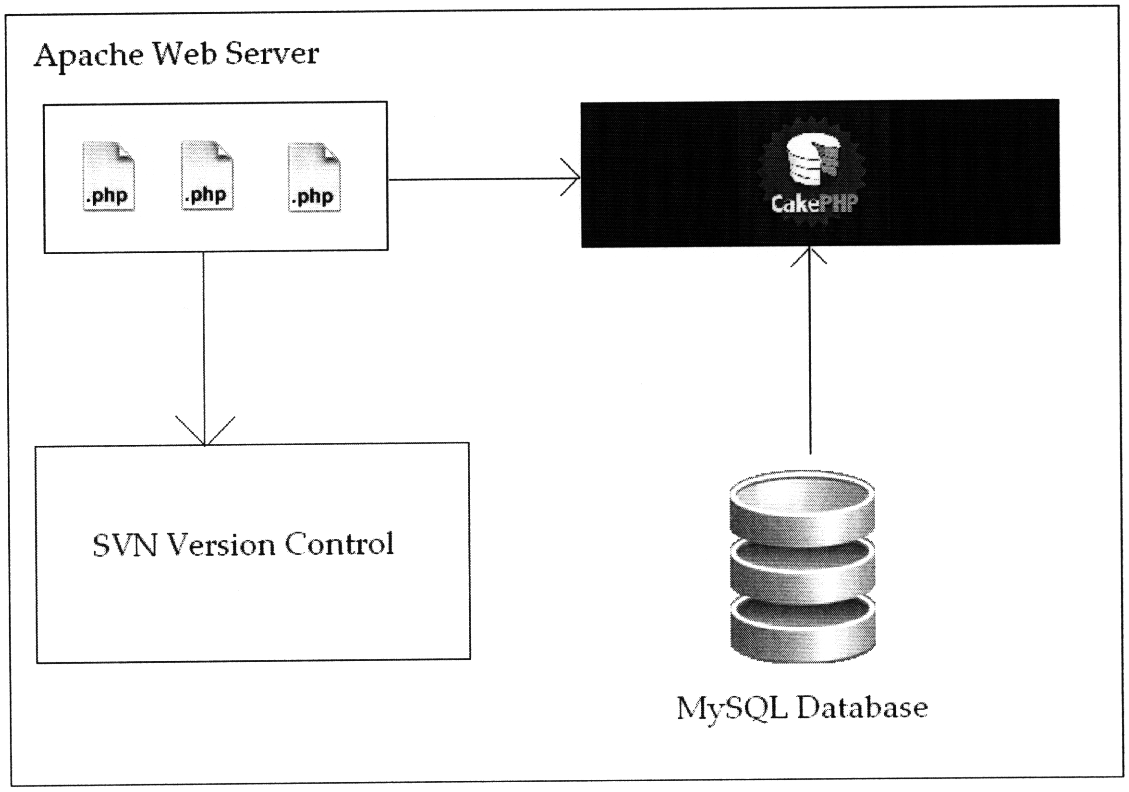


Figure 3. Server side architecture of ScratchR

To speed up development, we utilize the CakePHP framework which is an MVC framework developed for PHP. CakePHP offers several key advantages including the separation of the programming logic from the presentation and robust ORM (Object Relational Mapping). CakePHP is also supported by a large community of developers who contribute useful packages to the framework.

Front end development of ScratchR is done using 4.01 compliant HTML and JavaScript. We utilize two major JavaScript libraries; we use Prototype for AJAX functionality and JQuery for event handling.

For version control, we use SVN. In addition to the live server at scratch.mit.edu, we also utilize a set of test servers (one for each developer on the team). These test servers allow developers to test new features concurrently without interfering with the progress of other developers.

2.5 Design Process

ScratchR has been and always will be open to improvement. As the user base increases, users demand new features for improving communication and collaboration, administrators need new tools to better monitor the site and developers require continuous refactoring of the source code in order to improve productivity.

