Teaching Introductory Programming as an Online Game

Ben Leong, Zi Han Koh, and Ali Razeen
Department of Computer Science
National University of Singapore

September 2011

ABSTRACT

This paper describes how we successfully built a "game layer" over a traditional Scheme-based introductory programming module, by adding a game storyline and introducing game mechanics like experience points and achievements as rewards for completing assignments. In the process, we also introduced a feed-based system to allow students to discuss their assignments with the teaching staff in a conversational manner, and a self-paced learning system called *Paths*, which allows students to do auto-graded exercises after each lecture. 71% of the students found that our new approach made the course much more interesting compared to "traditional" courses, and the average submission times for assignments improved from less than one day (15.5 hours) before the deadline, prior to the introduction of the game layer, to more than 2 days (51.2 hours) before the deadline.

1. INTRODUCTION

While most students are taught Java as their first programming language at our college, we also offer students the option of learning Scheme. Typically, about 50 to 60 of the incoming 400 freshmen for our faculty will choose to take the Scheme module CS1101S, which follows closely to the MIT Scheme course based on the SICP [1]. The relatively small number of students enrolled in the course provides us with an opportunity to experiment with new ways of teaching introductory programming at the freshmen level.

In this paper, we describe how we managed to improve the engagement of our students by implementing a game layer, which motivates them to work consistently and allows us to quickly identify gaps in their understanding of the material, in Fall 2010. The course is being taught in Fall 2011 (in progress) with an improved version of the game system. The inspiration of our approach comes from the TED Talk given by Jane McGonigal [8], where she suggested that game mechanics can allow us to better engage the new generation of students who have grown up playing games.

To create this system, which we call *JFDI Academy*, the seven problem sets for the module were broken up into 22 "missions" which were framed around a storyline in a "Star Wars"-based universe. Students were required to complete these missions as assignments. Experience points were awarded for the completion of these assignments and also for (i) attempting bonus questions (called *side quests*), (ii) attending lectures, recitations and tutorials, and (iii) participating in the course online forum and online surveys. The general idea is to reward students for positive behaviors. Once a student gains sufficient experience points, he will "level up" and this is reflected in a leaderboard. The system is also linked to Facebook via Facebook Connect and feeds are posted onto Facebook when students level up or unlock certain achievements in the game. The final level attained by the students is used as the continual assessment grade for the course.

In addition to the game system, we introduced two features to improve the pedagogical process. First, we implemented a feed-based system to allow students to discuss their assignments with the teaching staff in a conversational manner similar to the comments on Facebook feeds. This helps them receive timely and meaningful feedback about their work. Second, we added a self-paced learning system called *Paths*, which allows students to attempt a set of auto-graded exercises after each lecture. Not only does this help students reinforce concepts taught during the lecture, it also provides the teaching staff with an overview on how well the students are doing and highlights the concepts that are unclear to the students.

We evaluated the effectiveness of the system with surveys and also compared the behavior of the students under the new game system to that of previous cohorts. A majority (71%) of the students found that it made the course much more interesting compared to "traditional" courses. The game layer was also effective in encouraging students to submit their assignments earlier and average submission times were improved from less than one day (15.5 hours) before the deadline to more than 2 days (51.2 hours) before the deadline.

The key difference between our work and previous approaches of using games to teach introductory programming is that the game system is simply a layer on top of the course. Students do not implement a game using Scheme nor do they learn Computer Science concepts using games. The SICP text remains unadulterated and students continue using Scheme to learn and work on their assignments. It is possible for a student to complete the course and ignore the game component.

2. COURSE OVERVIEW

Before we proceed to describe the innovations that we introduced, it would be appropriate to first describe and explain the organization and conduct of CS1101S. CS1101S is offered only in the Fall semester of each academic year.

The course is taught over a period of 13 weeks with a one-week midterm break after Week 6. An overview of the topics covered in the course is shown in Table 1. Only 10 weeks are spent covering content from the SICP [1]. The last three weeks are spent introducing the students to Java in preparation for the course on data structures they would take in the subsequent semester, which is taught in Java. The focus of the Java component is mostly on the syntax and ensuring that they can successful "port" their solutions for problems in Scheme to Java.

