Using Student-created Instructional Videos in CS Upper-level Courses: A Successful Strategy in a Functional Programming Course

Pedro Guillermo Feijóo-García[1,2][a] and Christina Gardner-McCune[1]
[1]Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL, U.S.A.
[2]Program of Systems Engineering, Universidad El Bosque, Bogotá, Colombia

Keywords: Active Learning, Student-created Artifacts, Video Tutorials, Computer Science Education, Strategic Learning, Instructional Strategies.

Abstract: In this paper we present findings on a pedagogical approach we designed to enhance students' understanding of Functional Programming, in which they were required to create two video-tutorials. The first video-tutorial assignment asked the students to develop explanations of Functional Programming concepts. The second video-tutorial required them to explain their solutions while completing coding exercises using Haskell. We present a detailed description of the activities, their evaluation, and their impact on students' learning, motivation, and performance. Our findings suggest that the use of a student-created video-tutorial approach can be effective for increasing students' understanding, performance, and engagement on Functional Programming assessments. This suggests that using student-created video tutorials may be a promising strategy to implement in other computing courses.

1 INTRODUCTION

The Computer Science Education (CS ED Research community has been interested in how students learn to program and use programming as a medium to achieve computational literacy (Guzdial, 2016). Many studies have been conducted over the past twenty years on the first year of Computer Science (CS) curricula, with most focusing on CS0, CS1 and CS2. Unfortunately, students struggle to master advanced CS concepts as well, and upper-level CS courses remain an under-researched area of CS ED research. Thus, we lack understanding of the source of students' learning challenges in these courses and on how to best teach upper-level CS courses.

Based on our years of experience instructing a senior-level Functional Programming (FP) course, we have observed learning challenges happening as students transfer skills from other paradigms to the functional one: these challenges are due to new restrictions, the strict need of recursion, and the understanding of functions in higher abstract levels: passed as parameters, anonymity (i.e., lambdas), and functions returning functions. Motivated by William Glasser´s often cited quote that highlights the power of teaching to learn: *"We learn 95% of what we teach to others,"* we asked students to create video tutorials as learning tools to help them increase their conceptual knowledge and fluency in FP and studied its effectiveness to begin growing the body of CS ED research in upper-level CS courses.

In this paper, we discuss our implementation of student-created video tutorials as a course assignment in a FP course at a university in Colombia. We also discuss our findings on the effectiveness of using student-created video tutorials on students' learning, motivation, and performance on course assessments.

2 RELATED WORK

Student-created video tutorials are part of a collection of learner-centered classroom techniques designed to promote student learning and engagement (Guzdial, 2016). Learner-centered design approaches place the learner at the center of the learning process, and invites instructors to consider students' current knowledge, knowledge boundaries, interests, motivations, and expectations (Guzdial, 2016; Bain,

[a] https://orcid.org/0000-0002-3024-1257

Feijóo-García, P. and Gardner-McCune, C.
Using Student-created Instructional Videos in CS Upper-level Courses: A Successful Strategy in a Functional Programming Course.
DOI: 10.5220/0009416404120419
In *Proceedings of the 12th International Conference on Computer Supported Education (CSEDU 2020) - Volume 1*, pages 412-419
ISBN: 978-989-758-417-6

2014). In this section, we present review several of these approaches: active learning, peer instruction, and other types of student-created artifacts.

Studies on active learning have demonstrated their ability to enhance students' performance, motivation, and engagement (Gehringer & Miller, 2009; Kearney & Schuck, 2004). Within Computer Science, studies have shown that active learning activities are effective in helping students to learn CS concepts (e.g., Frank-Bolton & Simba, 2018; Feijóo-García & Ortíz-Buitrago, 2018). Gehringer & Miller (2009) studied the use of active learning activities in introductory CS courses, i.e., CS1 and CS2. Their findings suggest that use of student designed games, diagrams, props, and videos, working on topics like debugging and sorting were effective techniques for increasing students' attention. In general, active learning have been found to be effective in introductory CS courses across institutions, diverse demographics, and countries (e.g., Feijóo-García & Ortíz-Buitrago, 2018; Kearney & Schuck, 2004; Murray *et al.,* 2017). They have also been shown effective in upper division courses on algorithms (Frank-Bolton & Simba, 2018).