In order to facilitate the development of ScratchR, the Scratch Team embraces the agile development methodology. To accommodate the adaptive nature of an online community, the Scratch Team meets weekly in order to prioritize objectives. Suggestions

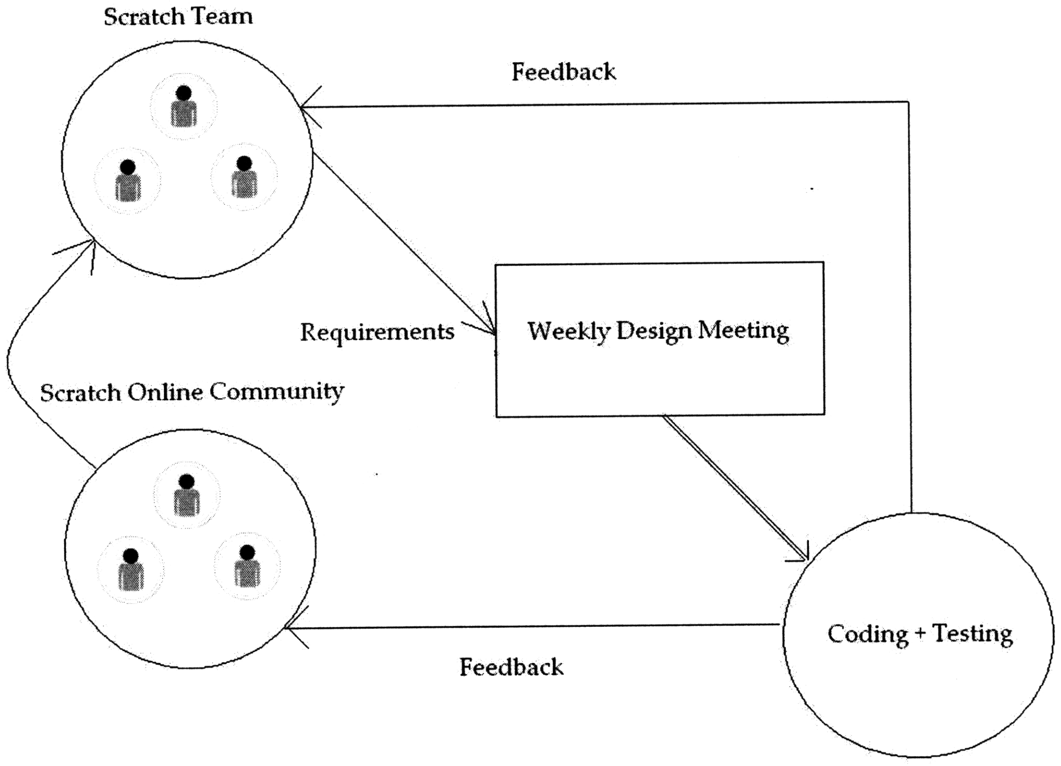


Figure 4. ScratchR design cycle

of new features are raised by team members during the meeting or by community members through the use of the online forums.

The core development process involves creating new features and repairing bugs. To facilitate the development process, we track each outstanding issue via a Bugzilla server that can be edited by everyone on the Scratch Team. Issues are prioritized based on the overall impact to the user experience and are ranked as follows:

- Blocker: Bugs that prevent users from accessing any of the core functionality on the site such as sharing projects, creating galleries or posting comments.
- Critical: Issues that cause significant discontinuity in the user experience, but are not critical to the core functionality of the site. Examples include database issues that cause project views to be counted incorrectly or a bug that prevents gallery owners from removing members from their galleries.
- Major: Typically includes updates to existing features or new features that have a significant impact on the user experience. Examples include the comment reply system, comment character counter, and improvements to the administration backend.
- Minor: Site enhancement such CSS problems or backwards compatibility with older browsers such as IE 6.

Implementation of bug fixes and new features not only depend on their severity but also upon the difficulty of the implementation taking into consideration the existing source code. In order to maximize my efficiency, I attempt to chunk the issues on Bugzilla into functional groups that are related to each other based on proximity in the source code (belonging to one controllers or a group of related view elements). By applying this methodology after all the blocker issues have been taken care of, I find that it significantly increases my productivity especially by helping to concentrate code refactoring efforts that result in future productivity gains.

3 Related Work

3.1 Agile Software Development

The design and implementation cycle of ScratchR takes advantage of several key principles of agile software development [3].

- Short development cycles. Development of ScratchR emphasizes short development iterations that include planning, requirement analysis, design, coding and testing.
- Individuals and interactions over processes and tools. The design process of ScratchR is driven by close interaction of the Scratch Team with the development team. Site administrators work closely with the Scratch online community in order to generate new requirements during each development cycle.
- Working software over comprehensive documentation. To serve a large community of users, the development of ScratchR emphasizes working software. Documentation takes a back seat to writing code that can be quickly deployed and tested and that generates an immediate impact on the user experience.
- Responding to change over following a plan. The online community behind Scratch changes constantly due to the influx of new users and the rapid exchange of ideas. Proposed features are often no longer needed weeks after they are proposed. To adapt quickly to the changing requirements of our user

base, the ScratchR development team favors extensible code and flexible features over long term planning.

