September 2014

Flipping The Classroom in CS 1101

Alexander W. Witt
Worcester Polytechnic Institute

Daniel Igor Gendin
Worcester Polytechnic Institute

Jeremy Vincent Macaluso
Worcester Polytechnic Institute

Long Hoang Nguyen Duc
Worcester Polytechnic Institute

Patrick Lenti Boudreau
Worcester Polytechnic Institute

See next page for additional authors

Author
Alexander W. Witt, Daniel Igor Gendin, Jeremy Vincent Macaluso, Long Hoang Nguyen Duc, Patrick Lenti Boudreau, Ryan James Melville, and Xiaosong Wen

Project Number: IQP-JB7 FLIPPING THE CLASSROOM IN CS1101

Improving Intro to Programming with Educational Technology

An Interactive Qualifying Project
submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
by

Patrick L. Boudreau

Daniel I. Gendin

Jeremy V. Macaluso

Ryan J. Melville

Long Nguyen Duc Hoang

Xiaosong Wen

Alexander W. Witt

Approved:

Professor Joseph E. Beck, IQP Project Advisor

Abstract:

The objective of this project was to design a variety of supplemental materials for use in introductory computer science courses at the undergraduate level. The supplemental content that was produced was based primarily on recorded and annotated lecture content retrieved from Gregor Kiczales's online course titled, Introduction to Systematic Program Design. From the fundamental programming concepts covered therein, the research produced here attempts to describe how variations in the presentation of such content influences the manner in which students are collectively able to learn and interact with abstract concepts taught in the classroom. Note, however, that this research is not intended to be a comprehensive study of all available technologies or methods of approach with respect to instilling good design practices and analytical techniques. Instead, the primary intent of this research is as an exploration of two different approaches that were continually adjusted in order to address both student preference and learning style. Additionally, the research was conducted under a set of assumptions that were both inherent to the two classes that were observed and to the previous experiences of the individual investigators involved in the study. Consequently, this research will effectively serve as a reference for other studies that either attempt to create better supplemental materials for introductory courses in programming or examine collective student response to certain methods for learning.

Contents

Table of Figures

Introduction

It used to be that taking a class meant sitting in a lecture hall, listening to a professor give a lecture and then going home reviewing the notes taken in class. Taking a class with a specific professor used to mean being enrolled as a student at the university where the professor is employed. This is not always the case anymore. The ubiquity of the internet has made it possible for a student to study a subject from the comforts of their bedroom. This model of learning is called a MOOC (Massive Open Online Course). All a student has to do is go to a video streaming website, like YouTube, and he or she can listen to lectures from leading professors in any industry from almost any university, no matter the students' geographical location, financial situation, or scores on standardized tests. While simply watching lectures on YouTube is a common act, successfully locating content that both adequately deconstructs the considerations for program design and involves students to a point where they are capable of utilizing the analytical skills that are discussed is a rather arduous process. Typically the video content available on the web explicitly concerns itself with determining the solution to specific problems without providing a student with "the bigger picture" which is often necessary for understanding of a topic. Even if a student is able to find a video that provides a good overview of a general topic the student is likely to still have problems with long term retention. A student simply watching videos in bed is unlikely to have a good grasp of the material because the student is missing a method to test his or her comprehension, and the student is also missing a way to practice the material he or she has learned. Just like it is impossible to learn to swim without getting in the water, it is impossible to learn to program without writing code.

To deal with the problems mentioned, alternative forms of presentation for lectures on program design, such as Khan Academy or Coursera, already exist. These MOOCs attempt to

solve the problem of comprehension by providing the student with supplemental material at the end of the lecture. This material allows the student to test his or her comprehension and practice the basic concepts presented in the lectures. However, the manner in which the content is presented does not lend itself well to long-term retention either. The main problem that these comprehension questions exhibit is a lack of balance, some of the questions are too easy, and can be answered through simple logical reasoning by someone who never even watched the related lecture, other questions are so complex that even a student who is experienced with the material being presented would have trouble, thus causing the student who is just learning the material to be completely lost. Very few modern MOOCs have questions that are well balanced in difficulty, and are thus accessible to a beginner without being so simplistic as to encourage not paying close attention to the lectures. Thus we decided to set out to create such problem sets. The software we used to create problem sets is called Assistments; it was developed at WPI in partnership with Carnegie Mellon with the specific purpose of developing supplemental materials online, for courses.

The IQP Team was tasked with creating content to assist with the understanding of video lectures in CS1101 (Introduction to Computer Science). This course is unique from most of the other computer science courses offered at WPI in that it is taken by a large number of students who are not majoring in computer science. Content for these students differs from all other content for computer science courses in that there is no assumed background knowledge and the emphasis is placed on broad concepts rather than specific technicalities. Also, because CS1101 is often a large lecture, usually around 120 students are registered for it; the average student does not receive much individual attention with regard to determining his or her specific issues and trouble spots. Our approach to presenting material had to fit the unique nature of CS1101.

How best to teach program design is not a simple question. Designing a good program is a difficult process because the idea of what a good program looks like is very abstract. The most effective approaches to teaching design analysis and creation appears to be structured in a manner that serves to show the student how fundamental units of code (otherwise known as primitives) can be used to manipulate data. After understanding how those primitives individually manipulate data, the notion of constructing a pattern of primitives whereby data may be manipulated for a particular end result is reinforced.

Another widely utilized technique for teaching design concepts and analysis follows from first enabling students to understand primitives and then permitting them to investigate the intended behavior to which a particular design must adhere. By explicitly outlining the expected behavior of a pattern by use of a comprehensive suite of test cases, students are able to understand the conditions that a program must address through the course of its operation. Advance knowledge of these conditions fosters an environment in which students can construct potential implementations in a piece-wise manner and readily observe whether or not the code that they have currently developed accords to the constraints imposed by the individual test cases. The merits of this method are that it enforces verification through testing, teaches the practice of reverse engineering, and outlines potential areas of concern for detecting errors in program logic. Additionally, the method enables students to initially perceive the program that they are designing as a “black-box” (i.e. an abstract machine that reveals nothing about its inner workings other than its behavior when it acts upon input). Consequently, students come to understand the process of design as the location of a solution that is capable of mapping a set of input to a corresponding set of output within a design space, rather than the location of a single, absolute solution.

Additional techniques that appear to enhance the retention of design concepts and analysis among students appear to be the use of templates and the use of visual imagery. Templates are useful because they provide a means by which functionality can be understood incrementally. By being introduced to a small number of templates that describe the manipulation of basic data types, students can come to understand how those templates may be modified for other applications. Similarly, templates are useful in that they can be used to reaffirm the direct correspondence between test cases and the manner in which a particular program is designed. For instance, if a test case is produced in order to account for a condition where a particular article of data is null, then the test for that condition must be present in the code that appears in the template. Yet another benefit of learning design through templates is that the relationships between program data can be explicitly considered and understood. For instance, when students design code that uses more sophisticated algorithms such as mutual recursion, the notion that there are two data types that communicate with each other is made clear.

Aside from templates, providing visual imagery is another technique that is used to teach program design to students. Visual imagery is effective in that it can serve to associate written code with a mental picture that describes the operation of that code. Furthermore, making that imagery dynamic enhances the association between the code and the mental picture that students form because the code that is being viewed is dynamic when it is being used. By being acquainted with code segments in a visual way, students can come to more clearly understand how to traverse different types of data, how to determine what kind of conditions must be considered when processing a particular type of data, and how to detect and resolve errors in program logic such as infinite loops prior to running a program. Overall, visual imagery helps

students to conceptually trace and analyze the logic of the program that they are attempting to produce. Another manner in which visual imagery assists students in learning the abstract nuances of program design is that it provides a means by which students can begin to break down the description of a program into individual steps where a particular type of processing occurs. This type of deconstruction ultimately benefits the student by enabling him or her to both immediately describes his or her code to others and translates it into the final result.

While these techniques individually enable students to acquire valuable insight into the fundamental building blocks of program design, the manner in which they are used in the classroom and the manner in which students utilize them ultimately determine how effective they are in practice. For instance, when exposing students to the exercise of designing programs from expected behavior, it often helps to explicitly show students how to produce test cases and how to map those test cases to the program that they serve to validate. It also is beneficial to indicate how test cases can be effectively used to isolate errors in program logic. In this way students not only understand how to develop programs but also how to debug them when they are incorrectly composed. Similarly, when using templates to assist students in understanding program design, it is beneficial to describe how each template works and how it can be used generically to produce a program. Providing analytical problems in which students must extend the templates that they have been exposed to is also a beneficial practice. With respect to visual imagery, it often tends to be applied effectively as a means of describing the operation of programs, the traversal of data types, the differences among data types, and the communication between data in different types of algorithms.

Aside from the way in which the techniques are applied in the classroom, the manner in which students come to use them is also another factor that greatly influences their effectiveness.

For instance, in some classroom environments, students will work collaboratively in order to solve problems that are more analytical. This approach tends to work well with these types of problems because students can compare ideas and navigate the design space together while collaboratively overcoming obstacles. In other classroom environments, the individual approach is taken and students learn to address the same obstacles on their own. Ultimately, factors such as the rigor of the material, the format of the material, the competence of students when they are grouped, and the number of students assigned to a group influence the degree to which a student is able to retain and participate in learning the content presented in an introductory course on program design.

This project in particular is targeted toward understanding how supplemental lecture content is received by students when the factors of collaboration, group size, material format, material presentation, and material difficulty are altered. The two types of presentation that we developed included review sessions and the tutoring software Assistments. During the review sessions we adjusted and observed the factors of collaboration and group size. Additionally, we altered the format of our material both during the review sessions that took place as well as the tutoring problem sets that were assigned. Afterward, we were able to draw conclusions on the basis of the feedback that we received from the students who were involved, and we gained insight on how to proceed in developing supplemental material for future classes.

Preliminary Findings:

Before settling on our final project idea we first tried several other methodologies using the Assistments software and having to do with the integrating online learning and the material of CS 1101. Several different approaches were tried, these ranged from in-classroom activities to

review sessions to simply providing extra practice. While each of these approaches had their merits, we believe that none of them warranted full implementation.