Students have 3 hours of lecture weekly, consisting of a 2-

Table 1: Topics Covered

Week(s)	Topic	SICP
1–2	Procedural Abstraction & Recursion	§1.1–1.2
3	Higher-Order Procedures	§1.3
4-5	Data Abstraction	§2.1–2.3
6	Generic Operators	§2.4–2.5
7	Midterm Exam	
8	State & Object-Based Abstractions	§3.1–3.3
9	Memoization & Dynamic Programming	§3.5
10	Streams & Lazy Evaluation	§4.2
11	Practical Exam	
12-13	Java Programming	-

hour lecture on Wednesday and a 1-hour lecture on Friday. They also attend a 1-hour weekly recitation taught by either the lecturer or a senior teaching assistant, where they are shown how the concepts covered in lecture can be applied to solve relatively simple problems. Finally, they attend a two-hour tutorial weekly to discuss more complex problems in small groups of 6 to 7 students led by an undergraduate tutor.

Traditionally, students have to complete 7 problem sets which constitute 30% of the final grade. They also have to take a midterm exam, a practical exam and a final exam, which together constitute 60% of the final grade. The remaining 10% is awarded based on class participation. Because CS1101S is not the mainstream introductory programming module, but an accelerated elective module that is typically taken by stronger and more motivated students, the students are not graded on a curve.

Students can get help by attending office hours or, more commonly, by posting questions in the course discussion forum, which tends to be quite active. Depending on the semester, there can be up to 3,000 posts over a 13-week semester. A large number of past year exams, together with their solutions, are also made available to the students online.

2.1 New Game System

The key innovation that we made to the course in Fall 2010 is the introduction of a game system to replace the 7 problem sets, that were typically issued in the past years, with 22 "missions" (See Table 2). One problem that we previously observed was that students tended to procrastinate and some students started on their problem sets one or two days before the deadline. Because students typically took between 10 to 30 hours to finish each problem set, starting late was often a bad idea for many of them. By dividing each problem set into 3 or 4 smaller "bite-sized" missions and having them due more frequently, it helps students to manage their time.

The experience points that students are awarded for each mission depends on the correctness of the answer submitted. In this way, the game system also introduces an element of peer pressure and competition. Many students play online games and are used to the idea of a leaderboard and achievements. By casting the assignment into the form of an online game, many students were encouraged to submit

Table 2: Missions

Week(s)	Missions	Topic
1–2	1–3	Procedural Abstraction
3–4	4–7	Higher-Order Procedures
5–6	8–11	Data Abstraction
7	12–15	Generic Operators
8–9	16	Lego Mindstorm Robot
10–11	17–19	Object-Oriented Programming
12–13	20–22	Streams

their solutions earlier so as to get onto the leaderboard. The leaderboard only shows the levels of the top 15 players. This is to avoid embarrassing the weaker students at the bottom.

While we had optional problems in the previous problem sets, many of them were often ignored. Under the new game system, these optional problems are called *side quests*. One goal of the side quests was to reduce the pressure on the students to get their answers completely right in the main missions by providing them with a way to make up for the lost experience points. Another goal is to provide students with opportunities for additional practice while keeping the additional work optional.

Timely feedback is very important for effective learning. We have two mechanisms in place to improve the feedback loop. First, while assignments were typically graded as a batch only after the due date, we do it differently under the new game system. Individual assignments can be graded as soon as they are submitted, and our policy is to have all assignments graded within 24 hours after submission. Students submit their assignments online and their tutors are immediately notified via email. Once an assignment is graded, the students will be notified via emails and also via in-game notifications. Next, each mission page also includes a section where the students can interact with their tutors by posting comments like those on Facebook feeds to seek clarification and discuss their assignments with the teaching staff.

2.2 Paths

To reinforce the concepts taught during lectures, a student has to practise and apply the concepts. While students are expected to do this in their assignments and tutorials, the assignments and tutorials tend to lag behind lectures by more than a week. If the student only starts to apply the concepts taught in lecture after a long delay, more effort is required on the part of the students to recall what was taught. Students also often have difficulty identifying the key concepts taught in the lecture, especially if several concepts are presented in the same lecture and in different examples. To address these issues, we implemented a self-assessment system called *Paths*.