- Sustainable development. The ScratchR development team embraces code refactoring as a way of life. In order to quickly develop new features needed by the Scratch online community, we continuously refactor the source code in order to increase its readability and to simplify its structure.

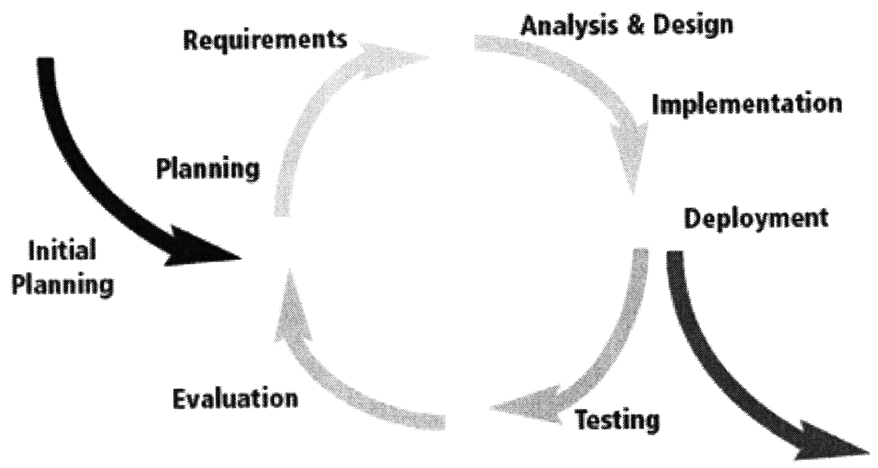


Figure 5. Agile software development cycle

3.2 Large User Generated Content Sites

The design of ScratchR takes inspiration from the leaders of the user generated content space. Sites such as Digg, YouTube and Flickr all contain numerous features that we have adapted to ScratchR. The comment reply system on ScratchR contains many of the same features as the comment reply system of Digg [4]. When designing our peer review system, the development team first examined the peer review systems of YouTube and

Flickr [5]. Furthermore, we plan to implement a free tagging system similar to the current tagging system of TED.com in order to further improve the project categorization capabilities of ScratchR [6].

4 Sites Issues as of August 2007

As the Scratch Online Community grows, it faces a growing number of issues that were not addressed in the initial launch of ScratchR. These issues range from the incomplete feature set in the comment system to the inadequate user interface of the galleries system. In the following section, I will introduce the major issues that have faced and currently face ScratchR.

4.1 Inappropriate Content

Dealing with inappropriate content is a major challenge with any user generated content site. This is especially true with the Scratch online community. Nudity on a site such as YouTube does not cause significant problems except to the few viewers who may find it vulgar; however, nudity on ScratchR can lead to schools and parents censoring the site for many legitimate users. Therefore, dealing with any inappropriate content that appears on the Scratch online community is a top priority for the Scratch Team.

Inappropriate content appear in many forms. Comments, projects, gallery descriptions and user names can all contain inappropriate content. Any user generated content can contain inappropriate elements; therefore, it is crucial for the Scratch Team to be able to find and remove all inappropriate text, images and projects that users post to the site.

This problem is complicated by the fact that different users find different content to be inappropriate. For example, a project parodying President Bush may be funny to the majority of the users of the Scratch online community, but a minority of users may find it inappropriate. Furthermore, themes such as violence and guns are present in many well made games appearing on the site, yet such themes are commonly found to be inappropriate for children.

One possible solution to the problem is to have all content reviewed before being posted to the site. By implementing this solution, there would be no inappropriate content appearing on the site; however, this solution takes an extremely large amount of man power to implement due to the number of comments and projects posted each day. Furthermore, members of the Scratch Team have expressed concerns over stifling the creativity of users on the site due to over censoring. Specifically, team members did not like the idea of new projects not showing up immediately on the front page as many team members felt that this feature is an incentive for users to start contributing and to continue contributing projects to the site.

Since it would not be feasible to automatically filter all the user generated content incoming to the site, any viable solution would rely on the vigilance of the users of the Scratch online community.

4.2 Comments and Spamming

Communication is the key element of the Scratch online community. In order for members to learn from each other, they must be able to share their ideas and critiques in an organized manner. However, comments that are not reviewed often result in spam that degrades the overall user experience.

When the comment system was initially introduced, it did not come with a spam filter. This caused many users to advertise their projects in the comments of other users' projects resulting in chaos and degrading the quality of discourse and communication about the actual project content.

