Incorporating Game Design in an Introductory Computer Science Classroom

Darren Seifert
Department of Math and Computer Science
Minot State University
500 University Avenue West
Minot, ND 58707
Darren.Seifert@minotstateu.edu

Abstract

A common theme among incoming Freshment in Computer Science is an interest in games and game design. However, the mental toolset students need to begin developing a video game is typically built over their first two years in college. This model eventually leads to students acquiring the skills necessary to explore game development, but it does little to expose students to game design as a potential career path at a point when they are still determining if Computer Science is the right field for them.

This paper discusses an effort to incorporate basic game programming concepts into the curriculum of a cohort of first-semester freshmen at Minot State University. Using a pre-developed space-based game environment, students were able to explore developing the logic to control a single spaceship. The student-developed logic module was loaded by a client executable and connected to a server to 'compete' against other students entries.

1 Introduction

Minot State University requires all incoming first-year students to participate in a First-Year Experience (FYE) course. The primary goal of these courses is to support students transitioning to university life through unique learning opportunities, peer mentorships, and opportunities to engage with their communities. These FYE are always paired with one or two introductory courses from a discipline to form a common theme. So, for example, we encourage our incoming Computer Science students to take the First-Year Experience course paired with their introductory programming course, either Computer Science I taught in C++ or an Introductory Languages Course taught in Python that is not a required course for Computer Science students, but serves as a good starting course for those unfamiliar with programming. The focus of the Computer Science Specific FYE course is on introducing students to careers working with technology and coursework typically revolves around experiential activities such as building a computer from scratch and constructing and programming Lego robots. For the final project in this course this past semester we hoped to leverage the skills that students were learning in their introductory programming courses to explore concepts in the area of game design.

Video games have recently become a popular educational tool, however evedince of their impact on education is somewhat mixed. Beal found that students performed better when utilizing a game to gain knowledge about cancer [3]. However, Sward et. al. found that there was no difference between groups of students utilizing a game and those using flash cards to master concepts in peadatric medicine [4]. Interstingly though students expressed that they preferred learning with the video game. Yaman et. al. found that supporting instruction with worked out examples was of particular benefit to students learning difficult concepts [5]. However, there is little research available on the use of video games to teach game design concepts.

To achieve this goal, a game environment was created using Direct 2D and C++. Our game consisted of two spaceships trying to collect asteroids to score points. Asteroids were semi-randomly placed within the game space and randomly assigned a point value within a predetermined range. No points were scored until a ship was able to successfully collect an asteroid and return it to its same colored base. To add an extra measure of excitement, ships could shoot both asteroids and other ships. If a ship was shot it immediately respawned at its starting point, but would lose any cargo it had accrued. Asteroids that were shot were, after a small delay, replaced on the screen by an asteroid with a new point total and location.

Ships each started with a predetermined amount of ‘fuel’. Anytime a ship needed to fire a thruster, whether to speed up, slow down or turn fuel fuel was consumed. To keep competitors from just randomly shooting each time a shot was fired additional fuel was consumed. Fuel could be refilled with no point penalt if it was able to navigate back to its base, but if a ship ran out of fuel while flying around, it would just drift in the last traveled direction and could not shoot. The intent of the fuel system was that it would require students to take time to think of a strategy to periodically return to home refuel. They

couldn't just try and collect all asteroids and wait out the game, or only focus on shooting their enemy.

Students developed their ship's strategey during class time over approximately 12 class periods, with materials given to students to review between classes. The development of the students 'ships' culminated with a double elimination tournament held during the last class period of the semester. A screenshot of the game interface is included in Figure 1.

Figure 1: The Space Game Server

2 Implementation

As mentioned in the introduction, one focus of the FYE course at Minot State was to explore unique learning opportunities. With that in mind, we wanted to structure this assignment in a way that would allow students to visualize not only what their code was doing, but to observe ideas their classmates were trying. This would allow, over time, a cyclic process of development, idea sharing and refinement. Additionally, since this project required students to program in C++ and roughly half the class was learning Python at the time, the assignment needed to be designed so students could approach the problem with differing levels of knowledge. Due to these reasons it was decided that utilizing flipped classroom instruction would be appropriate for this assignment.