Paths string together a series of questions, which we call steps, that students must complete in sequence so as to complete a path. The steps in a path may either be multiple-choice questions (MCQ), or more general coding questions where automated public and private test cases are run to verify the students' answers. The MCQ steps provide students with feedback when the wrong choices are made, and students are expected to keep attempting the question until they get it right. Coding steps, on the other hand, are opportunities for students to practise writing code. We display

the results of running the submitted solutions on public test cases to help students check their answers.

The path engine described above supports two different kinds of paths: (i) post-lecture paths and (ii) post-tutorial paths. Post-lecture paths consists of 8 to 10 steps that provide ample guidance and let students review concepts and any new syntax covered in lectures. These paths are meant to be completed within 2 days after the lecture, when the content is still fresh in the students' minds. To encourage students to attempt these paths promptly, we award bonus experience points for completing a post-lecture path early.

Post-tutorial paths are made available only after the associated assignment and tutorials are over. These paths test mastery of the material and little guidance is provided. As these paths are optional, students only unlock achievements for completing them and are not awarded any experience points for them. Despite the lack of direct incentives, over 90% of the students have attempted and completed post-tutorial paths.

3. IMPLEMENTATION

It was decided that the story would be a scifi adventure set in a "Star Wars"-like universe, narrated in a third-person perspective. Converting the problem sets into missions that fit into the plot points of the story was not as challenging as expected, since even in past years, each problem set usually had its own story to make it more interesting. For example, the third problem set where students learn about data abstraction had a story on cracking the RSA encryption system of an antagonist. This problem set was naturally split into Missions 8 to 11, where students hack the system of the villain in our story.

The JFDI Academy website was implemented with a PHP/MySQL backend on an Apache webserver and uses Javascript to support the interactive features. The comics were drawn by a freelance artist and the final comic consisted of 30 comic strips, with an average of 20 panels per strip.

3.1 Brief Synopsis of Game Storyline

Students follow the adventures of Lea, Kaz, and Zed, three initiates who have entered the JFDI academy to learn the ways of the "Force". While training under the tutelage of the Grandmaster, the three initiates uncover a plot by Darth, the evil Dark Lord, to destroy the academy. After frantically warning the Grandmaster, they hack the encryption system of the Dark forces, and delay Darth's plans by sending him false messages.

Meanwhile, the Grandmaster calls a council meeting with other Grandmasters to discuss the threat. After much deliberation, they decide to launch a counter-attack on the *Death Cube*, the primary battle station of the Dark forces commanded by Darth himself. Lea is first tasked with disabling the powerful shield that is protecting the Death Cube. Leading the initiates, she infiltrates the Death Cube and overloads the generator powering the shield. With the shield inactive, the council commands a fleet of attack ships to concentrate fire on the Death Cube and destroys it. However, Darth narrowly escapes the attempt on his life and goes into hiding.

The Grandmaster, sensing greater challenges ahead, teaches his initiates his ultimate skill and disappears into the Force. He leaves them a set of coordinates to find the next Grandmaster. Sad about the Grandmaster's disappearance, the initiates journey to the given coordinates, only to find that it leads to a middle of a lake. Seeing their own reflections on the lake, they realize that they have become to their own Grandmasters to master the Force and defeat Darth.

Figure 1: Screenshot of home page (student view).

3.2 User Interface

When a student logs in to the game website, he will see a home page similar to that shown in Figure 1. On the left, he can quickly see his current level, the experience points he needs to progress to the next level, and important notifications. The navigation bar on the top lets him access other portions of the website, such as the missions. General class announcements are shown in the center and there is also a "Notable Happenings" section on the bottom, which contains notable student activity (such as when someone is first to submit a mission).

A screenshot of the staff view is shown in Figure 2. As with the student view, there is a notification panel on the left that alerts the tutor that his students have either submitted an assignment or have left comments on the feed system. The staff view additionally contains a navigational bar that allows the teaching staff to access the administrative functions of the website.

On both the student and staff views, there is partially occluded comic section in the center. When clicked, this section expands into a full-sized frame like the one shown in Figure 3. Each panel within a comic strip is a separate image and the user can scroll through the comic. This approach allows us to simulate animation in the comic without having to use Flash (or other full-featured animation tools). Students typically have access to a limited number of panels. More panels are unlocked as missions are completed. In this way, the comics also serve as a form of reward for completing missions.