The comment system also lacked functionality for facilitating conversations. Users could only post comments about a project as there was no functionality for comment replies. This problem coupled with the lack of a spam filter placed severe limits on the usefulness of the comment system.

4.3 Galleries

The Scratch Team envisioned galleries as tools for users to build user generated categories of projects with coherent themes. However, the initial gallery system could not fulfill this role due to a variety of usability issues.

ScratchR launched with a rudimentary gallery system. Initially, users could only create private galleries, which were visible to the entire community but could only be modified by the gallery owners. It was also difficult for gallery owners to add projects to their galleries due to the unintuitive user interface.

4.4 Administration

The Scratch Team consists of less than twenty people with only two members dedicated to monitoring the website. In order for only a few administrators to monitor a site of more than 140,000 registered users, ScratchR requires an administration system.

As in every online community, the Scratch online community has its share of bad apples who attempt to ruin the experience for other users on the site. To address this issue, administrators must have the capability to easily track down malicious users and prevent them from causing further mischief.

ScratchR did not launch with an administration system, which meant that site administrators could not track the history of problem users. Furthermore, administrators could only delete user accounts. Persistent malicious users who were intent on causing trouble could not be eliminated; once one of their accounts was deleted, these users would create a new account and continue their negative activities.

4.5 Site Usability

Due to time constraints, ScratchR launched without AJAX functionality built into many features. Several key pages of ScratchR contain a relatively large number of SQL queries, which cause significant load times. Pages displaying details about users who have a large number of projects, about galleries with many comments and about popular projects loaded relatively slowly especially on dialup connections.

Since no AJAX functionality was available on the site at launch time, many of the actions available on the site were difficult to use due to the long load time in refreshing an entire page. Simple actions such as flagging a project, adding a tag or posting a comment caused tedious load times that degraded the user experience.

4.6 Harassment

As the online community around Scratch grows, disagreements inevitably erupt between users. The site moderators encourage users to work things out in a peaceful manner, but sometimes disgruntled and malicious users will attempt to ruin the experience of other users on the site through harassment.

Users have complained on the forums that their projects and galleries have been filled with hurtful comments and tags or that their projects have been wrongfully censored. Investigations into these occurrences often uncover a single individual using several accounts to harass the victim.

5 Developed Features

This section will describe the features and improvements that I have added to ScratchR. I will give design justifications and alternative designs where applicable.

