# LIGHTS, CAMERA, ACTION! VIDEO DELIVERABLES FOR PROGRAMMING PROJECTS*

*Christa Chewar and Suzanne J. Matthews*
*Department of Electrical Engineering & Computer Science*
*United States Military Academy*
*West Point, NY 10996*
*845-938-55{62, 77}*
*{christa.chewar, suzanne.matthews}@usma.edu*

**ABSTRACT**

Student-produced video deliverables describing their software development projects are a pedagogical option made practical by the prevalence of smart-phones and tablets in the modern classroom. We describe various uses of video deliverables in software development courses within our computer science program. We discuss insights based on our experience over two years, and present "Best Practice" recommendations, which may be useful for anyone interested in incorporating similar practices in their own programming courses. Overall, we have found that video deliverables provide many benefits and efficiencies for students and faculty alike.

## INTRODUCTION

Within the fields of computer science and engineering, much research has gone into exploring the utilization of video in the classroom. The majority of existing research concentrates on the creation of instructor-produced artifacts, such as lecture recordings [4,5,6,8,13,14,15,17], screen/podcasts [13,14], and visualizations [9]. Despite a prevalently positive view of the impact of instructor-produced video and related visualizations on student education, the experimental body of evidence suggests that the passive consumption of video lectures is of limited benefit [6, 8, 9, 15]. Video lectures

are useful to students who miss or are unable to attend lessons in person [15]. However, there seems to be a growing consensus of the importance of student engagement in the process of *creating* artifacts [9, 12]. When students create artifacts, they gain deeper insight into the processes being explored and develop a greater sense of responsibility for their learning [9, 12].

For the past two years, we have enthusiastically embraced a practice of requiring video deliverables for many programming projects within our Computer Science program. We define a *video deliverable* as a student-produced video artifact that is required for some graded event. Where students would previously submit only their code (and perhaps a short report) as the graded deliverables for programming projects, we have typically revised requirements to focus students on producing a short video demonstrating the program in use. Source code is still turned-in, with the expectation that it produces the same results seen in the video-but, the emphasis is on the video.

Our pilot efforts in requiring video deliverables were born from frustration with grading student programming projects, especially those which involve significant environment configuration to achieve a functional system. Applications that make use of device or API features such as sensors or location services require additional context to fully assess. Security concerns bring additional challenges with safely building student code. Even after time is taken to rebuild a student's project and situate it in an appropriate usage context, non-intuitive user interface design can mask otherwise solid development work that is difficult to grade exclusively through code review.

While live demos present another option for grading programming projects, we often feel overly constrained with the classtime available to accomplish all assessment. In high enrollment courses, live demos are often impractical, robbing students of the valuable experience to organize and communicate their projects to their classmates. Live demos are also notoriously catastrophic, especially for introductory level students who cannot resist the temptation to make a final improvement that results in an insurmountable compiler error. Lastly, students are generally terrible at predicting how long a presentation will take, and have poor time management when presenting. This can eat into valuable presentation time of other students. We also notice that students often felt that those who presented in a later time-slot had a greater advantage, having observed the presentations given previously by their peers.

We were also inspired by the success stories from courses in other disciplines that require students to produce video artifacts. In engineering [1] and business [2], educators found that requiring students to create podcasts encouraged students to develop desirable technological skills, and required greater levels of communication and team-work than oral presentations [2]. In a nursing program [11], students were asked to video their oral presentations about different techniques in medical diagnosis, and then perform peer assessment of their colleagues' videos. Students rated the exercise as being highly valuable, with the highest rated aspects being teamwork and collaborative assessment [11]. Studies of the use of student-created videos in obesity stigmatization [16], or chemistry labs [10] where students either reflect on describe a central process, found that

students displayed a deeper level of learning while developing valuable auxiliary communication skills.