Peer-instruction (i.e., peer-review and peer-tutoring) is a commonly used active learning technique in the CS ED community (Feijóo-García & Ortíz-Buitrago, 2018; Porter *et al.,* 2016; Cottam *et al.,* 2011). Peer-instruction positions the student as both an instructor and learner allowing them to learn from and with their peers (Porter *et al.,* 2016). Within CS1 and CS2 courses, peer-instruction has been found to increase students' understanding of topics, their communication skills, and their motivation (Feijóo-García & Ortíz-Buitrago, 2018; Porter *et al.,* 2016; Cottam *et al.,* 2011).

A core feature of peer-instruction is the requirement for students to explain their understanding of a topic to a peer (Feijóo-García & Ortíz-Buitrago, 2018). This is a feature that also exists in active learning activities that involve student-created artifacts (e.g. student-created instructional videos). Studies of student-created instructional videos in K-12, CS1, and an algorithms analysis course, report that instructors and students positively perceived using student-created artifacts to promote learning (Gehringer & Miller, 2009; Kearney & Schuck, 2004). Frank-Bolton & Sihma (2018) reported that student-created videos can promote students' understanding of advanced CS concepts. Additionally, they found that videos' creators performed better compared to students who simply watched the videos. In this paper, we refer to student-created instructional videos as student-created video tutorials as this is the name we are accustomed to calling them.

3 THEORETICAL FOUNDATION

The student-created video tutorials described in this paper were designed to promote significant student learning and engagement using a learner-centered design approach built on constructivist notions of learning and Fink's Taxonomy on Significant Learning (Fink, 2013).

Constructivism defines learning as a process in which knowledge is constructed and adapted by the learner based on the learner's assimilation and accommodation of new knowledge gained in from new experiences or reinterpreted past experiences (Bain, 2014). Both, constructivism and learner-centered design approaches, describe the learner as a dynamic individual, who learns through active engagement in their learning process (Guzdial, 2016; Bain, 2014). As Freire stated, the art of teaching implies the need for continuous learning (Freire, 2012).

Similarly, Fink's Taxonomy for significant learning (Fink, 2013) is based on constructivism and identifies six interconnected dimensions (Fink, 2013). Coding requires skills from two of these dimensions. The first is Foundational Knowledge, which refers to the individual's understanding of how a computer or system works according to its capacities and limitations. The second is the Application Dimension, which considers the coding skills needed to use the computer as a medium (Guzdial, 2016; Fink, 2013). We used these two dimensions of Fink's Taxonomy to help us focus what students focused on in their video tutorials.

Considering knowledge as something not transferable but constructed, we designed the student-created video tutorials to provide learners with an opportunity to consciously reflect on what they understood about foundational Functional Programming (FP) concepts, while verbalizing their understanding as they explained concepts for someone else to learn.

4 OUR APPROACH

This section describes the FP course in which student-created video tutorials were used to foster learning of CS concepts. It explains the video tutorial assignment, its evaluation, and how data are analyzed leading to our findings and results.

4.1 The Course

Our study is based on an undergraduate senior-level Functional Programming (FP) course at Universidad El Bosque in Colombia, South America. The course introduces student to the FP paradigm through programming in Haskell. Prior to this course, Java is the main language in which students are proficient. The course population consisted of sixteen male senior CS majors enrolled in the course during the semester this study was conducted. As a result, we were unable to have female CS students participate in this study. The university offers two curricular tracks: (1) traditional where students take courses during the day and (2) non-traditional where students take courses in the evening or weekends. The study population (N=16) included both traditional [N=9] and non-traditional students [N=7], and all students were between the ages of 18 to 35 years old.

The semester-long course featured three modules over sixteen weeks. Module 1 provided an introduction to Haskell and FP concepts such as lists and tuples, higher order functions, folds, and lambdas. Module 1 included a project, pre and post conceptual and coding assessments. Before our student-created video tutorial intervention, students completed the Module 1 project: a two-week assignment that asked students to conceptually understand all topics covered in the course so far and practically apply their knowledge by completing a programming assignment.

4.2 The Strategy

After the Module 1 project, we assessed students' conceptual and coding knowledge using two pre-assessments. The conceptual pre-assessment asked students to define FP concepts, and the coding pre-assessment asked students to code the solutions for a set of exercises.