5.1 Development Timeline

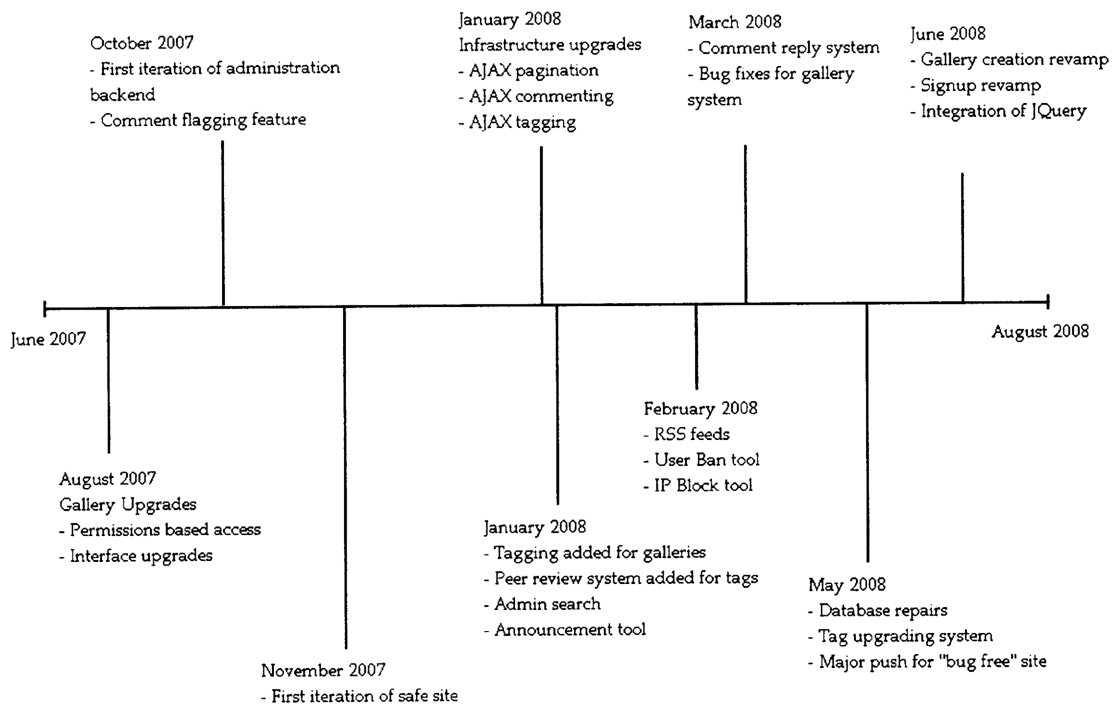


Figure 6. Development timeline from June 2007 to August 2008

5.2 Gallery System Upgrades

ScratchR launched without a complete feature set for the gallery system. Notable features that were not included in the initial release include permissions based access and the ability for owners to remove their projects from galleries.

Permissions Based Access

The current gallery system allows gallery creators to fine tune access to their galleries. Galleries have four permission settings: public, private, for friends, and customized. Pubic galleries are available to all users on the site; anyone can add their projects to public galleries. Private galleries can only be modified by the gallery creator. A gallery created for friends provides access to all current friends of the gallery creator. The customized setting allows creators to choose a subset of their friends.

This system for gallery access allows for self contained galleries that cater to a specific group of the Scratch online community. Teachers commonly use the system to create class specific galleries, while other users have created private galleries to store their favorite projects.

An alternative system that lets gallery creators edit an access list was not implemented due to the complexity of the task and the limited benefits that such a fine grained access list would provide. The final design provides a simple way for users to choose from the most commonly used access settings, while also giving them the ability for limited customization.

Interface for Adding Projects to Galleries

The initial release of ScratchR included the functionality for adding projects to galleries under the projects detail page. Many users found this interface unintuitive because it did not allow them to add multiple projects easily.

To address this problem, I added a popup to the gallery page that allows members of a gallery (users who have access to a gallery based on the gallery permissions) to add and remove all the projects that they own. Also, I extended the interface on the projects page so that users now have the ability to add a project to any gallery that they have bookmarked, created or joined.

Gallery Creation Interface

The current gallery creation interface has been upgraded from the version at launch time. The error reporting functionality has been updated so that all the errors generated on the creation form can be displayed simultaneously. This functionality helps to reduce the complexity of the gallery creation task so that users do not have to reenter information for multiple attempts at creating a gallery. Users are also given a list of predefined tags to use for their galleries. Instead of encouraging free tagging, the gallery creation process suggests a set of meaningful tags based on the top tags currently used on projects posted on the site.

5.3 Tagging System Upgrades

Upgrading Tags

The initial design of the tagging system allowed users to free tag projects; however, different users were not allowed to tag a project with the same tag. For example, if I had tagged a project as "simulation", then no other user would be able to tag that project as "simulation".

I introduced a system that allows projects to be tagged multiple times with the same tag. If multiple users tag a project with the same tag, that tag is upgraded. This behavior allows users to distinguish projects based on the weight of each tag on the project. A project tagged ten times as “simulation” and once as “art” has a stronger weight as a simulation project than an art project. To display this effect, tags with higher weight are displayed in larger fonts.

Gallery Tagging

In order to better categorize the growing number of galleries, I extended the existing tagging system from the projects to the galleries. In addition to the tagging upgrade, I also implemented a system that allows users to browse galleries based on tags. These upgrades aim to help users find the most relevant galleries based on their interests.

5.4 Comment Reply System

The initial launch of ScratchR included functionality for commenting projects and galleries, but lacked support for comment centric conversations. To address this problem, I implemented a three level comment reply system.

Users can now designate new comments as either root comments belonging to a parent project or as replies to comments that have already been posted. The system is based on an on demand comment retrieval process that relies on AJAX requests in order to display the replies belonging to a specific comment.

All root comments and their immediate replies are initially displayed. Users interested in expanding ongoing conversations can request the engine to display all of the replies belonging to a specific comment. Currently, the system allows one level of root comments and two levels of replies. The system can easily be extended to more level of replies due to the recursive nature of the algorithm behind the comment retrieval process, but testing has shown that two levels of replies are sufficient for the purposes of most conversations on ScratchR.

5.5 User Ignore-List

Although new administration tools have mitigated the amount of harassment on the site, it is difficult to monitor all user interactions and follow up on all reports of harassment. Often times, users who are harassed will leave the site without complaining to the Scratch Team. To address the issue of harassment, I implemented an ignore feature that allows users to ignore other users on the site. This feature empowers users to create a positive online learning environment.