In this paper, we discuss our efforts in incorporating video deliverables into various software development courses in our computer science program. We discuss insights based on our experiences, and present "Best Practice" recommendations, which may be useful for anyone interested in incorporating video deliverables in their own programming courses. Our experience suggests video deliverables provide many benefits for both students and faculty. In the future, we plan to further explore how to better integrate video deliverables in all aspects of our computing curriculum.

## METHODS FOR INTEGRATING VIDEO

We integrate video deliverables into four computer science courses that emphasize software development. These four courses are taken by undergraduate computer science students at all levels: freshmen and sophomores in their introductory computing course; mid-level students learning advanced programming concepts in a required course and in an elective; and seniors building a complex software system as part of their capstone requirements. In addition, the introductory course and capstone course are taken by other majors in the department, including electrical engineering and information technology majors, who were also required to turn in video deliverables.

Requiring video deliverables of our students was a relatively easy task. All students at our university are required to purchase a laptop and a tablet computer, both which come with video recording hardware. We point students toward free software, such as ScreenCast-O-Matic (http://screencast-o-matic.com) and the Windows MovieMaker video editing tool. While most students chose to produce screencasts (where the video focused on a screen, rather than an individual), many students found creative ways to incorporate skits or themselves into the presentations. Our video deliverable integration efforts in each of our four courses are summarized below:

CS1 - Introduction to Computer Science: This first course in the major introduces students to concepts such as modular design, problem solving strategies and basic data structures. In the final weeks of the course, students are required to implement an open-ended project with instructor approval. In previous semesters, we were plagued by delays mainly caused by students who laptops (or programs) "inexplicably" stopped working at their allotted presentation time. Video deliverables ensured that there was a working demo the day of presentation, allowed us to increase the number of presentations that can occur in a given day, and greatly reduced the amount of stress in the process. We post the videos on our class YouTube channel, where students can "like" or comment on them.

Object Oriented Concepts: A major course objective in this third programming course is a semester-long project that follows a test-driven development paradigm. Students create a simulation of a four player turn-based game of their choice. The autonomous players can be dynamically assigned various strategies, but move decisions

are always predictable and therefore testable. Once the graphical elements of the game take form, students use a series of video demos to show basic game play and testing of strategies, displaying the content and execution of unit tests, and interleaving integration tests. The videos clearly demonstrate a student's competency with predicting execution results of every user command, comparing actual program results, and identifying deficiencies.

Android Programming: This elective available to CS majors involves a substantial development project, where students create or extend an Android app that allows a user to interact with web services and a database backend. The major project deliverable is a video walk-through of the system, intended for another student that is new to Android and web development. The walk-though demonstrates all parts of the fully functional system in use, and then explains the most interesting parts of the code developed by the student. Within several minutes, the viewer sees approximately 1,000 lines of code, spread across multiple files. The narration explains the responsibilities of major modules during the application lifecycle, clearly showing the student's level of understanding.

Senior Design Course: All of our computing majors spend two semesters developing a capstone project in interdisciplinary teams, responding to the needs of real stakeholders. We follow an Agile Scrum approach using three week "Sprints," which requires students to produce a minimal product immediately, and continuously add features as requirements evolve. At the end of each Sprint, students submit a 10-15 minute video that demonstrates all current progress, reflects on stakeholder reaction, and proposes the work for the upcoming Sprint. Sprint reviews generally provide an excellent synchronization point for all team members, instructors, and advisors-having these in a video format allow comparison between teams, richer product demos, and asynchronous/repeated viewing. The smartest students record quick video segments every work session of results achieved, which they also use for internal daily "standup" updates. By the end of the year, each project accumulated approximately two hours of high quality video, fully documenting project rationale and results. This should be a valuable resource to next year's teams continuing the projects.

## RESULTS

We discovered numerous benefits in using video deliverables for software development projects. Video deliverables improve student organization and communication skills, streamline course administration, and facilitate the generation of artifacts for program-level assessment and reflection. Benefits from each perspective are discussed in turn. We also present some of the most common challenges we have encountered in requiring video deliverables for class project work, with the mitigations we have used. At this point, all observations are informal and anecdotal, but grounded in our experience.