Our first approach was to focus on the in-class environment and have the students work through intricate design problems in groups. In order to test our methodology we held meetings with the students in the evening. At the first meeting we attempted to give a single problem for the students to work through. The problem is provided below in Figure 1

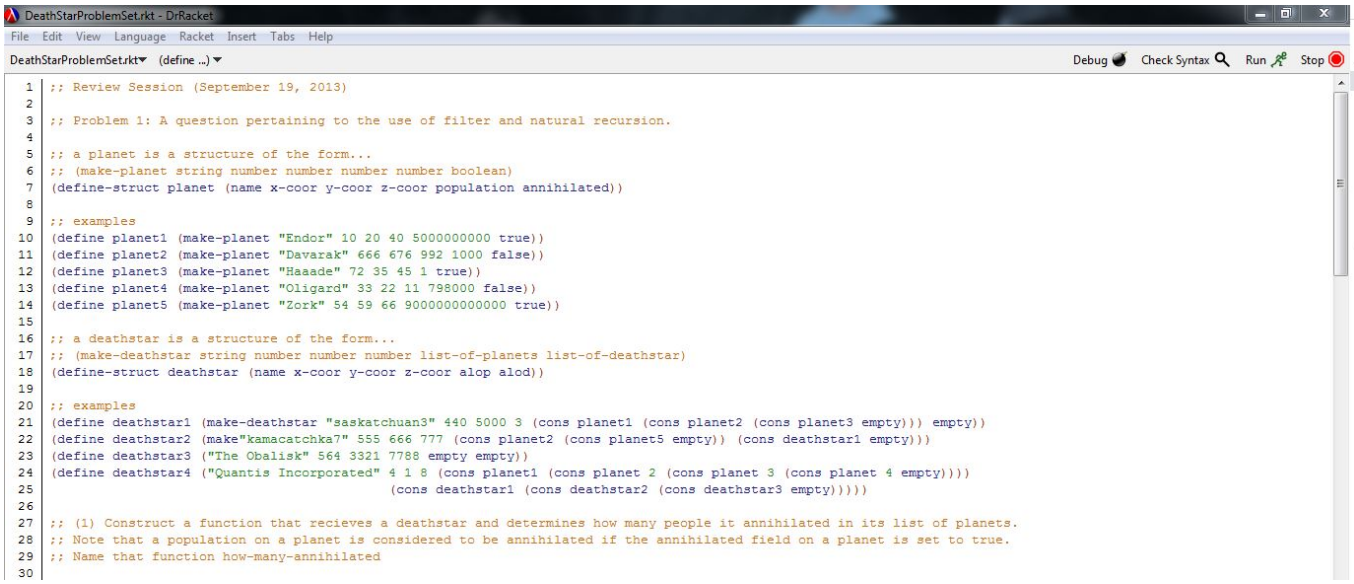


Figure 1: Review Session 1: Death Star Problem

While this problem covers many important topics, which the students should be comfortable with, we encountered several issues when we gave it to students. Because only 3 students showed up to the meeting we only had one group of students working on the problem, also, even though they were encouraged to work together, the students were reluctant to do so. This reluctance was likely due to the fact that the students were not familiar with each other, and because there were so few of them present everyone was too embarrassed to say that they were having trouble. Thus every student attempted to solve the problem individually, and the scope

and size of the problem did not lend itself to an individual solution in the amount of time the students were given. From our experience with that first meeting with the students we realized that we were not ready to implement anything during class. Thus we turned our attention to running effective review sessions.

In the meetings with students that followed, we gave the students several smaller conceptual problems rather than one large design problem. Some examples included a recursive algorithm for finding the factorial of a number, a function calculating the Fibonacci numbers, and writing a function that used Euclid's algorithm to determine the greatest common factor of two positive integers. While the students seemed to do better with these problems than they did with our question, problems still arose. The main such problem was that there was a noticeable disconnect between what the students were learning and what we were asking. Because our own academic schedules prevented from regularly attending the lectures and we were not always sure what had and had been covered in the videos, thus the questions we asked caused much confusion among the students.

One of the issues we had was that the terminology we used caused confusion among the students. For example, one of our questions asked the students to write a function that traversed a tree. However, the students were unfamiliar with the term traverse, and thus had difficulty with the question. This problem spoke to the wider issue of our questions being largely disconnected from the material being covered in the class as well as any shifts in the content being covered. Another issue was that we assumed the topics being covered were identical to those that we covered in CS1101, this assumption was of course false as the class in question was being taught by a new professor using a new format, thus we sometimes ended up asking questions on material that students never covered, for example, because accumulators were such a large part

of our introduction to programing, we assumed they were covered and asked a question on them, however, the students had never seen accumulators and thus were unable to solve the problem. This issue also stemmed from our disconnect from what the students were working on.

Realizing that our help sessions were not very effective and not very helpful to the majority of students, we decided to shift from preparing help sessions to creating general review content, to allow students to receive extra practice on topics that they were having trouble with, the software used to develop these review exercises was a web-based service called Assistments, which was developed at WPI for the specific purpose of creating online content to help students practice. Our decision to move away from focusing on help sessions is supported by the results of a survey we ran among the students of CS 1101 at the end of C-term of 2014. In the survey, out of the hundred students polled, only 18 said that they would have liked to attend review sessions in the evening. The full results for this survey question are presented in the figure below.

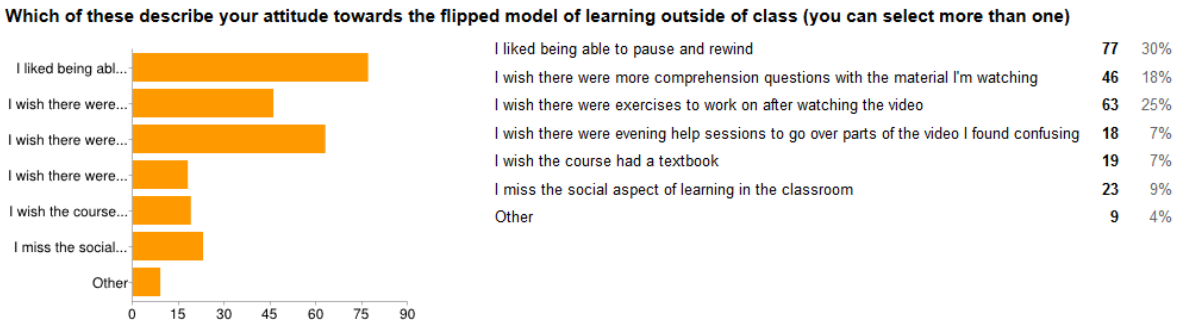


Figure 2: Students' attitudes towards the flipped model

The fact that less than a quarter of the students who were polled would have like review sessions confirms our belief that continuing to focus on review sessions would have been an ineffective use of our project.

While making review materials was the most effective of our initial methods (especially when compared to the ineffectiveness of our attempts to study group work), it was not without its flaws. We still had the problem of assuming that the materials were being presented in the same way that they were presented to us, when we took CS1101 previously. While this issue caused less trouble than it did during our initial help sessions it was still problematic. For example, one of our questions asked the student to write a function operating in a binary tree, however, the students had only seen arbitrary-arity trees and were unfamiliar with the idea of a binary tree. While the students were still able to do the question because the concepts from arbitrary-arity trees can be extended to binary trees, the inconsistency between our materials and the materials presented in the lecture caused noticeable confusion among students. Another problem we encountered was that entire topics were too large to effectively cover in a single set of questions. As a result, our questions were often too broad or too specific to provide the students with an effective review of the topic.

After exploring these methods, we decided that the most effective way to assist the students in understanding all the material would be to have review exercises for each individual lecture/ video, thus eliminating both the issue of topics being too large to cover in a single problem set, as well as the issue of disconnectedness form the lectures. Thus as a result our final method was a combination of the class by class approach from our early trials with the idea of individually completed review exercises, which was tested in our later trials.

First Iterations of Content

We tried several different versions of ASSISTments problem sets in A term and the first half of B term before we agreed on a standard format in the last half of B term.

In A term, we created problem sets which were not associated with a specific video but rather, every problem set had a topic, such as: General Recursion, Mutual Recursion, or the List template. There was no standard format for those problem sets, so they suffered from lack of focus and lack of relevance to what the students were doing in class. While the problem sets we developed in A-term were useful tools for gauging how students react to different kinds of problems and a good way to test what the Assistments environment was capable of, it was clear that the format of problem sets used in A term was suitable for review sessions, but was not suitable for long term implementation of supplementary material for the CS 1101 class.

In the beginning of B-term, it became clear that the main issue of the problem sets developed in A-term was a lack of focus. In order to fix this problem, we decided that there should be a problem set associated with every video lecture. This change made an immediate impact on the relevance and focus of our problem sets, however, problems still occurred. We initially simply included a link to the relevant video in the first question and having the students open the video in a different window and view the video in a separate window before returning to Assistments to complete the problem set. However, we quickly realized that this is not a good idea; the very act of switching between several windows made it much easier for a student to get distracted and open a non-related window. In order to decrease the possibility of the students getting distracted we decided to embed the relevant video directly into the problem set. A problem with an embedded video is presented in the figure below.