Then, the students were asked to create two video tutorials one for the conceptual and one for coding pre-assessments. Each video tutorial required the students to address and explain in detail the set of exercises given in the pre-assessment. In particular, they were asked to discuss the questions in which they struggled to complete on the pre-assessments. The instructor attempted to motivate students to complete the video tutorials as a grade substitution for their score on the pre-assessment. They were given one week to complete these tutorials.

Finally, after the creation of the video tutorials, the students completed a conceptual and a coding post-assessments, which kept the same format of the pre-assessments. Following these post-assessments, students were given a feedback survey to evaluate the effectiveness of the activity.

4.3 Data Collection

We evaluated the effectiveness of the student-created video tutorials from two perspectives: student performance and perception. We evaluated student performance from their grades on 1) the pre-assessments (ST1), 2) video tutorials elaboration and completion, and 3) the final assessments (ST2). We evaluated their perceptions of the activities' structure and value of the video-tutorials in their learning processes. The student feedback survey had seven multiple choice questions, and an open-ended question for them to comment on their experience creating the video tutorials. The multiple-choice questions and their answer options are shown in Table 1.

Table 1: Post-Survey Multiple-choice Questions & Options.

Questions	Option
Q1: How stressed did you feel when completing the video-tutorial assignments?	A: Not stressed at all.
	B: A little bit stressed.
	C: Stressed.
	D: Very stressed.
Q2: How much did you learn completing the video tutorials?	A: I learned very much and gained knowledge of concepts and skills I did not have before.
	B: I learned, but not much given my prior knowledge.
	C: I did not learn at all.
Q3: How much did you have to prepare to create these video tutorials?	A: I studied and prepared all the topics asked in it.
	B: I barely studied for this activity.
	C: I did not study at all.
Q4: How much time did you spend creating the video-tutorials?	A: More than 10 hours of dedicated work.
	B: Between 8 to 10 hours of dedicated work.
	C: Between 6 to 8 hours of dedicated work.
	D: Between 4 to 6 hours of dedicated work.
	E: Between 2 to 4 hours of dedicated work.
	F: Less than 2 hours of dedicated work.

Table 1: Post-Survey Multiple-choice Questions & Options (cont.).

Questions	Option
Q5: How much did you study the course topics, in the process of creating the video tutorials?	A: I had to study very much.
	B: I had to study for it, but not too much.
	C: I did not have to study in the process. I was already ready for it.
Q6: Would you like to have a similar activity in the future?	A: Yes, I would like to.
	B: No, I won't.

4.4 Data Analysis

Data was analyzed using quantitative and qualitative techniques, considering frequencies of responses for the final survey, students' grades on both types of assessments (conceptual and coding), and students' feedback on the activity.

We analyzed the pre (ST1) and post assessment (ST2) grades to determine the percentage of improvement (Im) students made between them, referring to it as their percentage difference:

$$Im = 100 * (ST2.grade - ST1.grade)/5.0 \quad (1)$$

Their improvement score can be positive, neutral, or negative. We analyze the open-ended data on students' perceptions and experiences with the video tutorials using an inductive categorization method (Benavides & Restrepo, 2000), identifying and labeling those categories that come from the students' voices. These categories are used to find patterns between participants, and to identify what students most appreciated, and what they least liked.

5 FINDINGS AND RESULTS

Table 2 presents the students' grades on the conceptual pre and post-assessments. We found that students improved (Im) with the video tutorial creation, with a peak of 58.3% for one student [S2], a mean of 26,3%, and a mode of 33.7% [N=2]. Only two participants reported negative improvement outcomes [S3 & S11]. S11 did not create the video-tutorial.

Table 2: Student Outcomes – Conceptual Assessments.

Student ID	Pre-assessment (ST1) Scale: 0.0 to 5.0	Post-assessment (ST2) Scale: 0.0 to 5.0	Im [%]
S1 Video: Yes	1.67	3.8	42.7
S2 Video: Yes	2.08	5.0	58.3
S3 Video: Yes	3.33	2.9	-8.7
S4 Video: Yes	2.92	4.6	33.7
S5 Video: Yes	2.50	5.0	50.0
S6 Video: Yes	4.17	4.6	8.7
S7 Video: Yes	3.33	5.0	33.3
S8 Video: Yes	3.33	4.6	25.3
S9 Video: Yes	2.92	4.6	33.7
S10 Video: Yes	3.33	5.0	33.3
S11 Video: No	3.75	0.0	-75.0
S12 Video: Yes	2.08	4.6	50.3
S13 Video: Yes	2.92	5.0	41.7
S14 Video: Yes	2.50	5.0	50.0
S15 Video: Yes	3.33	4.2	17.3
S16 Video: Yes	2.92	4.2	25.7

Table 3 presents the student grades on the coding pre and post-assessments. We found that students improved (Im) with the video tutorial creation, with a peak of 66.7% for four students [S1, S7, S12 & S15], a mean of 29.2%, and a mode of 33.3% [N=7]. We can observe that only one student reported negative improvement outcomes [S11], being one of the two participants who did not create the instructional video as requested [S9 & S11].