Each user on the site has a separate User Ignore-List which she or he can modify. Each list contains a set of user names. By adding a name to this list, a user causes all the actions of the offending user to become invisible. For example, if I add andresmh to my Ignore-List, I will not be able to see any of the projects, comments, galleries or tags posted by andresmh. Effectively, users on my Ignore-List do not exist from my perspective.

5.6 Infrastructure Upgrades

AJAX Tagging, Commenting and Pagination

To improve the user experience on the site, I implemented site wide upgrades to include AJAX functionality for several key features including comments, tags and pagination. AJAX technology allows for requests to a web server to update a particular section of the web page. For example, a standard link for posting a comment would reload the entire page with the new comment attached to the comment list, while an AJAX link for posting a comment would only need to update the comment list without reloading the entire page. This is accomplished through the use of JavaScript and XML where JavaScript code sends an asynchronous request to the web server and receives an XML response. The client parses the XML response and updates the web page that is currently viewed. Instead of reloading the entire web page, AJAX based links allow for a smoother user experience by significantly reducing the load time for highly used features. This feature also helps to reduce server load by reducing the number of page requests.

Upgrade to Cake 1.19

The CakePHP framework is constantly being upgraded with bug fixes and new functionality. To leverage the most out of the underlying MVC framework behind ScratchR, I upgraded the engine to 1.19 version of the CakePHP release. The new release includes better support for AJAX functionality and pagination.

RSS Feeds

To improve usability for more experienced users and to provide support to site administrators, I implemented a system of RSS feeds. Currently there are RSS feeds for the newest projects posted to the site, for featured projects on the site, for project listings of each user and for flagged projects. Each feed includes information concerning the project name, project description, project URL and project thumbnail. Using the feeds, administrators and users can keep track of the projects that they find most relevant.

5.7 Peer Review System

Dealing with inappropriate content is a top priority for the Scratch online community. Filtering all content to the site is not a scalable solution. Blanket filtering also stifles creativity. Therefore, the Scratch Team choose to forgo this solution and instead implement a comprehensive system of peer review mechanisms that ensure inappropriate content on the site will be found promptly and removed.

Flagging Comments

Each comment on the site is accompanied by a link that allows users to flag the comment if they find it offensive. Once a user flags a comment, that comment is hidden to that user. Furthermore, if many users flag a comment, the comment is automatically deleted. An email notification is then sent to the site administrators who retain the option to reinstate the deleted comment. This mechanism ensures that offensive comments are removed from the site without constant intervention from site administrators.

Flagging Tags

Tags are also accompanied by a link that allows users to flag the tag. When many users flag a tag, the tag is removed from the site. Unlike comments, flagged tags are still visible to the flagging user unless enough flags have been added to remove the tag entirely.

5.8 Safe Site

The peer review system of ScratchR uses the power of the community to create a scalable solution that prevents inappropriate content from causing disruption to a large number of users; however, the peer review solution exposes a number of users to each piece of inappropriate content before it can be flagged and removed. Even this brief exposure to a small subset of the Scratch online community can cause significant problems especially for the more conservative users of the site. Various teachers and parents have raised concerns over the effectiveness of the peer review system since it does not guarantee a completely safe site that only contains filtered content.

To address this issue, I implemented a safe site that contains only reviewed projects and galleries. The safe site takes advantage of the administrator review system that allows site administrators to review projects and galleries. Administrators have the option to mark projects and galleries as safe and unsafe. The safe site that is hosted at a different URL from the main site contains only projects and galleries that have been reviewed by site administrators. By using the safe site, teachers and parents do not have to worry about their kids being exposed to any negative material.

5.9 Administration Tools

Banning Users

Users who consistently display anti social behavior on the site through tactics such as harassment, spamming and posting inappropriate content have to be removed from the site to maintain a positive user experience. The User Banning administration tool allows administrators to ban users from using the site. Banned users are not allowed to use any site functions and are presented with a message stating the reason why they have been banned.

IP Blocking

Malicious users who are especially persistent will attempt to circumvent the user banning feature of the site by creating new accounts. To combat this problem, the IP blocking tool allows administrators to ban specific IPs from accessing the site. Since one single IP may host malicious users and legitimate users, this feature is only used in extreme circumstances.

User History Tracking

The administration backend contains comprehensive posting records of each user on the site. User histories provide information concerning flagged comments, flagged projects, censored projects, censored comments and much more. It allows administrators to accurately track problem users who violate the Terms of Use of ScratchR.