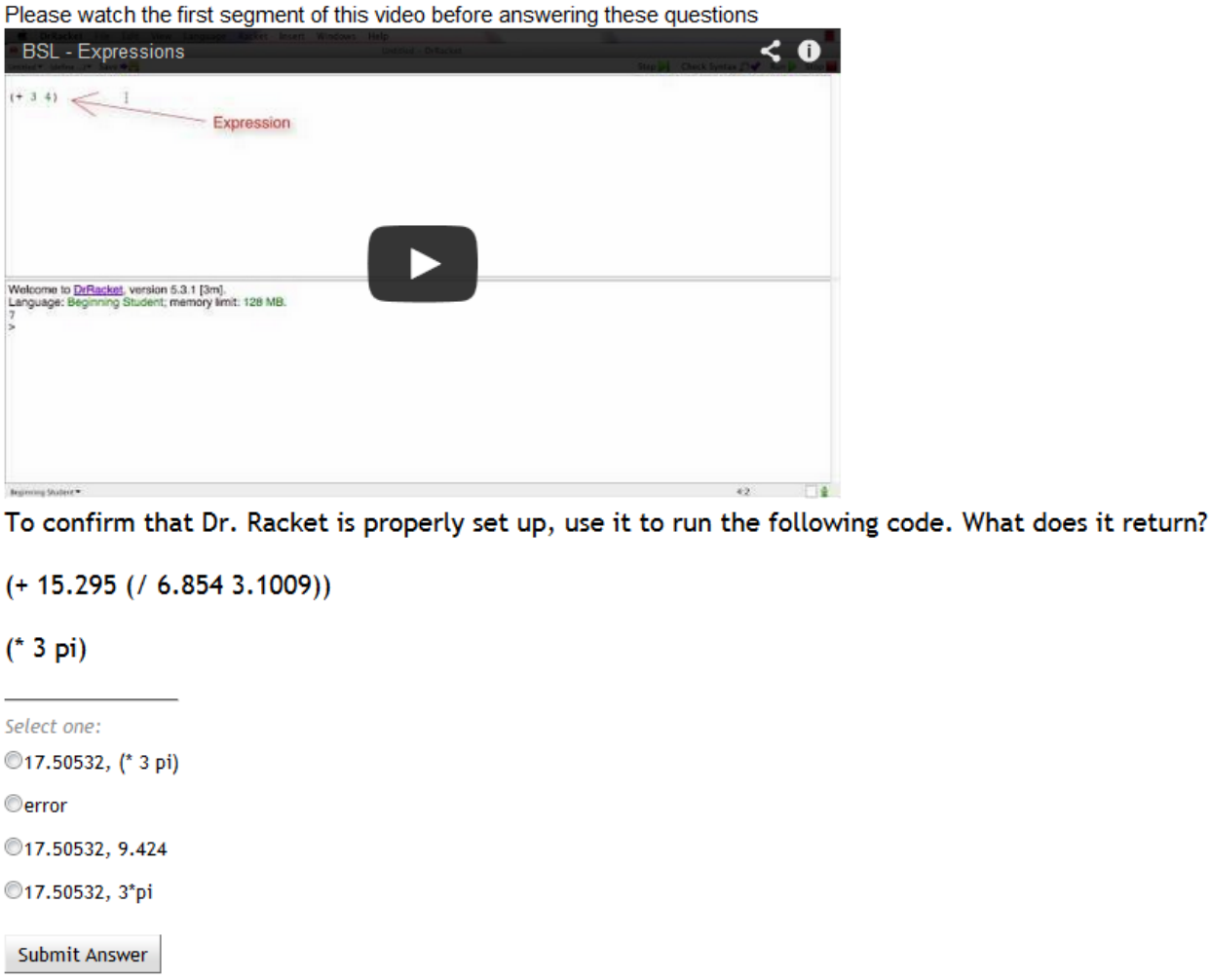


Figure 3: Problem With Embedded Video

After deciding that every problem set should contain an easily accessible version of the relevant video, we were faced with the question of whether some problem sets should be based on several related videos. While initially it seemed promising to group related videos into a single problem set, we decided against it. The main reason for not combining multiple videos into a single problem set was that it decreased the flexibility of a professor using our content. With, every problem set having a single video, a professor using the content we developed has complete control over what videos to assign and what order to assign them in, however if a problem set is based on several related videos, then the professor using our content would have

to assign those videos together on the same day rather than just having to assign them in a particular order. Thus the problem sets would dictate the scheduling of the course and we believed that it was best to leave the scheduling of the course up to the professor, and just have the problem sets as a resource to reinforce the concepts presented in a specific video. Another reason we decided that every video should have its own problem set was feedback from the students, who did not like several videos in a single problem set because it forced them to devote a single large chunk of their time to the assignment rather than providing them the flexibility of viewing the assignment in two smaller chunks.

Another issue we encountered had to do with the complexity of some of our problem sets. In several problem sets we tried to go in to depth in a topic that the video covered quickly or teach an idea that the video ignored. While this initially seemed like a good idea, it caused those problem sets to become very long and complicated. It soon became clear that the text in a problem was not an optimal way to teach a concept. We thus decided that a problem set should only provide practice and review for the topics in a specific video. If in the future it is believed that an important topic was skipped, a separate video should be made for that topic and a problem set can be made to go along with that video. However, it was definitively decided that a problem set should not include topics not mentioned in the video it accompanies.

The last issue we encountered was the issue of partitioning a video. There were two different opinions on how to approach video partitioning. One way, was to pause the video every 5 or so minutes (this time could vary depending on the pacing of the video) and ask a few questions after every section before proceeding to the next section of the video. The other approach to partitioning the video was to ask all the questions at the end. Have the students watch the entire video, and then proceed to answer all the questions. While both approaches

seemed to have their merits, with partitioned videos keeping the students attention by interspersing long videos with small tasks that forced the students to review what they had just seen, and not partitioned videos allowing students to understand the full context of the topic being presented before asking questions about it, thus decreasing the possibility that a student will be confused by the questions due to lack of familiarity with the topic. Overall though, we decided to partition most of the videos, this was partly due to the fact that we were wanted to make sure that students would not lose focus during longer videos and partly because students who prefer un-partitioned videos can easily turn off the partitioning and watch the video all the way through before proceeding to answer all the questions. In order to test whether our intuition was correct we polled 100 students who had completed CS1101, their responses are presented in the figure below.

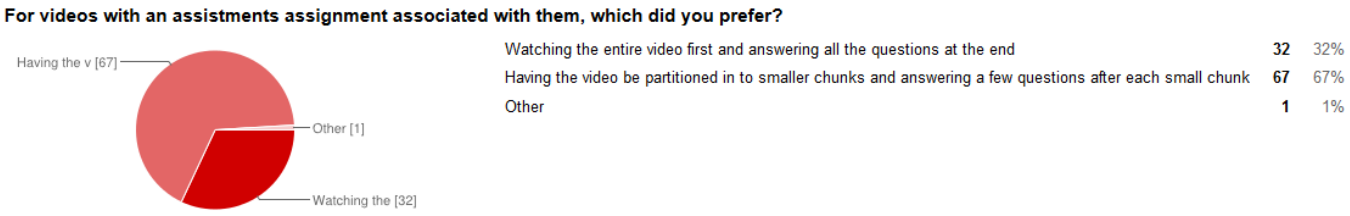


Figure 4:Student Responses on partitioning

The general consensus among the students seemed to match our own, thus most of the problem sets were based on partitioned videos. While most of our problem sets are based on partitioned videos, not all of them are. Videos that were too short or did not have natural breaks in their flow were not partitioned.

The last issue we faced in our initial iterations of content was how complex a problem should be. With some of the videos covering complex and fairly involved concepts, there was a natural tendency to create longer more complicated problems. However, we quickly realized that

this tendency was not a positive one. We realized that if it takes one of us longer than a few minutes to understand a problem, a student seeing the concept for the first time is unlikely to understand the problem at all. An example of one such problem is presented in the figure below.

Question 1: For the following problem, please produces the Stepper output for the program listed below when the given check-expect is run from within Racket. You may choose to either run the Stepper from within Racket (this is suggested, especially if you have no prior experience with the Stepper utility) or you may choose to simply follow the code and indicate the given sequence of output.

Important note: In order to appropriately indicate the answer that you are providing, please concatenate the sequence you have selected (combine it) into a single integer value. For example, if you have selected a sequence of 1, 2, and 3, it must be submitted as 123 in order to be properly submitted as a candidate solution.

The code and check-expect (copy and paste the code and check-expect listed below into Dr Racket and apply the Stepper utility in order to determine the answer to this question):

```
(define (element-count loe counter)
        (cond [(empty? loe) counter]
        (else (element-count (rest loe) (+ counter 1)))))

(check-expect (element-count (list 0 1) 0) 2)
```

Choices for used for constructing sequence (all choices are unique (cannot be used again) and must be used once):

1.) all definitions have been successfully evaluated.
2.) (empty? (cons 1 empty)) *Determine whether or not the rest of the list is empty.*
3.) (+ 0 1) *Add 1 to the current value of the counter.*
4.) (+ 1 1) *Add 1 to the value of the current counter.*
5.) (rest (cons 1 empty)) *Take the rest of the list from the check-expect, which is empty.*
6.) (list 0 1) *Form the list in the check-expect.*
7.) (rest (cons 0 (cons 1 empty))) *Retrieve the rest of the list.*
8.) ((empty? (cons 0 (cons 1 empty))) *Indicate that the list in the check-expect is not empty.*
9.) (empty? empty) *Determine whether or not the rest of the list is empty.*
10.) (element-count empty 2) *Apply element-count to the rest of the list with the current counter value of 2.*
11.) (element-count (cons 1 empty) 1) *Apply element-count with the rest of the list with the current counter value 1.*
12.) (element-count *Apply element-count to the list in the check-expect.*

```
(cons 0 (cons 1 empty))
 0)
```

13.) (cond *Access the condition in which list empty in element-count*

```
(true 2)                    and return the value of the counter.
(else
 (element-count
  (rest empty)
  (+ 2 1))))
```

14.) (cond *Add 1 to counter because the list is not empty.*

```
(false 0)
(else
 (element-count
  (rest (cons 0 (cons 1 empty)))
  (+ 0 1))))
```

15.) (cond *Evaluate the condition in which the list is not empty.*

```
(else
 (element-count
  (rest (cons 1 empty))
  (+ 1 1))))
```

16.) (cond *Access the condition in which the list is not empty.*

```
(false 1)
(else
 (element-count
  (rest (cons 1 empty))
  (+ 1 1))))
```

17.) (cond *Apply the element-count method to the rest of the list.*

```
(else
 (element-count
  (rest (cons 0 (cons 1 empty)))
  (+ 0 1))))
```

Figure 5: Example of an overly complex problem statement

Many such problems were eventually fixed by splitting them into chains of smaller problems. This method of splitting one large problem into several smaller ones both made the problems easier to grasp as well as providing the instructor with data about what particular step of the solution the students struggle with the most. An example of the first in a chain of small related problems is presented in figure 6.