As we can observe in both, Tables 2 and 3, the video-tutorials creation helped to improve their outcomes in conceptual and coding assessments. This distribution can be seen in Figure 1, which shows that students performed above the average for both types of assessments: coding and conceptual. Additionally, considering the strategy, students' performance improved significantly for those students who created both video tutorials [N=14, discarding S9 & S11].

Table 3: Student Outcomes – Coding Assessments.

Student ID	Pre-assessment (ST1) Scale: 0.0 to 5.0	Post-assessment (ST2) Scale: 0.0 to	Im [%]
S1 Video: Yes	0.00	3.3	66.7
S2 Video: Yes	3.33	3.3	0.0
S3 Video: Yes	3.33	3.3	0.0
S4 Video: Yes	3.33	5.0	33.3
S5 Video: Yes	0.00	1.7	33.3
S6 Video: Yes	3.33	3.3	0.0
S7 Video: Yes	0.00	3.3	66.7
S8 Video: Yes	0.00	1.7	33.3
S9 Video: No	1.67	3.3	33.3
S10 Video: Yes	3.33	3.3	0.0
S11 Video: No	1.67	0.0	-33.3
S12 Video: Yes	0.00	3.3	66.7
S13 Video: Yes	1.67	3.3	33.3
S14 Video: Yes	0.00	1.7	33.3
S15 Video: Yes	0.00	3.3	66.7
S16 Video: Yes	0.00	1.7	33.3

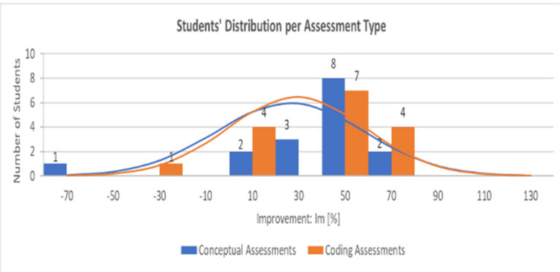

Figure 1: Students' Performance Distribution.

This is based on a paired-samples Student's t-test comparing grades between the pre-assessments and post-assessments ($t(14) = 5.8,\ p < 0.01$).

Table 4 presents the inductive categories identified and labelled according to the participants' voices and their experiences. Complementarily, Table 5 matches the students with these inductive categories, and their corresponding conceptual and practical (i.e., coding skills) reported improvement.

Table 4: Inductive Categories.

Category ID	Category Label
C1	The student gave positive feedback about the strategy.
C2	The student explicitly mentioned they learned.
C3	The student explicitly liked the strategy.
C4	The student explicitly said the strategy helped him/her prepare the topics.

Fourteen [N=14] out of sixteen students gave positive feedback about the strategy, with ten out of 14 of these students explicitly expressed liking the activity. Similarly, seven students explicitly expressed that they learned from the activity, and six students expressed that the pedagogic approach helped them to prepare the topics.

Table 5: Inductive Categories and Students' Results.

Student ID	Conceptual Im [%]	Coding Im [%]	C1	C2	C3	C4
S1	42.7	66.7	1	-	1	-
S2	58.3	0.0	1	1	1	1
S3	-8.7	0.0	1	-	-	1
S4	33.7	33.3	1	-	1	-
S5	50.0	33.3	1	-	1	-
S6	8.7	0.0	1	-	-	1
S7	33.3	66.7	1	1	1	
S8	25.3	33.3	1	1	1	
S9	33.7	33.3	1	1		1
S10	33.3	0.0	1	1	1	
S11	-75.0	-33.3	-	-	-	-
S12	50.3	66.7	1	1	1	-
S13	41.7	33.3	1	1	-	-
S14	50.0	33.3	1	-	1	1
S15	17.3	66.7	1	-	1	1
S16	25.7	33.3	-	-	-	-

Despite no existing correlation between the conceptual improvement index (Im1), the coding improvement index (Im2), and the inductive categories (see Table 6), students responded positively to the strategy, with opinions like:

"The video-tutorial schemes eliminate stress and allow the subjects to be prepared, more learning is achieved because there is commitment to make quality material".