Admin Search

The administrator search provides site administrators an easy way to access the database records in the site pertaining to users, projects and galleries. Administrators can search users based on name, email and country. Similarly, administrators can search projects based on creator, name and the review status of the project. Effectively, the search allows administrators to browse relevant sections of the site database.

Announcement Tool

The announcement tool lets administrators change the current announcement displayed on the site. Up to three different announcements can be added in which case each announcement will have a chance of being displayed on each page view.

6 Analyses of Site Upgrades

Since many sections of the site have been upgraded simultaneously, it is difficult to obtain an exact measure of the impact that each feature and bug fix has on the overall user experience. ScratchR does not use ratings based user feedback; therefore, we can only measure effectiveness through subjective measures such as the number of user complaints on the forums and feedback from members of the Scratch Team. Some feature upgrades are isolated enough to have a measurable impact on user satisfaction based on usage numbers. For example, streamlining the gallery interface leads to an increase in the average number of projects per gallery since it is easier for users to add and remove their projects. Also, the effectiveness of a feature such as the comment reply system can be directly measured by the number of replies generated on the site. The following sections assess the impact of key site upgrades.

6.1 Gallery Upgrades

The effectiveness of the gallery upgrades can be subjectively measured by the number negative comments concerning gallery related functionality on the ScratchR forums and by feedback from the Scratch Team. After the bulk of the gallery upgrades were completed in 2007, there have been only a few complaints concerning the gallery functionality most of which were related to bugs in the permissions based access feature. The feedback from the Scratch Team has also been positive especially after May 2008 when the gallery system was declared “bug free”.

The number of galleries created since August 2007 and the average number of projects in each gallery give objective measures of the effectiveness of the gallery upgrades. Since August 2007, more than 10,000 active galleries (galleries which are not empty) have been created. Furthermore, the site now contains many galleries that have more than 1,000 projects. Compared with the top galleries in August 2007 which had around 100 projects, this increase reflects the effectiveness of the user interface upgrades concerning project addition and removal.

6.2 User Ignore-List

After the implementation of the Ignore-List feature, the reports of harassment received by the Scratch Team have decreased significantly. In fact, no complaints concerning harassment have been posted to the forums since the implementation of the Ignore-List. As of August 2008, 530 users have been ignored.

6.3 Comment Reply System

Since the implementation of the comment reply system, 138,822 project comment replies and 37,311 gallery comment replies have been posted to the site. Of the 500,000 comments posted since the implementation of the comment reply system, around 25% of the posted comments are comment replies. The comment reply system also appears to increase the rate at which comments have been posted to the site; more than 70% of all comments posted on the site were posted after the implementation of the comment reply system (the sharp increase in the number of users since March 2008 also contributes to this statistic).

6.4 Administration Tools

The administration tools help increase the efficiency and productivity of site administrators. Effectiveness of these tools can be examined through administrator feedback and through the number of user complaints. The site has grown to more than 140,000 registered users while the Scratch Team only dedicates two members to full time monitoring of the site. The number of malicious users on the site has also decreased significantly. Before the implementation of the IP Ban and the User Ban tools, we received multiple complaints per week concerning user harassment and cyber bullying. Now that 223 accounts have been banned and 58 IP addresses have been blocked, we only receive sporadic requests concerning malicious users on the site.

7 Development Principles

7.1 Database Design

Bugs stemming from faulty database tables and missing table columns composed of some of most difficult problems that I encountered while developing ScratchR. To minimize headaches associated with trying to retroactively repair a faulty database, I suggest the following principles of database design for a PHP driven application:

- Store IP addresses as BIGINT for ease of access and storage
- Do not use INT fields in storing a set of options because you will forget what the values represent, instead use ENUM. The savings in storage space in storing the values as INTS are usually not justified.
- Keep column additions to a minimum by planning ahead for future features. Adding columns causes faulty data to be stored for existing records.
- Always backup tables before making any database changes, because you don't want to lose precious user data.
- Conduct thorough testing for any piece of code that stores values into the database, because tracking down faulty values and repairing them after the fact is a nightmare.

7.2 Error Prevention and Testing

Handling coding errors through prevention and testing is a critical part of the development process. While developing ScratchR, I found that repairing bugs took upwards of 30% of the total development time. To help mitigate the amount of time spend in debugging and testing, I recommend the following principles:

- Release often. Bugs that slip by developers will often be found by users. Release often to take advantage of the user base especially for site wide upgrades and to reduce error propagation from one section of the source code to another.