The Flipped Classroom method employs a host of multimedia tools such as video lectures, online readings and practice problems as homework. Class time is typically dedicated to group-based interactive problem-solving activities. It has come about largely as a result of the introduction of large scale extremely low-cost technological tools to education [1].

2.1 Course Software

To support the in-class development that accompanies the flipped classroom model, the game was broken up into three elements. The game server, shown in the introduction, was responsible for managing all game objects and visualization of the game. This server software ran continuously throughout each class period on a projector in front of the classroom. The client software, shown in Figure 2, ran on students desktop systems while they were working in the lab. However, students did not develop this software or have access to its source code. Due to the somewhat limited programming background of most first semester students, it was decided that giving them direct access to the client software source code might overwhelm some students and introduce additional issues.

Student development was limited to the creation of a .dll file referred to in class as the 'Space Brain' that implemented each ships decision model. When the client software was launched, it would load this .dll file to determine what choices the ship would make when connected to a game.

One issue with this software design model was that students had a limited ability to debug their code. When implementing the .dll runtime debugging tools typically used by our students are unavailable. To counteract this, a messaging system was implemented. Students could send messages in their code that would be displayed in the Brain Output section shown in Figure 2. This allowed students to keep a record of exactly what a value was at a specific point in their code.

In addition, much of the information available to students when writing their code was displayed in real time as it arrived from the server. This included information about the location of both the player, their current opponent, and any asteroids currently on the screen.

To aid in development, each client could disconnect from the game at any point without disrupting the game. This allowed students to leave a match and begin developing revisions to their code as soon as a problem was spotted. Their opponent would be able to continue observing their ship until the end of the match if so desired. A disconnected client caused the corresponding ship to drift with no further instructions, so if they were spinning in a circle when they disconnected their ship just kept spinning neither slowing down or speeding up. The winner of the match was determined by points much as if both competitors had remained in the match until the end.

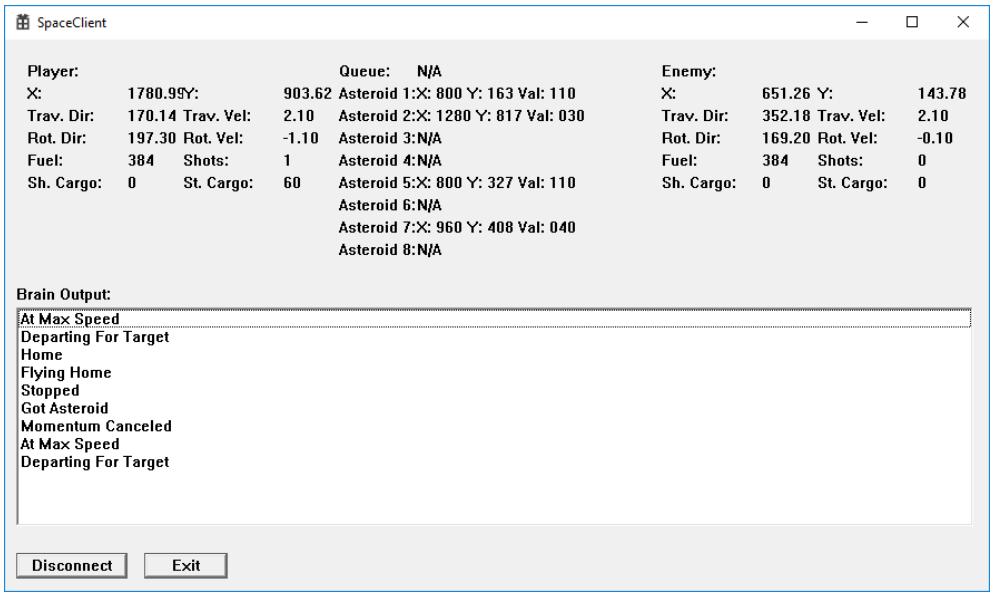

Figure 2: The Space Game Client

2.2 Course Materials

Flipped classroom instruction is a growing trend in higher education that has shown to have a positive effect on student learning and engagement [2]. For this assignment, students were given access to a series of reference materials to read over before beginning the process of creating their software. The first document students were given access to a document on differences between basic programming concepts in Python and C++. This document covered topics such as declaring variables, differences in conditional statements, loops, switch statements, and creating functions.