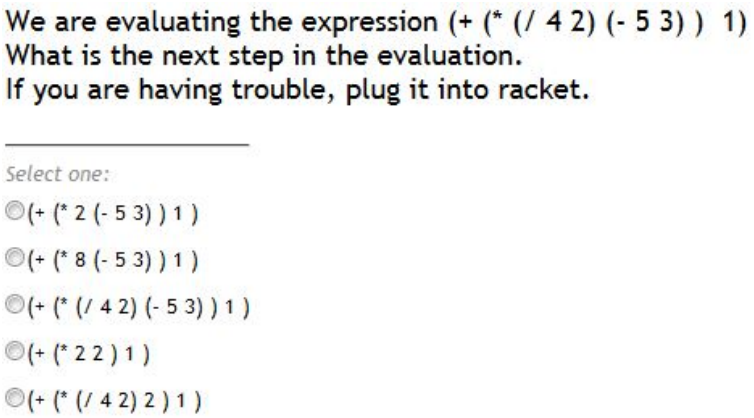


Figure 6:First in a chain of simple related problems.

The above problem is the first in a chain of 4 related problems which incrementally break down the question into simpler and simpler forms. Thus all of the questions that we deemed too complicated were either removed entirely or made into a series of several questions.

After having all the issues mentioned above, we were finally ready to agree on a standard format of a "good" problem set.

Final Format of a Problem Set

After having tested several failed models, the project group agreed on a final format that all problem sets had to follow.

Content for problem sets was developed in Microsoft Word or a similar medium to avoid spelling and grammar mistakes; as such mistakes made our content look unprofessional and could lead to unnecessary confusion among students.

Each question had to be of a reasonably short length in order to avoid confusion and increase the questions accessibility. The general rule of thumb we followed was that a question should be short enough so that a member of the IQP team would be willing to check it. Further

information on the problems of having questions that are too long and complicated is presented in the section on “First Iterations of Content.”

Because Assistments did not have a built in way to skip a question or give up, in order to allow students to proceed past a problem they did not know how to do, the final hint on every problem had to contain the answer. For problems with multiple possible correct answers only one was displayed as correct in order to minimize confusion among students.

In order to make sure the students were watching the videos rather than just attempting to skip to the questions, the first question on a non-partitioned video was always “Did you watch the entire video?” with “Yes” as a correct answer and “No” as an incorrect answer. The same was not done for partitioned videos, because we felt it would be too tedious to have three or four questions asking whether a section of the video was watched.

Every problem set ended with the question “On a scale of 1 to 10 how well did you understand the material presented in the video?” This question existed both in order to allow students to leave feedback on a particular problem set, allowing us to see what problem sets need improvement, as well as providing the instructor with a simple way to gauge the classes comfort with a specific topic.

Initially, the problem building environment we were using brought the user to a new page for every problem, thus no problem could use information from a previous problem, because the student did not have access to that information. Also, in order to allow the student to refer back to the video, it had to be included in every question. In this case for a video that was partitioned in to sections, text was provided before every problem to let the student know whether the question referred to a part of the video they had already seen, or whether the question referred to

a new section of the video. The first question in a section was preceded by one of two possible standard texts. The first question on the first section of any video was proceeded by “please watch the first segment of this video before answering these questions,” while the first questions of all the following parts of the video were proceeded by “Before answering this question, please watch the next segment of the video.” All intermediate questions in a partitioned video were preceded by “This question is from the Nth segment, which you just watched.” Where the N was replaced by the number of the section of the video in the partition

However, we later discovered a more advanced editor which allowed us to create questions in which the student to view all previous questions by simply scrolling up in the page. In problems using this format, the video only every section of the video only appears once, it only appears in the first question pertaining to the section. If the student wishes to refer back to the video all he or she has to do is to scroll up to the question in which the video appeared. Using this format a student does not need to read the text proceeding a problem to determine whether the problem refers to a new section, rather, he or she can easily see that a problem is based on a new section of the video because that section appears in the question, and nowhere else. While we believe that the format created by the more advanced editor is better in terms of both readability and professionalism, we did not convert the problems created in the less advanced editor to the more advanced format as we did not believe it to be a good use of our time, thus there are still several problem sets which are in the old format.

With fill in the blank problems we created simple questions which only had one or two correct answers. This is because the fill in the blank option in Assistments only checks if the answer the student provides is fully identical to the correct answer in all attributes including spacing and capitalization. Thus we had to be careful and make sure that our fill in the blank

questions had very few correct answers. If a fill in the blank question was too complex it was either turned into a multiple choice question or split into several smaller questions.

In order to be user friendly, we limited our problem sets to no more than ten questions, not including questions like “Did you watch the entire video?” or “On a scale of 1 to 10 how well did you understand the material presented in the video?” We did this because we believe that too many questions would be discouraging to students and would thus make them put less effort into trying to figure out the correct answer.

All problem sets had to be peer reviewed by a member of the IQP team who was not involved in writing them before they could be considered complete. This was done in order to minimize errors as well as make sure that the problem set was reasonable. The checker of any problem set had to make sure that all the problems were doable by a student with no prior experience, thus the problems were meant to be simple for a student who was comfortable with the material of the course. The checker also had to make sure that problem set did not contain errors of either a grammatical or technical nature. Finally the checker had to validate that all the questions were based on material from the relevant video as opposed to material that is purely review and material from outside sources.

Having put these standards in place for all of our problem sets, we were able to create professional looking, problem sets of a higher quality that we would have otherwise created.

Results from Students Working with the Content

After we had agreed on a standard format for the problem sets, we were ready to test the problem sets by giving them to students and gauging their responses. While many attributes of

the students' performance interested us, what we wanted to know most about the problem sets was, whether the problem sets were beneficial and necessary, what the students found most helpful in the problem sets, what the students wanted fixed in the problem sets, and what attributes the students wanted to be added to the problem sets.

We had two main sources to gather data from. One, source was the comments that students left on the questions. Even though we did not assign many problem sets, around 10, we received around 294 comments and thus had plenty of data on what the students thought of particular problem sets. Our second source of data was a survey conducted at the end of C-Term in the CS 1101 class that had tried some of our problem sets. 100 students responded to the survey, thus we yet again had a large sample size. It should be noted, that because not all the students responded to the survey, our results were subject to non-response bias and likely exhibits under-coverage of students who were not passionate about the course, or students who were doing very well in the course and did not need the bonus points being offered for completion of the survey. However, because the rate of non-response was fairly small, less than twenty present, we considered the bias that arose from it to be negligible.

The result from working with the students that we most cared about was whether the problem sets were necessary and helpful to the students understanding of the material. The results from the general survey held at the end of the CS 1101 class are presented in the following figure.

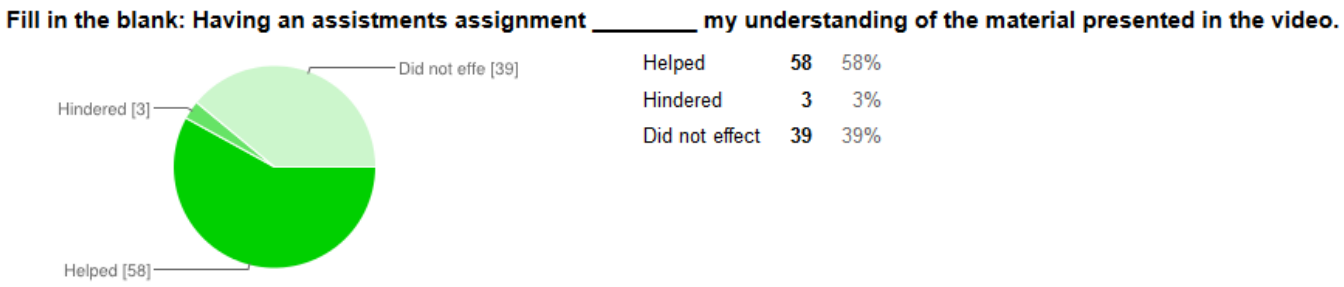


Figure 7:Student responses on helpfulness of problem sets

From the results of the survey, it can be seen that the majority of the students found that even the twelve or so test problem sets they were exposed to, helped their understanding of the course (Due to timing constraints we were unable to run all our problem sets on the students). Very few students felt hindered by the problem sets. Those that did were primarily hindered by inconsistencies in formats well as the errors that occurred in some of our problems. The majority of those errors have since been fixed thus it is our expectation that in the next group of students who use our content fewer, if any, students will feel that their understanding of the material was hindered by the problem sets. Interesting results also arose from asking the students their attitude on the "flipped model" of learning. Those results can be found in the figure that follows.