- Batch related bugs. Fixing bugs related to a single controller or a group of related views will reduce the amount of testing needed and also speed up the development process. Furthermore, it is easier to focus on one section of the site at a time instead of attempting to fix multiple bugs occurring at disjoint sections of the code.
- Test major changes on all browsers and platforms. Browser dependant bugs difficult to debug; therefore, every major release should be tested thoroughly in all supported browsers.
- Balance testing and development especially if the development team is small. Some features cannot be thoroughly tested by one or two people; therefore, it is best to focus on repairing the bugs that users find instead of spending too much time on testing which gives diminishing returns. I recommend monitoring the deployment of these difficult to test features in order to repair bugs as they are found.
- Release large feature sets after peak time. Using this method, many features can receive significant testing and patches before bugs can affect a majority of the user base.

7.3 Agile Software Development and Individual Programmer Development

By leveraging the principles behind agile software development, we were able to develop and maintain ScratchR with a small team of developers. Agile software development principles support a short development cycle with many iterations and constant user feedback. I found that the mantra of "release early and release often" applied well to a majority of the features I've developed in the past year; however, this methodology requires working on many sections of the project in conjunction in order to meet the latest demands from the Scratch Team and Scratch online community.

From my experience, releasing features at a steady pace has to be balanced by focusing on development practices that help to increase the skills of the developer. I found that improvements in my skills came most readily when I was able to focus on one section of the source code and develop a "revamp" of an entire section of ScratchR. In this way, I forced myself to examine how the main components of the engine were designed. Many times, I found that the underlying design could be changed in order to resolve a large number of issues affecting one section of the site.

Ultimately, I believe that a balance has to be achieved when refactoring large scale projects. In conjunction to utilizing quick releases every two to four weeks, there must also be ongoing work behind the scenes in order to facilitate redesigns of outdated and problematic sections of the source code. If the later is not done, then major underlying problems in the design will be ignored in favor of hacked together patches that become increasingly difficult to implement.

8 Future Work

8.1 Karma System

As more users join the Scratch online community, the site will require additional administrators. In order to mitigate a large increase in the number of support personnel needed for the site, a system of self policing based on the notion of karma has been proposed by the Scratch Team.

In this scheme, each user action will cause a shift in that user's karma rating. Positive community building actions such as uploading constructive projects, creating useful galleries and posting helpful comments will increase a user's karma rating. Negative actions such as spamming, posting negative comments and uploading projects with inappropriate content will decrease a user's karma rating.

Users with high karma will be granted additional privileges on the site and will have a larger impact when they use the peer review system. In contrast, users on the low end of the karma spectrum will lose privileges such as posting comments and uploading projects. With a well implemented system, users will be encouraged to contribute positively to the site and refrain from violating the Terms of Use.

8.2 Sprite Sharing

Sprites constitute a large part of the Scratch experience because every project contains sprites. To improve the quality of Scratch projects found on ScratchR and to help users find relevant sprites for their projects, the Scratch Team plans to adapt ScratchR to allow for users to share their sprites online. Sprites sharing will share many similarities with project sharing including tagging, commenting and tags-based-browsing. Once this feature has been implemented, users will always have access to a large collection of readily available sprites to use in their Scratching endeavors.

8.3 Project Collaboration

Currently, users can only collaborate on projects offline since there is no online mechanism in the fashion of Base Camp. To increase collaboration through the sharing of Scratch code and ideas, a new section of the website dedicated to project collaboration will be created.

Project groups will function as central online locales for users to share sprites, code segments and discuss project ideas. Users will have the ability to create project groups and invite their friends to join. Our hope is to create a more dynamic environment where users who have specialized knowledge of certain areas of Scratch come together to share their expertise and combine their knowledge to create more comprehensive projects. Less experienced users will also have hands on opportunities to create larger projects which may lead to greater participation.

9 References

[1] Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. and Resnick, M. "Scratch: A Sneak Preview"; Second International Conference on Creating, Connecting, and Collaborating through Computing, 2004, Kyoto, Japan, pp. 104-109.

[2] Monroy-Hernández, A. and Resnick, M. "Empowering kids to create and share programmable media interactions". ACM, New York: 2008.

[3] Beck, K., Beedle, M., Kern, J., Thomas, D. and various others. "Manifesto for Agile Software Development". <http://agilemanifesto.org/>, 2001.

[4] "All news, videos and images". Digg. <http://digg.com/>, 2007-2008.

[5] "Broadcast yourself". YouTube. <http://youtube.com>, 2007-2008.

[6] "Inspired talks by the world's greatest thinkers and doers". TED. < http://www.ted.com/>, 2007-2008.