**Student Benefits**

Video deliverables improve student communication and time management skills. Student outcomes for computing sciences typically include effective communication with diverse audiences, and effective time management. Video deliverables force students to better organize and prepare their thoughts. For a video product, our students often require about an hour after coding is complete to write a script, prepare visual aids, practice narration, edit, and upload the final video. As a result, students procrastinate less and avoid last-minute efforts that often lead to unfortunate code errors. We also find that requiring students to take the extra time to create video artifacts helps them recognize the "big picture" aspects of their projects, rather than merely producing working code.

Video deliverables enable student peer evaluation and self reflection. Video artifacts enable us to provide examples to students of "good" and "bad" demonstrations. Unlike oral presentations, video artifacts can be replayed. This enables students to have a frame of reference as they work to improve their own communication skills. A collection of such artifacts also enables the creation of video-based electronic portfolios that students can use to highlight their undergraduate programming accomplishments.

Video artifacts enhance continuity with future teams of students working on multi-year projects. When students working on a current iteration of a project watch video produced by previous students, they have the opportunity to more deeply understand the motivations behind a design decision, or hear how a program works in the voices of the developers themselves. This is very valuable, as it's often impractical for current students to contact alumni students who worked on a project and ask about the reasoning behind certain design decisions. This is also preferable to the written manuals that we ask students to produce before they wrap up a project, which tend to contain insufficient detail.

**Benefits to Course Adminstration**

Video deliverables streamline course administration. Live demos are often infeasible in courses with large number of students or overly complex setup requirements. For example, grading student projects that integrate a web service, database, and an Android application requires a suitable platform to build the project, specific emulator settings or a compatible device to test the app, and authentication using with multiple user roles to appreciate the features.

Video deliverables for these types of projects help us avoid issues associated with environment set up and project building for grading or program assessment, especially when there could be security concerns with students' files or version incompatibilities. Setup and tear-down times required for live demonstration are eliminated, increasing the number of student demos that can be shown in class. If all the videos cannot be shown during class hours, we randomly share a collection of the video artifacts. The instructor and students can watch the remainder of video artifacts outside of class. Since all videos are turned in at the same time, it also removes worries that students presenting later gain

unfair advantages by seeing peers present work first.

Video deliverables facilitate more detailed instructor feedback. We also found that video deliverables permit us to give more accurate and robust feedback ("your statement at 1:43 is incorrect"). Replaying videos and observing student behavior during the video also helps increase an instructor's confidence when assessing an individual student's level of knowledge and engagement with the project.

**Program-Level Benefits**

Video deliverables create valuable assessment artifacts for Computer Science programs. Accreditation processes often require computer science programs to produce artifacts that provide evidence of enabling the stated student outcomes. Without providing reviewers with a means to quickly appreciate scope and quality of student products, it's difficult to use student software development efforts for assessment purposes. Video deliverables can be used for assessment of embedded indicators and act as samples of student work. This process becomes easier as assessment materials are organized and distributed digitally. Our department used these artifacts in our last ABET evaluation cycle.

Video deliverables are useful for benchmarking student success across semesters. Changes in development environments and software libraries can make student code difficult to rebuild after several years, making it challenging to directly compare software products that result from project work. However, video demos can avoid this altogether, allowing easy benchmarking of metrics such as "here's what students can typically create after one semester learning how to program" or "here's how students performed software testing." Video artifacts can also be used during recruiting events to inspire future generations of students. Some students in the most recent semesters of CS1 report that they enrolled in the course after being inspired by some of the project videos made by students in previous semesters.

**Challenges & Mitigations**

Being required to produce video deliverables has caused some angst and apprehension among students, especially since many students had never prepared a video as a graded class requirement. Over a few semesters, we collected a list of the most common negative reactions from students. This is summarized in Table 1, along with the suggestions we offer in response to mitigate the student's concern. There are certainly some other challenges for instructors related to video deliverables that we have overcome, but presentation of these is left for the "best practices" collection.