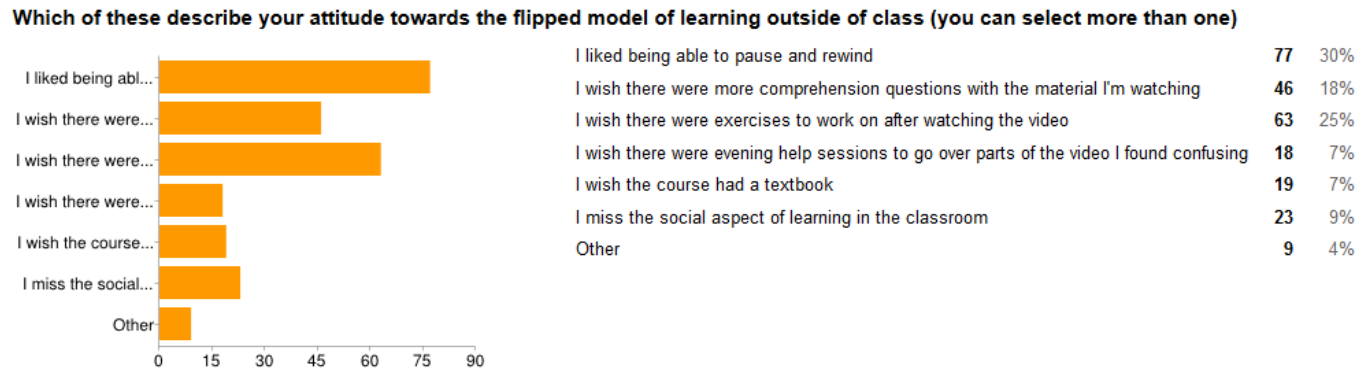


Figure 8:Student Opinions on the "Flipped Model"

Of the 100 students surveyed, 63 felt that they would have benefited from exercises to work with after watching the video, while 46 felt that they would have benefited from

comprehension questions on the material. These results show that many of the students felt that extra practice problems associated with each video would be beneficial. From the comments that the students left, it can be seen that that many students cared about doing well on the problems sets, and took them very seriously. We believe this, because the students, in general, seemed to prefer hard conceptual problems to simple comprehension questions. On several questions that used multiple choice to fill in the blanks in a partially completed program, several students complained about how the multiple choice made the question too simple and less rewarding. However, on a similarly worded question, where, rather than multiple choice, students had to use a word bank to complete the problem, several students praised the problem for its difficulty, saying that, in spite of its difficult nature, the problem was rewarding to complete. The fact that students seemed to prefer harder, more conceptual problems, seems to indicate that the students took the problem sets seriously and strove to do well on them. Both the comments on problems and the responses to the survey seem to indicate that the inclusion of problem sets into the course is beneficial to the students' comprehension of the material.

Having confirmed that the students found our problem sets to be beneficial, we became interested in what aspects of our problem sets the students found most beneficial. As mentioned previously, most students found simple exercises in generating code or using new language constructs to be the most helpful. We followed this advice and added simple coding exercises to assignments where the felt fitting. However, a good number of students also enjoyed having comprehension questions and our assignments also have a good amount of comprehension videos to allow the student to actively watch videos that are more theoretical. In the survey, we asked the students what format of question they found most helpful. The results obtained from the students are presented in the figure that follows.

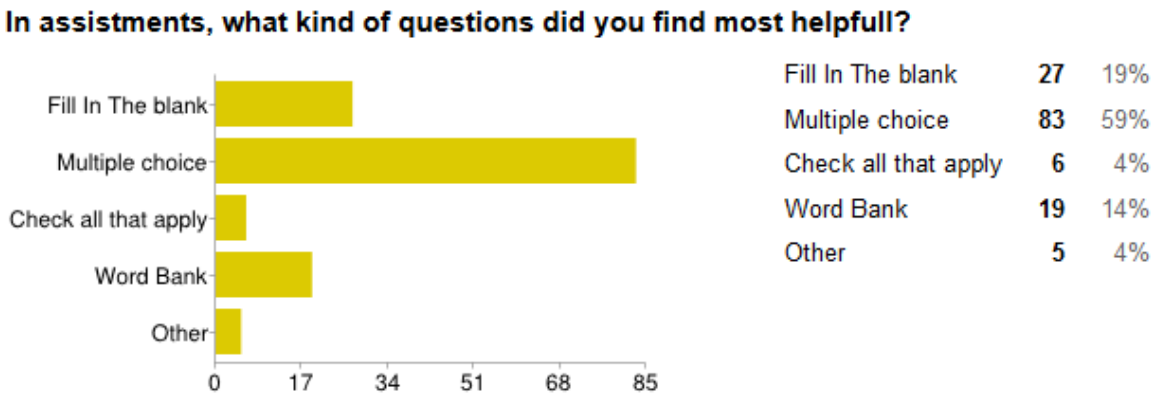


Figure 9:Student Opinion on Most Helpful Question Type

The majority of students seemed to prefer multiple choice questions above all other types. While this result was mildly surprising as our intuition told us that multiple choice questions were the simplest and least interesting question, upon analyzing the students' responses we believe that we understand why the students prefer multiple choice questions to all others. Multiple choice as a format is the most straight-forward out of all the formats, thus multiple choice questions led to the least amount of confusion among students and were thus the most helpful in terms of practicing the material. We yet again heeded the wishes of the students, as most of our questions are multiple choice. We incorporated much of the students' feedback about what was effective in our early assignments into our later problem sets.

Next, we wanted to know what the students thought had to be fixed in our problem sets. Both in the survey and in the comments on specific questions, many students complained about mistakes in spelling and grammar, pointing out that such mistakes made the problem harder to read and understand. As mentioned previously, in order to accommodate that criticism as well as make our problems of a more professional quality, before any problem set could be considered complete, it had to be checked by a person who did not write it.

From the figure above showing what question type the students found most helpful as well as from student comments, it can be inferred that students deeply disliked questions of the "check all that apply" format. While we initially believed the format to be extremely useful as it allows us to write fairly difficult and involved questions, we understand why students had problems with the format. Since the assistments system does not tell the students which of their responses were right and which of their responses were wrong, it was likely frustrating for students to understand the question but miss one small detail and be marked completely wrong. We took the students views into account when creating our content, thus "check all that apply" questions are far less prevalent in our later content than they were in our early content. However, this does not mean that we have no "check all that apply" questions as we still believe that there are certain topics that are best covered by that format. In analyzing students' feedback we incorporated both positive and negative feedback into developing our final product.

The last area we were interested in was what the students wanted added to the problem sets which we had not done. Many of the students wanted more practice in code generation. We figured out a way to allow the students to practice code generation, by having them fill in key features in a partially complete problem. Many students also wanted the problems to be longer and more intricate to allow deeper understanding of the material. We tried our best to create a mix of shorter and longer questions in order for our assignments to be beneficial to students of different levels. Overall though, there were very few features and formats that the students felt were missing from our problem sets.

Overall, the student responses reaffirmed the beneficial nature of our work as well as providing us with feedback on what aspects of our problem sets were most beneficial, and what aspects required improvement.

Where to Go Next

Extensions of our Project for the Near Future (Version 1.1)

While we are proud to say that we accomplished much with our project and believe it to be in fairly good shape, there are several issues and areas of interest that, due to logistical issues and time constraints, we were not able to fully explore. Thus we would like to leave some advice as to some potential areas of research that future participants in this project can explore.

The first, and most important, of these areas is student feedback. We were very fortunate in the amount of feedback the students gave, considering the fairly limited amount of content we exposed them to. We received nearly 300 comments on our problems as well as the fact that 100 students completed our survey, which provided us with a fairly large amount of feedback on several different aspects of the course. However, it would have been beneficial if some of the contact we had with students was face to face and not just through comments and surveys. It would have been, for example, interesting to have the students come in for a review session in the evening and have them simply complete an assistments problem set from that night, as we observed. This simple experiment would likely provide us with very valuable feedback as it would allow us to gauge the students' reaction to an actual problem set. It would also be useful in testing some of the assumptions we developed over the course of this project about the optimal formatting of content as well as what kind of questions work best. These meetings would also be beneficial as they would allow us to see how the students approach our content. These review sessions would also allow us to meet with the students face to face and discuss with them what they think of the assistments problem sets ant the flipped model as a whole. While much of the feedback we would get would likely be similar to the feedback provided in the survey, it is also

likely that the students would have relevant thoughts that we did not think to cover with our survey questions. Overall, we believe that face to face meetings with the students using the content could provide future groups working on this project with some interesting and informative feedback.

Another action we would have like to perform but were unable to, due to time constraints was a scientifically rigorous study to measure how helpful the content was. While the majority of students claimed that they found the problem sets helped their understanding, we had no concrete data about the level to which having the problem sets increased the students' grasp on the content. What we would propose is to perform a study in which half the class completes the video with the problem set, and the other half of the class only watches the video and does not complete the problem set. We would then use the next day's quiz scores to compare the results from the students who had access to the problem set, to the results of the students who did not. These results would provide us with data on both whether our content benefits immediate understanding, based on how the students do on the new material part of the quiz, as well as whether our material affects retention, based on the review part of the quiz. In order to avoid any ethical issues we would perform what is sometimes known as a cross-over study, meaning that we would perform the experiment twice and switch the roles of the groups for the second trial. Meaning that people who previously only watched the video would now complete the problem set and vice versa. While time constraints and scheduling issues prevented us from performing this study, we believe that the performance of such an experiment would be greatly beneficial to any students who will work to expand off our results in the future.

A mix of rigorous experimentation and more face to face contact with the students is how we envision the future of this project.

Technological Extensions (Version 2.0)

I. Current technologies

a. Assistments

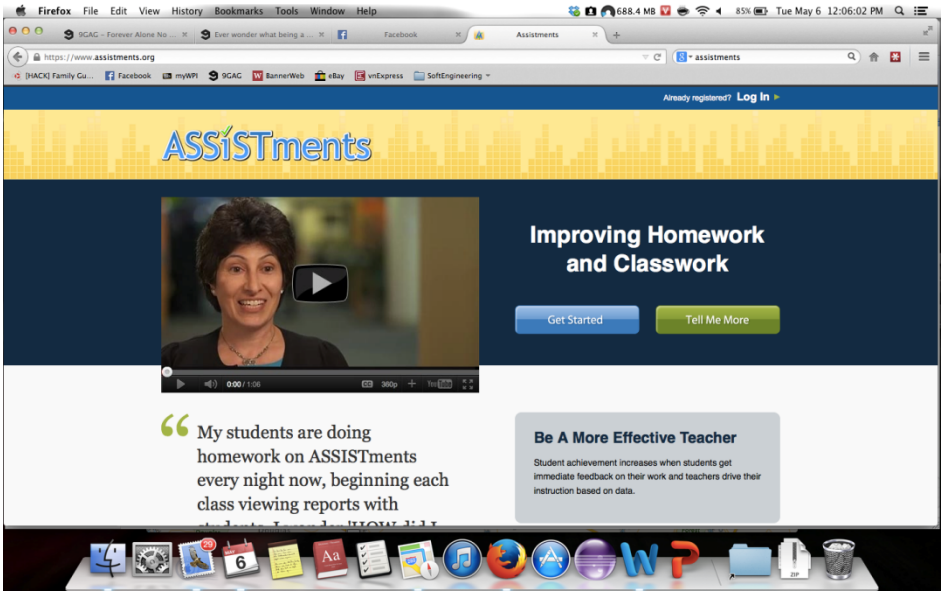


Figure 10:Assistments Home page

A system created by faculty and students at Worcester Polytechnic Institute dedicated to improve E-learning. It has various question types and statistical tools that provide immediate feedback to the teachers. From that data, the teacher can evaluate the performance of his or her class.

b. Type of questions

There are 5 types of question:

- Algebra question: The system grades the student's answer in form of numbers or mathematical expression.
- Fill in: The system grades the student's response character by character.
- Open response: The system saves the student's responses that need manually grading from the teacher.
- Multiple choices: The system grades on the choice the students made in the question.