Table 6: Performance and Perceptions Correlation.

	Im1 [%]	Im2 [%]	C1	C2	C3	C4
Im1 [%]	1.0	-	-	-	-	-
Im2 [%]	0.6	1.0	-	-	-	-
C1	0.6	0.4	1.0	-	-	-
C2	0.4	0.1	0.3	1.0	-	-
C3	0.5	0.5	0.5	0.2	1.0	-
C4	0.0	0.2	0.3	0.2	0.2	1.0

Figure 2 presents the students' responses for the question *"How much time did you spend creating the video- tutorials?"*. 56% [N=9] of the students indicated they spent less than six hours creating the video tutorials, with 25% [N=4] indicating they spent between two to four hours.

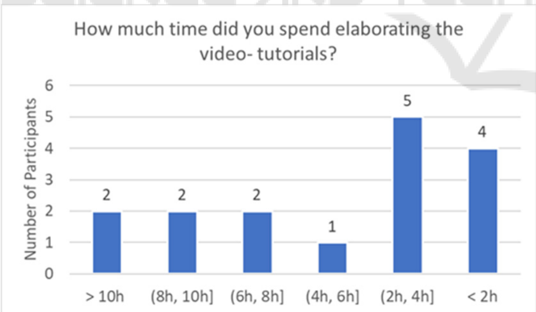

Figure 2: Responses on Time Demanded by the Activity (see Table 1 – Q4).

Figure 3 presents the students' responses for the question *"How much did you study the course topics, in the process of creating the video tutorials?"*. As it shows, 62% of the students indicated they had to study very much, in order to create the video tutorials. These responses suggest that the creation of video tutorials engages students in intentional study of course materials, helping them reflect on their learning processes.

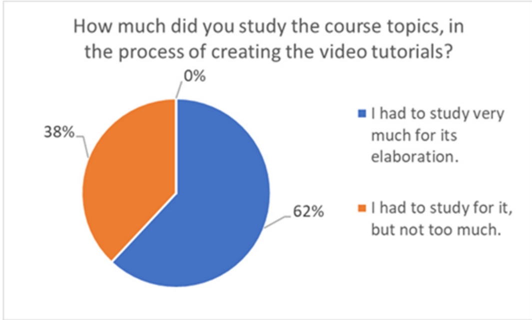

Figure 3: Responses on Students' Preparation.

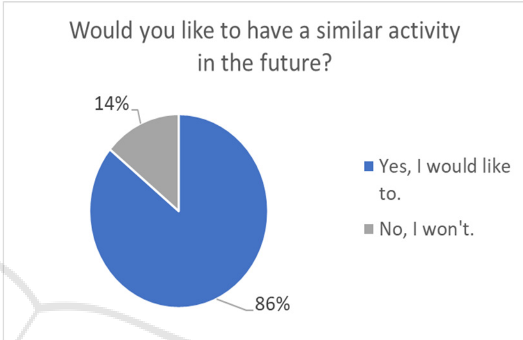

Figure 4: Responses on Students' Satisfaction.

Additionally, Figure 4 shows that 86% of students liked the activity, and that they would like to be evaluated with similar approaches in the future. This suggests that the activity engaged the students, improved their performance conceptually and practically (i.e., coding skills), and motivated them to learn.

6 CHALLENGES AND LESSONS

In this section we present the challenges we experienced while applying this strategy in our course, in addition to some lessons we learned in the process. The challenges we can highlight are:
Grading: The videos created by the students had an average duration of 20 minutes, varying due to the number of exercises they were asked to explain: it depended on how many questions they missed or got wrong in the pre-assessments. Grading the videos was an interesting pedagogical task that required significant time and effort from the instructor. We learned that we could face this challenge using co-evaluation strategies (e.g., blind peer-review). We also considered the potential for the activity to reinforce students' content knowledge as a result of reviewing the student-created video tutorials from

their peers. However, this is a hypothesis we still need to study.