**Table 1. Student-centered challenges with using video deliverables, with suggestions for mitigating.**

| Challenge | Mitigations |
|---|---|
| students don't like to see/hear themselves | • captions can often be just as effective as narration<br>• students can be allowed to write the script for narration, which is read by a friend for the video<br>• encourage students to keep the camera on the product, not themselves |
| students may dislike the process of making a video | • having multiple courses within an academic program use video deliverables increases the "value" of learning basic videography<br>• include a "Getting Started" guide resource on class websites, linking to free screen recorders/editing software |
| students may be confused what a good video looks like | • have a few examples of effective and ineffective videos, with a few notes highlighting positive and negative aspects<br>• use some classtime to allow students to watch and react to each other's videos |

## CONCLUSIONS & FUTURE WORK

As popularity of devices such as smart-phones, laptops and tablets soar, it becomes very likely that all students have a readily accessible high-quality video recording device. The ubiquity of this hardware along with freely available video software makes it easier than ever before for students to create video artifacts.

Spurred by the limitations of oral presentations and the success stories of student-created video artifacts in other fields, we injected video deliverables in various computer science courses at our university. While integrating video was not without its challenges, the many benefits we observed easily compensate. Our students grew leaps and bounds as communicators and had a better ability to appreciate the "big picture" view of the projects when asked for video deliverables. These artifacts also gave students an

opportunity to watch themselves and their peers academically mature over the course of several semesters. As faculty, we found video deliverables were also valuable for assessment and recruiting purposes.

Based on our two years of experience with integrating student video requirements into programming courses, we have collected the following list of "best practices."

- **Introduce tools.** Spend just a few minutes of class time introducing a few tool options, such as a free screen recorder utility and perhaps an editing tool. A completely inexperienced student can become confident with screen recording in a few minutes of practice.
- **Discuss content expectations.** Provide guidance on exactly what students should show in the videos. Certainly, seeing the working program or system is important, but should they show and/or explain the code? What about testing? Should the video be a recording of the product, or of the presenting the product? Should they be drawing on a blackboard or a piece of paper, or talking with refined visuals? Showing examples of "good" and "bad" videos is very useful here.
- **Coach effective recording techniques.** Give clear guidance to counteract common challenges students face in recording. While recording a program displayed on an external screen is possible with a phone/tablet, this technique requires special attention to ensure adequate focus on text (zooming both really helps). If text cannot be read on the phone/tablet's screen during the recording, it is unlikely to be readable in the video. Angling the recording device differently than the external screen should also be avoided.
- **Discourage inappropriate showmanship early.** Train students to keep the camera mostly focused on products, not "talking heads" or unrelated images. Gimmicky transition effects can consume a lot of a student's time without adding any content value.
- **Specify video format and maximum resolution.** This is especially useful if videos will be presented in class on a single computer. We commonly request the MP4 video format and 720p maximum resolution. While YouTube reduces resolution of uploaded videos, larger videos take longer to upload.
- **Keep it short.** Five minutes is plenty of time for simple demonstrations, but even two minutes can be quite effective. Ten minutes works well for a full code walkthrough and functionality demo of a moderately complex system. Up to fifteen minutes of video is generally useful for a team's Sprint review (representing approximately three weeks of work).
- **Submit files, not links.** Have files on-hand to ensure later accessibility. If students are simply allowed to upload videos to their own YouTube channel, they may later delete them. This eliminates the ability for these artifacts to be used in future assessment or reflection efforts.
- **Require "credits" at the end of a video.** Students should acknowledge assistance,

not only with the project, but also with the video production. Credits should clarify permissions and/or fair use in an educational setting of any music/visuals that were not the student's own work.