- Check all that apply: The system grades on the choices the students made in the question.
- Rank: The system strictly grades on the order of the answers students chose.

c. Statistical data

Assistments provides detailed information about students' performance and class progress. Teachers can easily view how the students performed on a specific assignment. Because every student's action is logged into the database, the teacher can lookup useful data such as the amount of time it took the student to answer that question. Furthermore, the data can be used to evaluate the performance of the whole class so changes can be made in the syllabus to maximize learning efficiency.

d. Format of the current CS 1101 assignments

Because of the nature of this project – Flipping the CS 1101 classroom, heavy use of videos and web-based response is used in conjunction with feedbacks from students.

In every assignment, each student is assigned one or more videos that he or she needs to watch. Because we want the students to focus on specific parts of the video, we decided to set a start and stop point in it.

After the student had watched the video, the student will be asked if the he or she has completed watching the video carefully. If the students answered "No", the system will advise the student to watch it again. Otherwise, the students can proceed to the main questions of the Problem Sets.

II. Future technologies

The research group has been working on Assistments since A-term 2013 to C-term 2014 and found several changes could significantly improve Assistments system:

1. Machine-graded response

   Because **CS 1101** is the course that involves programming, there might be infinite number of ways to solve a single problem. Therefore, grading strictly character-by-character is not an efficient way to test the student's programming skill.

   We believe that implementing a new grading method - Java-based computer grading – would expand the realm of possibility for students to be freely creative on the answers they provide.

   The teacher will have to code a simple Java application that receives the answers input of the students, parse, evaluate and return the correctness of the student's response to Assistments. In detail, we could assign a jar file to a specific problem and make it the main grader for the problem.

   Another implementation is the installment of separate compilers of a specific language such as *Racket, Java and C*. In software engineering, the test-driven development approach makes it feasible for the teacher to test the student's program using Unit tester (for example, JUnit based in Java.)

   This is a very innovative and revolutionary method to actually teach Computer Science – especially in the introduction class like CS 1101. It is undisputable that the best way to *learn how to code is to code*. Students should be given a chance to submit a code that can be evaluated directly and see immediate result. This method is widely used in large programming contest such as the USACO, International Olympiad in Informatics, and ACM contests.

   A lively example of this implementation is the Polish webpage http://spoj.pl/

Figure 11:Example Alternative Resource

This page provides many Computer Science problem sets and automatically grades the student's code by compiling the program, running it and comparing the output it provided.

This method will also solve the hassle of grading because it is computerized. Human grader can make mistake while reading and compiling the code. With a standardized machine-testing, Assistments should be able to resolve the problem and saves the teacher's time.

Therefore, we think that implementing this feature and using it in WPI Computer Science courses is the first thing to do.

2. Enhanced question editors

It is very wonderful that we can create questions on a WYSIWYG (What you see is what you get) editor in Assistments. It provides core functionalities that make us able to create basic HTML-based question. However, we have found out that the question editor is very hard to use and inflexible.

We could hardly paste the content from Microsoft Office Word 2010 to the web editor because the text format and layout are broken.

Because we are talking about an introduction to Computer Science class, code-highlighter is a great feature to have. According to the students' feedback, they have a hard time reading the code because the in **DrRacket**, the codes are highlighted with colors so they can easily understand the overall structure of the code.

For advanced users, we might want to add the HTML code editing functions so we can easily design the layout of the question text as we want.

An example of an advanced text editor can be improved to be like this:

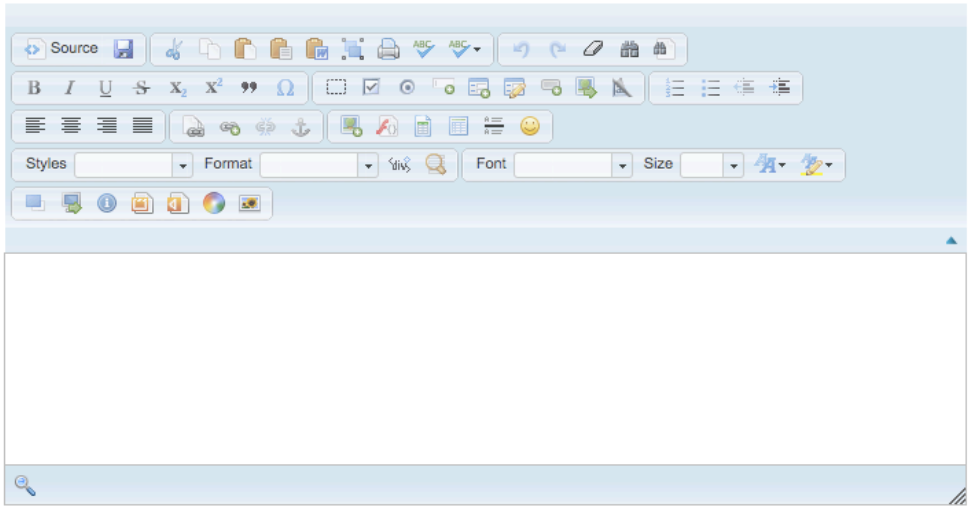


Figure 12:Example Editing Environment

3. GUI redesign

During the research, we have also found out that the interface of assignment builder very difficult to use. We had a hard time navigating the problem sets and accessing statistical data linked to a specific assignment.

Moreover, there are some errors caused by Assistments server when the teacher submitted an assignment. The server then returned a XML-styled error message although the question and answer choices are legitimate.

Conclusion:

We began this project with two main goals on which to base our work. The first goal was to develop a curriculum for the CS1101 class (Introduction to Program Design) based on the online lectures of professor Gregor Kiczales. These materials had to act as supplements to the lectures in order to help increase the retention of information by students trough focusing their attention on the vital points of each video and asking questions on those concepts. Our second goal was to explore the "split model" in general, to look at its advantages and disadvantages and to see what improvements should be made to the "MOOCs" (Massive Open Online Courses) that are currently being offered. Overall, both of these goals were accomplished and suggestions for future iterations of this project were made.

Upon beginning the project it was important to us to gauge the student's reaction to certain types of material and the flipped model in general. Thus we began by holding optional review sessions in the evenings for students to attend and try out some of our materials. While the majority of the materials presented at these review sessions were not used, these review

sessions provided us with invaluable information regarding the types of materials we should present and how we should present them. Mainly we realized that every piece of material should be tied to a specific lecture and that the questions we ask need to be specific, pf a reasonable difficulty, and only based on the terms and concepts mentioned in the lecture.

Having tested some of our ideas on the students we were ready to decide on a final format for an assignment. These criteria can be found in a previous section of the paper, or in our appendix, but the basic idea of all the criteria is that every question in an assignment must be of reasonable difficulty and length as well as a good review of the specific concepts covered in the video. The standards warned about not trying to include concepts not covered in the video, as this was something we found to be ineffective and distracting from the main purpose of the assignments. We also set standards for how we wanted the questions to be interspersed in the video, as we found a problem with many MOOCs to be the fact that the questions were all clumped together, and we found that the MOOCs that were most effective were the ones where the questions were interspaced throughout the video.

After we agreed on a set of standards for the development of problem sets, we created an individual problem set to accompany each individual video. We initially considered the idea of grouping several videos into a single problem set, while this idea initially seemed logical, we quickly realized that it decreased the flexibility of both the instructor and the students and, because flexibility is one of the most important attributes of a MOOC, we abandoned the idea and made individual problem sets for each video. While the problem sets ranged in style, the questions which we found most effective were questions where students actually had to generate code. This was accomplished through providing the students with a template and having them fill in blanks with short lines of code. This method also taught the students good programing

etiquette as it forced them to produce programs that are well formatted and follow the design recipes. The students responded well to these kinds of questions and often stated that they liked them the best even though these questions were substantially harder than simple, conceptual multiple choice questions. Thus we successfully created a curriculum based on the provided videos and completed our first goal.

Our second goal was to gather data as to whether our materials were relevant and useful. We based the format of our materials greatly on Khan Academy which is one of the most popular sources of MOOCs; we liked how Khan Academy mixed conceptual and practical questions. However, where we thought that Khan Academy was lacking was that their videos were entirely separate from their problem sets, where as we wanted to create a system in which the problem sets were strongly dependent on the videos.

When we asked students what they thought about the supplemental materials that we provided them, an overwhelming majority said they found them helpful and liked having materials that allowed them to practice the concepts they saw in the videos. Thus we believe that the responses we received from the students validate our work and show that the materials we provided made a positive impact on the students. If we had more time and resources we would have like to perform a scientifically rigorous test to see how much of an effect our materials had. However, this is something we leave to future iterations of this project. Overall though, we successfully created a set of supplementary materials and used student responses to verify the usefulness of our materials. We also looked at other similar studies for inspiration. Thus, we completed both of our initial objectives.

Acknowledgements:

Our group would like to thank Professor Joseph E. Beck of the Worcester Polytechnic Institute computer science department for providing the initial direction for this project and access to all of the resources that we required. His helpful guidance and suggestions were genuinely important and without them, the outcome of this project would certainly not have been the same.