In Figure 4, we show a mission page where the student has already submitted his work. The details and files of

Figure 2: Screenshot of home page (staff view).

the mission are located on top while the student's code is placed in the center. Notice that the tutor has added an annotation to line 52 of the student's code. The feed system at the bottom is then used by both the tutor and his student to discuss the highlighted mistake.

The progress of the students in completing the paths is accessible in the "Path Statistics" view shown in Figure 5. The history of all the answers submitted to the system (including wrong ones) is captured and this allows the teaching staff to see the common mistakes that are made by the students and to intervene when a student is "stuck."

Achievements are unlocked after completing certain missions, side quests, and for accomplishments such as receiving perfect grades for six missions in a row. The achievement page is shown in Figure 6. Each achievement is also associated with a number of achievement points. These are separate from experience points and are not used in computation of students' grades. They are used to rank the students in the leaderboard.

3.3 Facebook Integration

When students level up or unlock an achievement, they have the option of publishing a feed to their Facebook account. Figure 7 shows two examples of feeds published by our students. These feeds let students show off their learning progress to their friends. We believe that this generates a "feel good" effect and provides positive feedback to the students.

4. EVALUATION

To evaluate the effectiveness of the new game system, we conducted a brief survey in the middle of the semester and a more detailed survey at the end of the semester. We also compared some metrics recorded during the semester to metrics for the course when it was taught in previous years.

The following are some brief findings from our surveys:

Figure 3: Screenshot of sample comic panel.

- **Game System**. 76% of the 51 students who responded found the game system to be helpful to their learning. Most of the students said that the game system made learning more interesting. They also commented that the system helped to promote sustained and continuous learning throughout the semester. Some students appreciated doing the assignments in "small chunks" as missions instead of large infrequent problem sets.

- **24-hour Grading**. 64% said that the 24-hour grading was helpful to their learning. Most of the students remarked that it was extremely helpful to be provided with timely feedback. A small number remarked that a grading turnaround time of 48 to 72 hours would have been good enough.

- **Improved Interactions**. 82% agreed that mission feeds helped to improve their interactions with the teaching staff. A small number however did not use the mission feeds to discuss assignments with the teaching staff and preferred to discuss the assignments offline with their peers.

- **Game Elements Improved Motivation**. When asked which features of the game system encouraged them to finish their assignments, 71% said that the concept of levelling up was helpful. 33% said that the leaderboard and achievements were important motivations. We suspect that these are the regular gamers among the students.

- **Path System is Effective in Reinforcing Lectures**. The students unanimously agree that paths are helpful and add value to their learning, with 55% of students strongly agreeing so. Likewise, all students unanimously agree that that lecture paths help to reinforce the concepts taught during lectures, with a larger majority (61%) strongly agreeing so. 78% felt that other modules should also adopt post-lecture paths even though this would increase their workload.

The assignments for CS1101S have always been submitted online. When compared to past years, the students tended

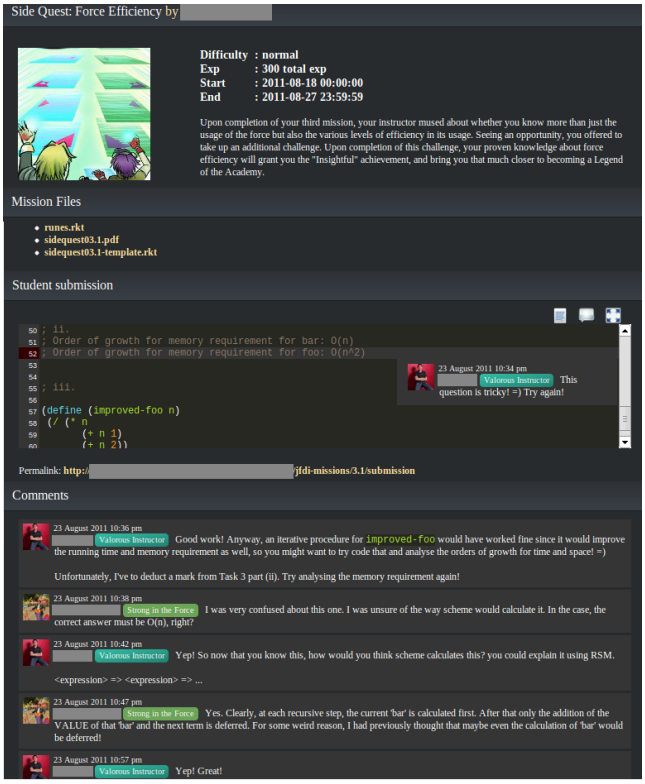

Figure 4: Screenshot of a mission feed.

