ABSTRACT
We provided 21 short YouTube videos for an Introduction to Programming in Java course. Students were surveyed on how often they watched the videos and did the readings, and how much these activities contributed to their learning. When professors reduced lecture time and increased lab time, students watched videos and read significantly more. Their test scores were at least as high and they indicated they would prefer to not have more lecture. The YouTube videos also provided a source of outreach for the university, drawing a large number of views, including the 13-17 year-old demographic.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education, H.5.1 [Information Interfaces and Presentations]: Multimedia Information Systems – video, D.3.3 [Programming Languages]: Language Constructs and Features.

General Terms
Languages.

Keywords
YouTube, Java, videos.

1. INTRODUCTION
There seems to be a Catch-22 in teaching computer science. Ideally, we would like to spend class time helping students engage with the material and not simply lecturing; however, this requires students to have some knowledge of the material before class. So we assign readings, but repeated studies have shown that only 20-30% of our students will have read before class starts [1]. As a result, we often spend a significant amount of class time summarizing and presenting the readings, but this simply reinforces to students that they don’t need to prepare for class [2]. The internet has also changed the dynamic for teaching. More and more lectures from prestigious universities are available online (see, e.g. [3,4]). If we end up mostly lecturing with students as passive learners in the classroom, why should they choose to come to our classes, instead of watching lectures on the same topics online?

A significant amount of ink has been spilled trying to figure out how to get students to read more before class (e.g. [1,2,5,6]). Generally, the ideas rely on negative motivation—you should design your course so that students will not succeed if they don’t read. This is accomplished by quizzing, random questioning and monitoring compliance. There are, however, positive suggestions on motivating the readings, selecting them more purposefully and assigning readings closer to the due date.

The question left unasked, though, is “Is reading the best way for students to prepare for class?” Since between 75% and 83% of students are visual learners instead of verbal learners [7,8], readings are not playing to their strengths. Additionally, some faculty may even feel a bit hypocritical pushing student reading so much when they managed to be very successful in academia without reading before class, or sometimes even purchasing the textbook.

After asking this question, and considering both the prevalent learning style of our students and the fact that college students on average have spent almost four hours a day watching videos and less than an hour a day reading [9], we started exploring the idea that videos might be a better way to get students to prepare for class. We first asked a number of students, “if we asked you to watch a short YouTube video before each class, would you do it?” Unanimously, the students all said this would be a great idea.

Of course, saying you will do something is different than actually doing it, but the positive response was sufficientl y motivational to put the effort in to make videos for our introductory Java course. To make sure students would stay motivated and focused, our goal for each video was to be less than five minutes in length. We ended up averaging four minutes per video, with a median time of three and a half minutes, and only two videos exceeding six minutes. YouTube imposes a maximum length of 10 minutes, which ensured even the longest video did not become exceedingly long.

As it turned out, students did end up watching the videos, and we capture some of their feedback in Section 4. Section 3 gives more details on how we made the videos and their content. Work by others on “inverted classrooms” and “hybrid classrooms” are described in Section 2. At the end, Section 5 describes possibilities for future work.

2. RELATED WORK
Using media outside the classroom to free classroom time for discussion, experiments and labs is not a new idea. The “inverted classroom” [10] basically swaps what are traditionally in-class
activities (lectures) with what are traditionally outside of class activities (homework assignments). Lage, Platt and Tregalia used this in an economics course. Students were supposed to watch a recorded lecture before coming to class. The instructors began each 75 minute class by answering student questions on the videotaped lectures, then used a hands-on activity to demonstrate the concept, followed by worksheets and review questions. They reported that generally students liked this format. (Only two of their 189 students requested a change to a lecture-oriented class, and the students’ response to “I believe I learned more economics with this classroom format” was 3.9 on a 5 point scale).

Kaner and Fiedler [11, 12] use the term “blended learning” to describe their makeover of a course on software testing to move lecture outside the classroom as videos. They describe how they adapted their course (providing information on hardware and software used, totaling $8000, and also the amount of work involved—35 hours per each hour of videotaped lecture). They do not provide detailed student feedback.

Day and Foley [13] ran an experiment with two sections of a human-computer interaction course at Georgia Tech. In the experimental section, classroom lecture time was replaced with hands-on learning activities. Students watched 27 web lectures (totaling 9 hours) to learn this content. Day and Foley made strong efforts to control as many variables as possible in what they called a “quasi-experiment” and found that students who were in the experimental (web video) section (on average) performed better on every course assignment.

Gannod, Burge and Helmick [14] applied these principles to a software engineering curriculum at Miami University. They provide the results of student surveys in a special topics course on service-oriented architecture. In that course, they made 65 podcasts available to students (ranging in length from a few minutes to 50 minutes). During the course meeting time, instructors answered questions on the podcasts and then students worked on assignments. Their results with this model were mixed. 100% of students indicated that “podcast lectures are helpful...and allow class time for assignments.” 86% agreed that “we like the inverted class structure.” On the other hand, 92% agreed that “the class shouldn’t rely so heavily on podcasts” and 56% agreed that the instructors should “use podcasts to supplement class lectures instead of replacing them.”