Time: Based on the questions asked by the students in the process of creating the instructional videos, and on the perceptions presented in figure 2, we find that this strategy was time-consuming for the students, especially for those who were required to explain more exercises. From students' feedback, we learned that we might improve the strategy by limiting the number of exercises to explain. This can be achieved by letting the students pick a 1-2 exercises they considered were the most challenging from the pre-assessments.

7 DISCUSSION AND FUTURE WORK

Our findings suggest that student-created video tutorials help students to improve their performance, and that they can be used for conceptual or practical (i.e., coding) understanding. Likewise, they suggest that these kinds of approaches are engaging and meaningful to the students, considering the positive reactions and feedback students provided about their experience creating these instructional videos.

Our study provides evidence that using student-created video tutorials in a Functional Programming (FP) class, leads to significant learning on programming concepts and skills. We conclude that it is possible to use classroom models different from the traditional ones, with strategies that can empower and hold students accountable for their learning processes. More importantly, we experienced that students learn more when they have to explain or teach a certain topic. Instruction is a powerful learning tool that we, as CS Educators, should not ignore.

The study we report in this paper responds to a quasi-experimental design, adapted to the classroom environment described in section 4.1. Future work may involve using comparing the student-created video tutorial activity against traditional problem-solving practice assignments with two different groups of students in order to validate not only how using student-created video tutorials impacts students' learning processes, but also to measure the effectiveness of this activity.

Considering the challenges presented in section 5, we may use peer-reviewing as part of the strategy, not only to limit the time and effort in regards to grading, but also to evaluate differences between explaining and reviewing in terms of educational effectiveness.

Furthermore, we find it interesting to observe how students who have created content provide feedback to their peers', compared to students who are asked to review without having created any content on the topic. *How does the notional machine (Guzdial, 2016) of a student vary when s/he has been asked to instruct a certain CS topic?* This is a question we find interesting to explore within the CS ED community and for contexts regarding CS upper-level courses. This question is one of the next steps we find for the current study.

ACKNOWLEDGEMENTS

The authors would like to thank the students who actively participated in this experience. Their participation and feedback were essential for this study to succeed. We also want to thank Carlos Ortiz Buitrago for his contribution on the curricular structure of the Functional Programming course used for the study, and for his feedback on the learning strategies that have been adapted in it.

REFERENCES

Bain, K., 2004. *What the best college teachers do*. Cambridge: Harvard University Press.

Benavides, M. O., & Restrepo, C. G., 2000. *Métodos en investigación cualitativa: triangulación*. Red Revista Colombiana de Psiquiatría.

Cottam, J., Menzel, S., & Greenblatt, J., 2011. Tutoring for retention. Paper presented at the 42nd ACM technical symposium on Computer science education (SIGCSE '11), pp. 213-218. https://doi.org/10.1145/1953163.1953227

Feijóo-García, P. G., & Ortíz-Buitrago, C. H, 2018. The Godparent Plan: A Pedagogical Strategy for CS1 Accompaniment and CS2 Pedagogical Enhancement. *International Journal of Engineering Pedagogy (iJEP), 8*(1), 43. doi:10.3991/ijep.v8i1.7596

Fink, L. D., 2013. Creating significant learning experiences: an integrated approach to designing college courses. San Francisco: Jossey-Bass.

Frank-Bolton, P., & Simha, R., 2018. Docendo Discimus. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE 18*. doi:10.1145/3159450.3159466

Freire, P., 2012. *Cartas a quien pretende enseñar*. Biblioteca Nueva.

Gehringer, E. F., & Miller, C. S., 2009. Student-generated active-learning exercises. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education - SIGCSE 09*. doi:10.1145/1508865.1508897

Guzdial, M., 2016. Learner-Centered Design for Computing Education: Research on Computing for Everyone. Ed. John M. Carroll Synthesis Lectures on Human-Centered Informatics. Morgan& Claypool.

Kearney, M., & Schuck, S., 2004. Authentic learning through the use of digital video. In Proceedings of the Australian Computers in Education Conference. pp. 1–7.

Murray, D. E., Mcgill, T. J., Thompson, N., & Toohey, D., 2017. Can Learners Become Teachers? Evaluating the Merits of Student Generated Content and Peer Assessment. *Proceedings of the 2017 InSITE Conference*. doi:10.28945/3700

Porter, L., Bouvier, D., Cutts, Q., Grissom, S., Lee, C., McCartney, R., ... & Simon, B., 2016. A multi-institutional study of peer instruction in introductory computing. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (pp. 358-363). ACM.