In the future, we plan to explore how to integrate video deliverables into electronic portfolios. While electronic portfolios [3,7] are certainly not a new idea, the ready availability of modern video-making software and hardware opens new doors to their construction. YouTube is certainly a tempting platform to use. However, questions arise on how best to organize and maintain large collections of videos over several years using a free YouTube account.

While oral and written communication modes are traditionally exercised in college programs, communication through recorded video is arguably becoming just as important for education modules, concept and product demonstrations, and collaboration. We hope that some of the techniques referred to in this paper will assist other programs in better integrating video deliverables in their software development courses.

## DISCLAIMER

The opinions in this paper are those of the authors and do not necessarily reflect the opinions of the U.S. Military Academy, or the U.S. Army.

## REFERENCES

[1] Alpay, E., Gulati, S., Student-led podcasting for engineering education, *European Journal of Engineering Education*, 35 (4), 415-427, 2010.

[2] Armstrong, G. R., Tucker, J. M., Massad, V. J., Achieving Learning Goals with Student-Created Podcasts, *Decision Sciences Journal of Innovative Education*, 7 (1), 149-154, 2009.

[3] Arter, J. A., Spandel, V., Using portfolios of student work in instruction and assessment, *Educational measurement: Issues and practice*, 11 (1), 36-44, 1992.

[4] Brecht, H., Learning from online video lectures, Journal of Information *Technology Education: Innovations in Practice*, 11 (1), 227-250, 2012.

[5] Crook, A., Mauchline, A., Maw, S., Lawson, C., Drinkwater, R., Lundqvist, K., et. al., The use of video technology for providing feedback to students: Can it enhance the feedback experience for staff and students?, *Computers & Education*, 58 (1), 386-396, 2012.

[6] Dickson, P. E., Warshow, D. I., Goebel, A. C., Roache, C. C., Adrion, W. R., Student reactions to classroom lecture capture, *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pp. 144-149, 2012.

[7] Lankes, A. M. D., *Electronic portfolios: A new idea in assessment*, Syracuse,

NY: ERIC Clearinghouse on Information & Technology, 1995.

[8] Manley, E., Urness T., Video-based instruction for introductory computer programming, *Journal of Computing Sciences in Colleges*, 29 (5), 221-227, 2014.

[9] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C. et. al., Exploring the role of visualization and engagement in computer science education, *ACM SIGCSE Bulletin* 35 (2), 131-152, 2002.

[10] McCormack, S., Ross, D. L., Teaching with Technology: Using Websites and Videos to Increase Understanding of Bacterial Transformation, *Science Teacher*, 77 (7), 40-45, 2010.

[11] Pereira, J., Echeazarra L., Sanz-Santamaria S., Gutierrez J., Student-generated online videos to develop cross-curricular and curricular competencies in Nursing Studies, *Computers in Human Behavior*, 31, 580-590, 2014.

[12] Stasko, J. T., Using student-built algorithm animations as learning aids, *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education*, 29 (1), 25-29, 1997.

[13] Viel, C. C., Melo, E. L., Pimentel, M. G., Teixeira, C. A. C., Presentations preserved as interactive multi-video objects, *Proceedings of the Workshop on Analytics on Video-Based Learning*, 34-37, 2013.

[14] Wan, Z., Fang, Y., The role of information technology in technology-mediated learning: A review of the past for the future, *AMCIS 2006 Proceedings*, 253, 2006.

[15] Wieling, M. B., & Hofman, W. H. A., The impact of online video lecture recordings and automated feedback on student performance, *Computers & Education*, 54 (4), 992-998, 2010.

[16] Zahn, C., Schaeffeler, N., Giel, K. E., Wessel, D., Thiel, A., Zipfel, S., Hesse, F. W. , Video clips for YouTube: Collaborative video creation as an educational concept for knowledge acquisition and attitude change related to obesity stigmatization, *Education and Information Technologies*, 19 (3), 603-621, 2013.

[17] Zhang, D., Zhou, L., Briggs, R. O., Nunamaker Jr, J. F., Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness, *Information & Management*, 43 (1), 15-27, 2006.