Hybrid classes (a combination of distance learning and classroom learning) also use videotaped lectures combined with in-class activities [15]. Linsday [16] describes this as the “best of both worlds”, but Reasons, Valadares and Slavkin [17] ran an experiment where students in a hybrid class did worse than both traditional and distance learners.

In each of these cases, the principle is to provide more time for student interaction in the classroom by having students spend a significant amount of time watching videotaped lectures before coming to class. While we have a similar goal (making the classroom more interactive), our approach is different. Rather than having students watch a full lecture before coming to class, we instead provide very short videos that give the highlights and introduce students to the material. By keeping the videos short, we hope to maintain a high level of motivation and viewing. This was the first question students always asked when we were sounding out the idea—“how long will they be?”

3. THE VIDEOS
We began the process of creating the videos by looking through the syllabus for the introductory Java class and identifying 21 topics for which we would make videos. Eighteen of these were language topics not specific to an Integrated Development Environment (IDE). Those topics are:

1. Hello World
2. Basic Types and Objects
3. If Else Statements
4. While Loops
5. For Loops
6. Exceptions and Input
7. Drawing a Picture
8. Static Methods & Constants
9. Animation
10. Reading a Text File
11. Writing a Text File
12. Arrays
13. 2D Arrays
14. ArrayLists
15. Creating a Class
16. Java Constructors
17. Equality
18. Inheritance

For each video, we began by writing a script. The script consisted of the words that would be spoken by the narrator, the Java source code for all the programs that would be discussed, and, for some videos, animations done in PowerPoint.

For each video, we first recorded the audio. Getting quality digital audio was the worst part of the project, especially as we did not have much money to allocate to purchasing hardware or software. We tried several different microphone and software configurations. The built-in microphone in our Fujitsu T series laptop generated audio files with too much background noise. A Cyber Acoustics AC-102b headset (~$15) with microphone did much better; however, there was still a certain amount of audible buzzing. An RCA Digital Voice Recorder VR5220 (~$40) had the least noise, but had the unfortunate side-effect that it gave the narrator’s voice a pronounced isp, even using the highest quality setting on the recorder. After more research, we purchased a Plantronics Audio 655 USB headset (~$31) to avoid going through the microphone jack on the laptop. This produced the best quality sound, but still required cleaning up the audio with Audacity [19] using its “noise removal” feature at a very low setting.

Getting high-quality video was much easier. We recorded the video using a free screen capture program, VidShot Capturer [20].
While listening to the audio, the narrator used the mouse and keyboard to highlight text, progress through a PowerPoint animation or run the programs. We did experiment with recording the video and audio simultaneously, but found that this generated more out-takes. Also, we determined it was easier for us to perform the actions in the video in time with pre-recorded audio than it was to try to make the audio match with pre-recorded video. Since there isn’t constant action in the video, it was hard for the narrator to gauge the pace at which to read. When arriving at the next action, there might be an awkward pause if read too fast, or the narrator would suddenly realize he was a sentence or two behind.

Windows Movie Maker [21], available as a free download, enabled us to combine the video and audio files into a movie suitable for uploading to YouTube. Additionally, we used it for adding some captions in the videos and putting a title at the end.

We elected to use YouTube for hosting the videos because it was simple and free. One unanticipated benefit was that a large number of people from around the world have discovered and viewed the videos. The video on Making a GUI in NetBeans is the most popular, and averages 500 views a month from users around the world. Thirty percent of these viewers are 13-17, and 55% are above 24 years old.

Although we do not have any hearing-impaired students in our program, we do make the videos available to the world on YouTube, so we added closed-captioning. YouTube Subtitler [22] made this very easy. The first step is to split the script into short lines or pairs of lines (40 characters or less) separated by blank lines. You then paste this into the text window and push synchronize.

As shown in Figure 1, you can now watch the movie while advancing through the subtitles. Underneath the video on the right is a lightning button. Simply push the button when a subtitle should appear and release when it should leave. After working through the whole video, you can download a text file with the timings, or simply click a link to upload the subtitles directly to the YouTube video (assuming you are already logged into YouTube). Adding subtitles not only makes your videos available to a wider audience, but also provides additional information to Google about the content of the video, which can make it more likely to be a search result.

For the content of the videos, we provide one or more Java programs that are concrete examples of the concept being introduced. Source code for the videos can be downloaded from http://java.martincarlisle.com. The first video, “Hello World”, simply walks through a Java program that prints “hello!” to the screen, explaining each word that appears. Some introductions to Java treat this as a “magical incantation” [18], delaying explanation to later, but we have found that freshman and sophomore college students are extremely uncomfortable with that much abstraction right away, and need time before they can reason about programs without having some idea of how each part works.