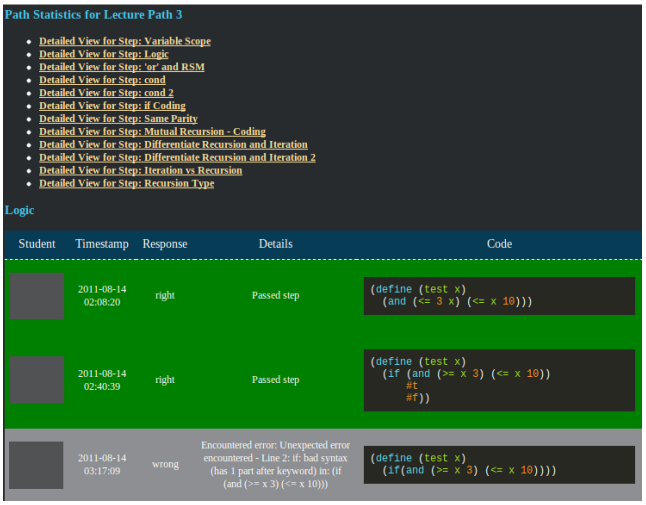

Figure 5: Screenshot of path statistics.

to submit their assignments much earlier after the introduction of the game system. The average assignment submission time was improved drastically, from less than one day (15.5 hours) before the deadline to more than 2 days (51.2 hours) before the deadline.

Our college conducts a standard teaching survey (on a 5-point scale) at the end of each semester. The module and teacher ratings for CS1101S over the past 4 years is shown in Figure 8. The course was taught by the same instructor for all 4 years. It seems that the introduction of the game system in Fall 2010 had a positive impact on the module rating. It however had no noticeable impact on the teacher rating.

The course is not graded on a curve. Students are graded based on their demonstrated ability and effort is made to ensure that the same standards are preserved across years. From the the distribution of the final grades of the students over the past 4 years, we found that the students seem to perform marginally better last academic year (Fall 2010) compared to previous years. This suggests that the game system is helpful in improving the effectiveness of our teaching, though we would have to admit that this data is not entirely conclusive since we are not able to control for the intrinsic aptitude of the students between cohorts.

4.1 Drawbacks

A drawback with our approach is that some students feel more stressed as they feel they have a lot of deadlines to meet and a lot of assignments to submit, even though they have a similar amount of work as students who took the course in previous years. This perception problem might overwhelm some students. In our course, we address this by encouraging students to start and submit assignments early.

Another drawback with our approach is that there is a very large upfront cost of implementing the system, including preparing the storyline, comics, missions and side quests. However, this is a one-time cost as the system can be reused in subsequent semesters.

While our approach works very effectively for CS1101S, which typically only has 50 to 60 students per semester, it is not entirely clear if this approach is scalable for a large course with hundreds of students because the system requires a lot of manpower to execute. We suspect it would be feasible if we maintained the current student-to-tutor ratio so that it remains small. We leave it as future work to explore the issue of scalability.

5. RELATED WORK

The game layer was introduced to address the problem of procrastination in the completion of assignments. We also wanted to better engage students in their learning [5]. Our path system is similar to Singpath [2], an online system that was developed to teach Python programming in a self-directed way.

Existing approaches in using games to teach introductory computer programming revolve around delivering the material using games [6, 4], or by teaching students to develop games using tools such as Flash and Alice [7, 3]. Our game system differs from these approaches in that the technical material is not taught via a game. Rather, the game layer was designed to improve student engagement and motivation for learning. Under the new game system, students have numerous opportunities to reinforce their understanding, and they receive quick and detailed feedback on their assignments. Nevertheless, like in previous years, students continue to work in Scheme on their assignments and to learn concepts such as recursion.