Additionally, our group would like to extend our gratitude to the WPI CS 1102 class with whom we worked from August 29, 2013 to October 17, 2013. Their participation in classroom surveys and in the review sessions that our group operated were very informative and provided a valuable information-base upon which we developed the preliminary versions of the problem sets that we later distributed.

Finally, our group would like to thank the WPI CS 1101 class with whom we worked from January 16, 2014 to March 7, 2014. Their participation in classroom surveys and their completion of our problem sets through the ASSISTments platform enabled us to acquire valuable insight with respect to problem design, student preference, student learning style, and the overall effectiveness of certain approaches to teaching introductory-level program design.

Appendix:

Data from the survey that was not directly mentioned:

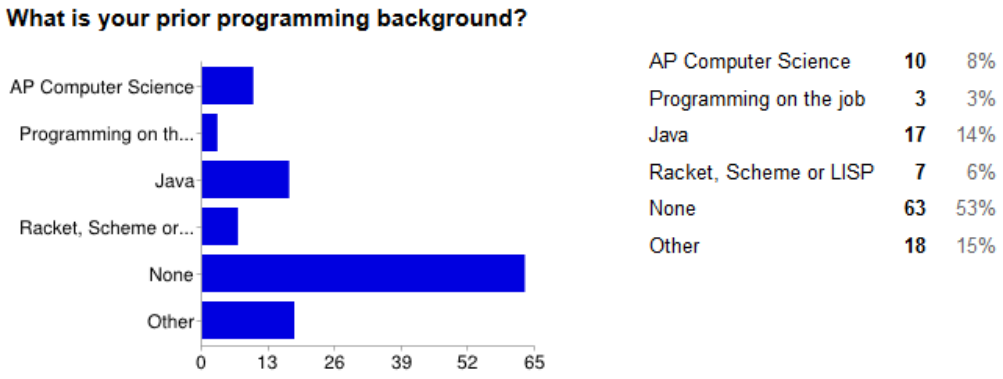


Figure 13: Students' Prior Programing Background

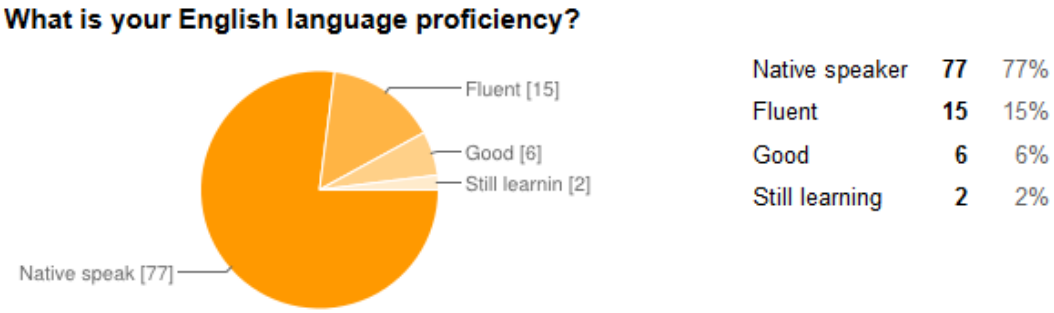


Figure 14: Students' English Proficiency

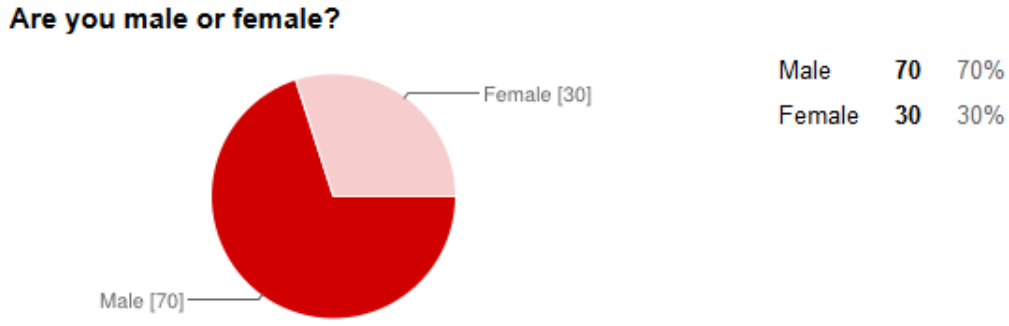
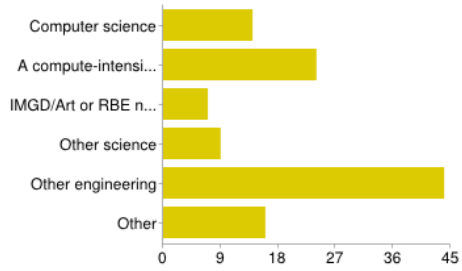


Figure 15: Students' Gender Distribution

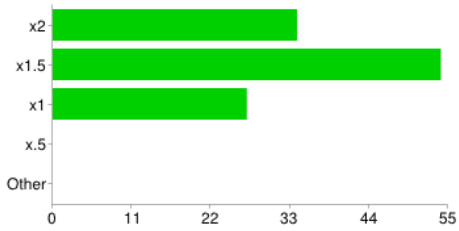
What describes your current or intended major?



Computer science	14	12%
A compute-intensive field (e.g., IMGD/Tech, RBE/CS)	24	21%
IMGD/Art or RBE not focusing in CS	7	6%
Other science	9	8%
Other engineering	44	39%
Other	16	14%

Figure 16: Students' Majors

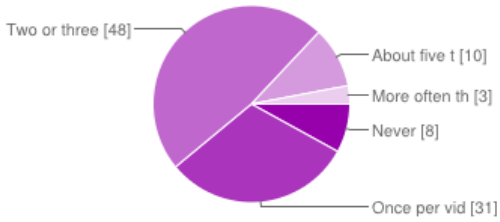
What speed did you typically use for playback on the videos?



x2	34	30%
x1.5	54	47%
x1	27	23%
x.5	0	0%
Other	0	0%

Figure 17: Playback Speed Statistics

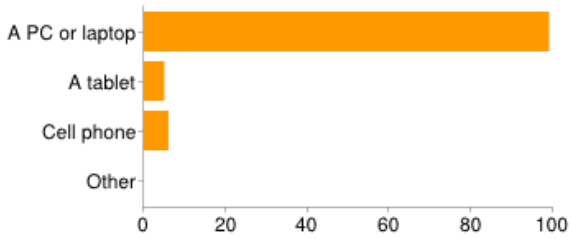
How often did you typically pause or rewind the videos?



Never	8	8%
Once per video	31	31%
Two or three times per video	48	48%
About five times per video	10	10%
More often than 5 times per video	3	3%

Figure 18: Pause and Rewind Statistics

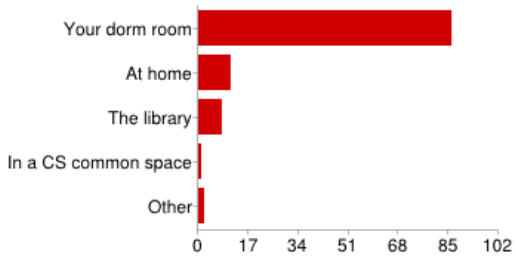
What device did you typically use to watch the videos?



A PC or laptop	99	90%
A tablet	5	5%
Cell phone	6	5%
Other	0	0%

Figure 19: Device Statistics

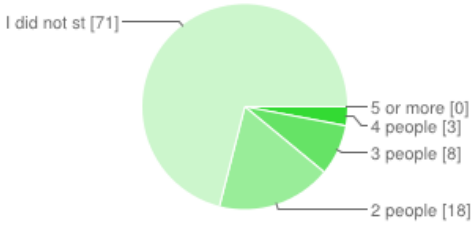
Where did you watch the videos?



Your dorm room	86	80%
At home	11	10%
The library	8	7%
In a CS common space	1	1%
Other	2	2%

Figure 20: Study Location Data

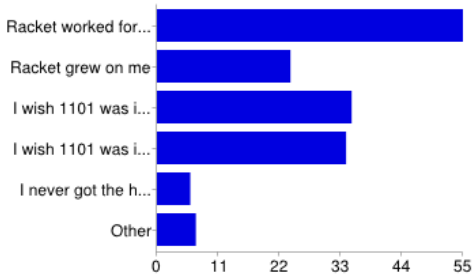
If you studied in groups, how large was your group size?



5 or more	0	0%
4 people	3	3%
3 people	8	8%
2 people	18	18%
I did not study in groups	71	71%

Figure 21: Study group Size Data

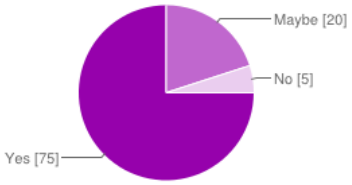
What did you think of the Racket language? (check all that apply)



Racket worked for conveying the core concepts for this course	55	34%
Racket grew on me	24	15%
I wish 1101 was in Java	35	22%
I wish 1101 was in Python	34	21%
I never got the hang of the parentheses	6	4%
Other	7	4%

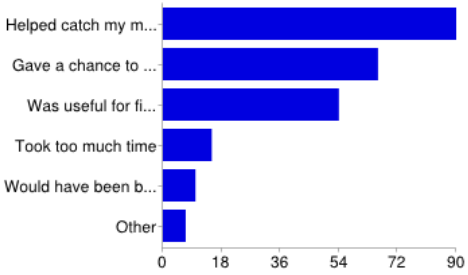
Figure 22: Thoughts on Racket

Was it useful to go over the quiz questions right away after the quiz?



Yes	75	75%
Maybe	20	20%
No	5	5%

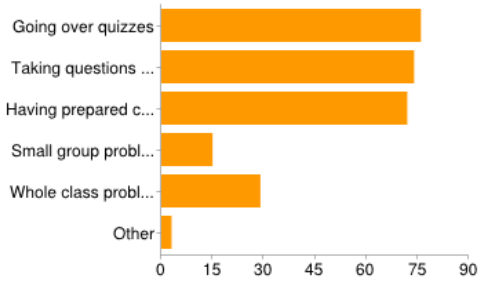
Going over the quizzes right away (check as many as you agree with)...