Later videos walk through small Java programs that perform tasks like printing the numbers from one to five, converting liters to gallons, counting the number of words in a file, or sorting a set of numbers. Each program is designed to demonstrate a particular feature of the language (for loops, static methods, Scanners, ArrayLists).

4. RESULTS

We used the set of videos in an Introduction to Programming in Java course at the United States Air Force Academy. The Air Force Academy is an undergraduate institution with an enrollment of approximately 4,400 students. All of the students are traditional students, live on campus, and are between 17 and 27 years old. The Introduction to Programming course is taken by approximately 60 sophomores and juniors each year who are majoring in computer science, computer engineering, or information systems. The course is offered as a double-period lab (i.e., there is a one hour lab period immediately following each lecture hour in the same classroom). This is the first course in the computer science major (during the freshman year students take only core requirements). The videos were given to the students as links to YouTube from the syllabus on the course web site. The course had four offering times taught by 3 different professors. All three professors have over ten years of experience as full-time faculty members. Each of the four offerings used the same syllabus and graded events. Professor 1 created and narrated the
videos. His course offering was first, and both Professor 2 and Professor 3 observed his lectures before teaching theirs.

Professor 1 spent the least amount of time lecturing. He began each class by highlighting both correct and incorrect examples from student submissions on the previous labs. (Incorrect examples were anonymized). He would answer questions, and show very briefly the new concept. After this, he had the students work on the day’s lab, and remained in the classroom to answer questions.

Professor 2 gave the longest lectures. He presented each concept in detail in the classroom before starting the labs. He taught two of the four offerings. Professor 3’s style was somewhere between that of Professor 1 and Professor 2. He did prepare PowerPoint slides and gave a lecture at the beginning of each class, but it was shorter than that of Professor 2.

Fifty-seven students across the four offerings completed a survey providing feedback on the videos, the readings (from [18]) and the lab activities. Students were asked how often they watched the videos and read before class. For the videos, readings and labs, they were asked on a five point Likert scale if each helped them understand the material and if they were enjoyable. They were also asked if they would like more lecture, and had open responses for comments on the videos, readings and labs.

Fifty-seven students across the four offerings completed a survey providing feedback on the videos, the readings (from [18]) and the lab activities. Students were asked how often they watched the videos and read before class. For the videos, readings and labs, they were asked on a five point Likert scale if each helped them understand the material and if they were enjoyable. They were also asked if they would like more lecture, and had open responses for comments on the videos, readings and labs.

Common to all three professors was that the students did not want more lecture. Only one student each for both Professors 1 and 3 agreed that they “would prefer more lecture time.” (Four of Professor 2’s students agreed with this statement, which was a bit surprising, as his lectures were the longest. He did have 8 strongly disagree, a rate four times that of the other two professors). Also, all four sections rated the labs as most helpful (4.5, 4.1 and 4.9 respectively).

One surprising result was that Professor 1’s students were more positive about the videos than the others. Eight of fourteen students reported watching the videos 90-100% of the time and their written feedback was the most enthusiastic—“I wish every class did this” and “videos are better than reading.” We believe this is because Professor 1 narrated the videos and the students felt more connected to his voice. Professor 1 did not advertise the videos any more than his colleagues. One suggestive comment from a student in a different class was that they wanted to “see who is talking,” which obviously doesn’t affect the content.

Professor 1’s students also scored highest on a programming test (averaging 82.9%). Professor 3’s students averaged 78.5% and Professor 2’s students averaged 73.0%. The entering GPAs were 2.95 for Professor 1, 2.91 for Professor 2 and 2.89 for Professor 3. Since the class sizes were so small, the two-tailed t-test did not allow us to reject the null hypothesis that the means were equal. Nonetheless, the fact that Professor 2’s students did worse than Professor 3’s despite having a higher incoming GPA is suggestive that this might be related to the fact that they spent so much less time preparing for class.
5. CONCLUSIONS AND FUTURE WORK
Creating short videos can be a positive way to get students to engage with the material before coming to class. Students indicated the videos helped them learn the material. The professors who reduced their lecture time found that students prepared more for class not only in watching videos, but also in doing the reading. These students preferred the shorter lectures and having more time to work on programming in class. They also performed better on the test (though the sample size was not sufficient for this to be statistically significant). The videos are not only helpful for an on-campus course, but placing them on YouTube can be a simple outreach for your university (we had a large number of views from 13-17 year olds).

One possible future experiment is having multiple different narrators for the same videos. This would allow us to test how important it is for students to have a connection to the narrator. Another useful experiment would be to create videos for a larger course, which would provide bigger sample sizes for the statistical analysis.

6. REFERENCES