6. CONCLUSION

In this paper, we described how we successfully implemented an approach to “gamify” a traditional introductory programming module. Many students appreciate this new

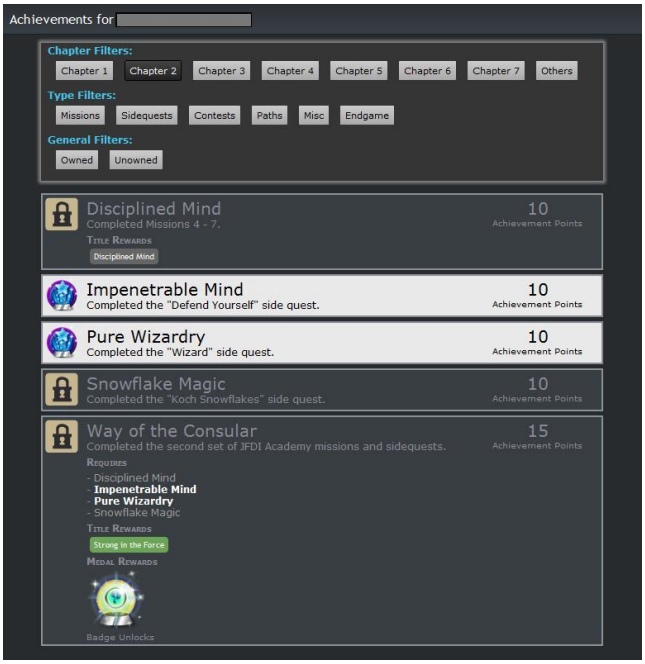

Figure 6: Screenshot of sample achievements.

Figure 7: Sample Facebook feeds.

approach and found that it made learning more interesting, but a small number did not find it helpful. Our approach is interesting because such students can effectively ignore the game component and still complete the course successfully. While we have found that the game layer can significantly improve student engagement, we would like to highlight that teaching is still fundamentally a human activity. Effort still needs to put into traditional modes of instruction like lectures and tutorials for our approach to be effective.

7. REFERENCES

[1] H. Abelson and G. J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, Mass., USA, 2nd edition, 1996.

[2] C. Boesch. Singpath. http://www.singpath.com.

[3] S. Cooper, W. Dann, and R. Pausch. Teaching objects-first in introductory computer science. In *Proceedings of SIGCSE '03*, February 2003.

[4] M. Eagle and T. Barnes. Experimental evaluation of an educational game for improved learning in introductory computing. In *Proceedings of SIGCSE '09*, March 2009.

[5] P. Eggen and D. P. Kauchak. *Educational Psychology: Windows on Classrooms*. Prentice Hall, 8th edition, 2009.

[6] A. Hicks. Towards social gaming for improving game-based computer science education. In *Proceedings of FDG '10*, June 2010.

[7] S. Leutenegger and J. Edgington. A games first approach to teaching introductory programming. In *Proceedings of SIGCSE '07*, March 2007.

[8] J. McGonigal. Gaming can make a better world, March 2010. TED.com, http://blog.ted.com/2010/03/17/gaming_can_make/.

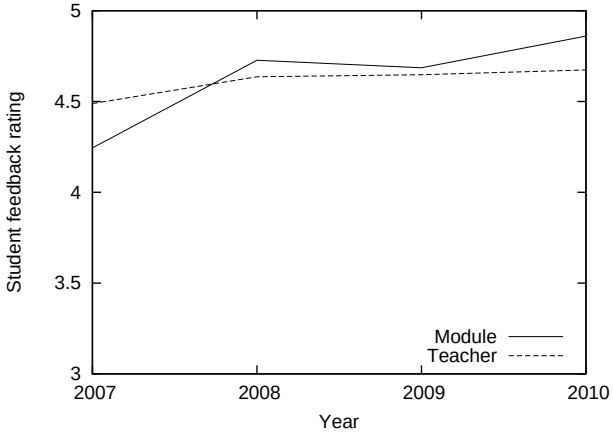

Figure 8: End-of-semester course feedback ratings.