Helped catch my mistakes and misconceptions quickly	90	37%
Gave a chance to ask questions about the content	66	27%
Was useful for finding poorly written quiz items	54	22%
Took too much time	15	6%
Would have been better if solutions were just posted online	10	4%
Other	7	3%

Figure 23:Quiz Related Statistics

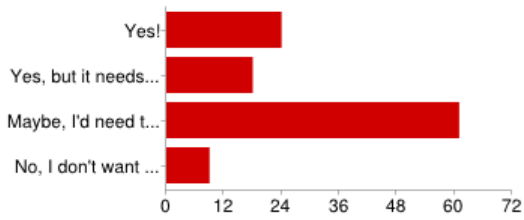
Which of the following do you think were a good use of class time?



Going over quizzes	76	28%
Taking questions about areas of student concern	74	28%
Having prepared content on material I thought would be challenging	72	27%
Small group problem solving	15	6%
Whole class problem solving	29	11%
Other	3	1%

Figure 24: Opinions on Class Time

Should WPI offer more courses in a flipped manner?



Yes!	24	21%
Yes, but it needs to be handled better	18	16%
Maybe, I'd need to see another example	61	54%
No, I don't want to see someone else try this experiment	9	8%

Figure 25: Students' Opinion on Flipped Courses

Proper Assignment Format:

Checklist for each Assignment Section:

1. Write content in Word (or equivalent). Grammar/spelling will make me very unhappy
2. Each question is reasonably short enough that we (the IQP team) would be willing to write/answer/check.
3. When video is over 8 minutes long, segment into chunks
4. All problems, not in test mode, have a hint ***available*** that provides the ***correct*** answer
    a. it is good to have intermediate hints that reteach, or give one of the answers or exclude some answers
5. Stick to content in the video. for example:

. stuff you think is obvious but is not in the video

a. review material

6. Video appears on every question
7. Do not have questions refer back to prior questions (students cannot see it)
8. Checker: if you're stuck, it's probably too hard

. be careful with factual question, those might be hard if you don't remember the video

a. does the question seem answerable from the content?

b. if you're going "huh?" that's dangerous

How to begin a non-segmented video

1. Say "please watch the entire video before answering these questions"
2. Question #1: Did you watch the entire video? "True" is the correct answer

a. immediately tell student to watch the entire video

How to begin a segmented video

1. Say "please watch the first segment of this video before answering these questions"
2. For a question that continues from the same segment, say "this question is from the Nth segment, which you just watched"
3. For a question that advances to a new segment say "Before answering this question, please watch the next segment of the video"

How to conclude a video

1. Ask students to rate their understanding on a scale of 1 to 10
2. have this item in *test mode*

Test mode: just accepts input, but does not give feedback
Takes first answer and goes on

Example Problems

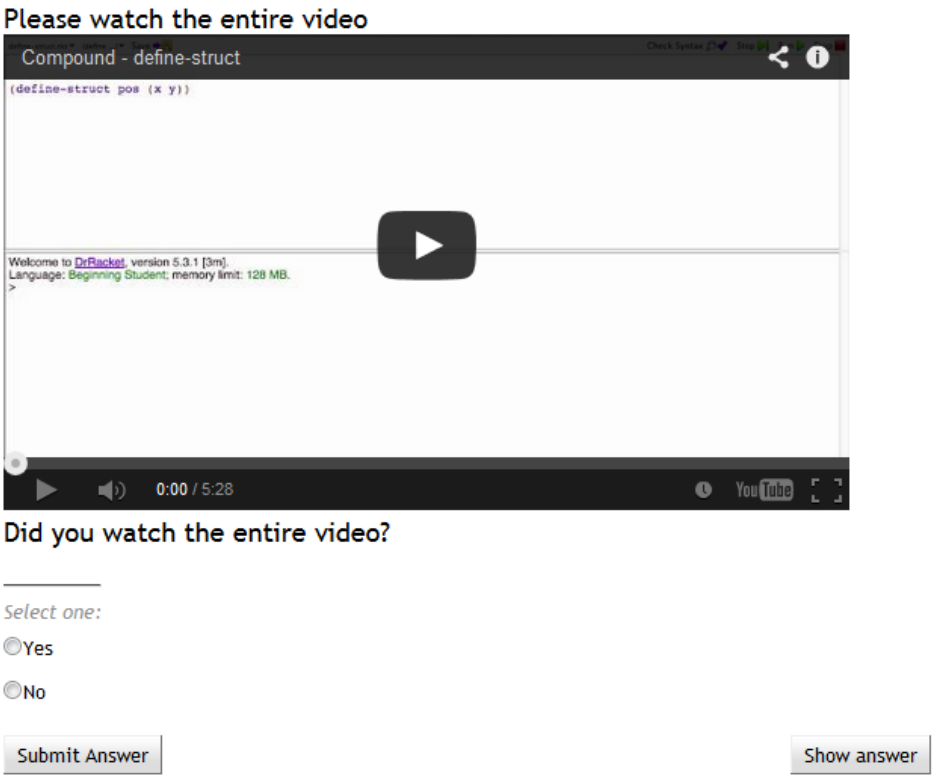


Figure 26: First Question Of Non Partitioned Video

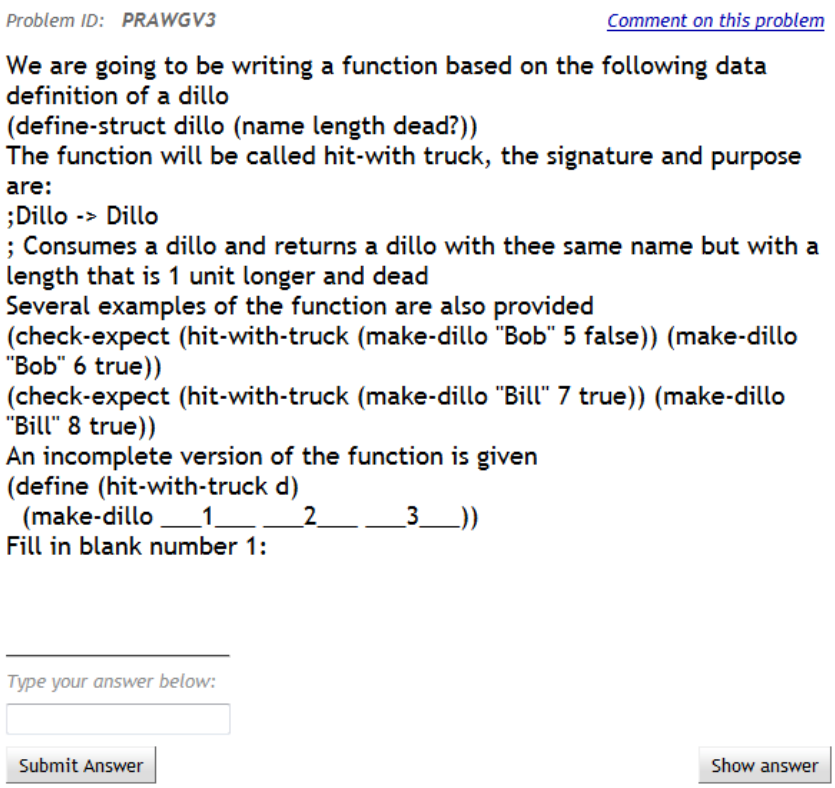


Figure 27: Complex Fill In the blank Problem Example

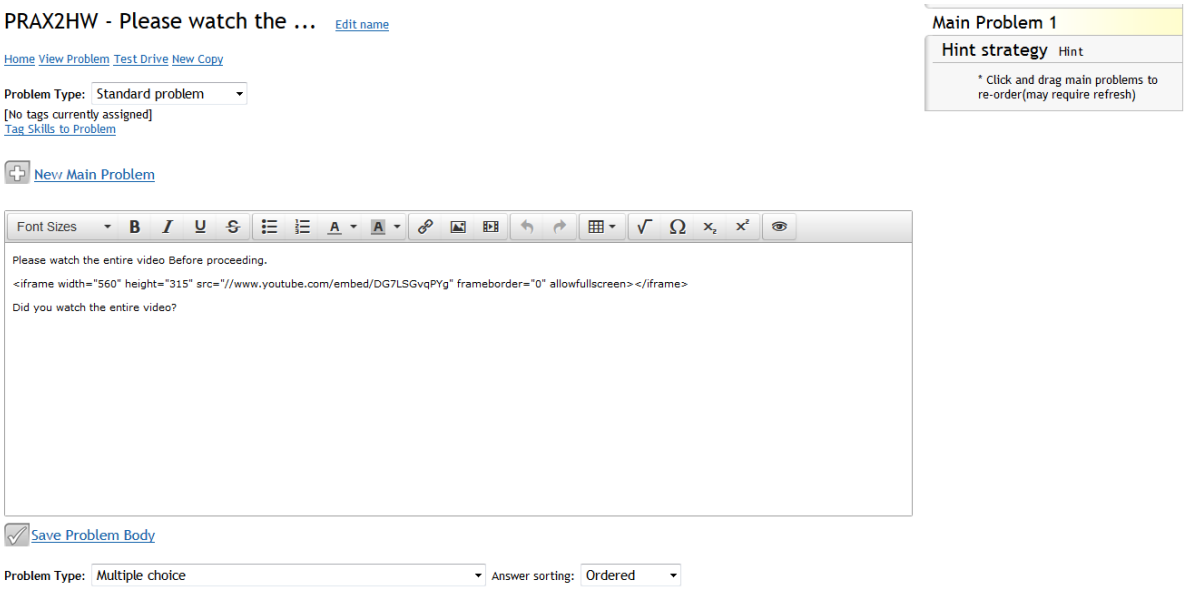


Figure 28: Assistments Advanced Editor