The second document details two data structures that students had access two from within the .dll they were composing. The first of these was responsible for detailing information about the current state of the game. Information such as the location of the ship, which direction it was moving, which direction it was facing, how fast it was moving and rotating was available for both the player's ship and the enemy. Additional information was included to help player's make decisions including the value of the asteroids in the inventory, the amount of fuel left, how much time was left in the game, and where their base was located among other items. This document also detailed a second data structure that enabled students to determine which action(s) their ship should take during the next iteration of the game loop. As mentioned, students were given very minimal set of control options including the ability to fire left, right and rear thrusters. Each of these elements was described along with relevant information about data type, range of values and what the values meant in the game.

Lastly, students were given a document detailing how to calculate the angle between two objects in the game given their location. This calculation is the basis for flying to asteroids, returning home, and firing at the enemy. This document detailed the logic behind calculating angles and how to implement the calculations in C++.

Over the course of the time spent on this project students were given additional materials to review before classes. The first of these explained what a game loop and how their code fit into the overall game model. As students only received new information from the server at the start of their code block, they couldn't implement code that, for example, looped and waited for their ship to reach a certain location on the screen as the position would never update. This limitation forced many students to change the way they approached the design of their program and was a crucial topic early in the process.

A bit further into the development process students were given materials that would help them understand how to implement key pieces of code that may be needed to develop a functional 'brain' for their spaceship. Topics of these updates included an explanation of how to establish a series of goals for your ship, how to rotate to a specific angle, and how to fly to a set of coordinates. The intention with these updates was to provide students with a small boost that would enable each of them to produce a capable program if they were able to turn the explanations into code.

3 Results

As this was the first time teaching this lesson, I chose to do a pre/post questionnaire with students rather than a formal evaluation of learning. After an explanation of the project concept and what exactly they would be developing, students completed a short survey to gauge their interest in video games, software development in general, and video game development specifically. As all FYE courses at Minot State are open to enrollment by students of any major this was an important first step. Each question was presented as a 6-point Likert Scale with 0 being no interest and 5 being very interested. The percentage of students expressing a strong interest (options 4 or 5) for these questions are displayed in Figure 3.

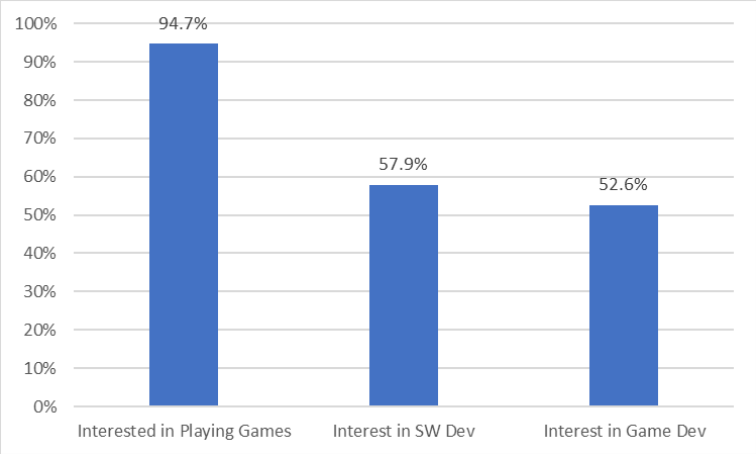

Figure 3: Interest in Gaming/Game Development (n=19)

As can be seen in Figure 3, this cohort of students had a strong interest in playing video games and, most likely due to the makeup the class, a somewhat lesser interest in both software development and game development specifically. Students were also asked whether they felt they felt they understood the process of developing video games, had any experience developing games, and whether they thought this proposed project would be beneficial to them. As the previous set of questions identified that a portion of the class had a limited interest in software development as a potential career path, the response here were further broken down into groupings that had expressed an interest in software development and/or video game development by selecting a 4 or 5 in that category.

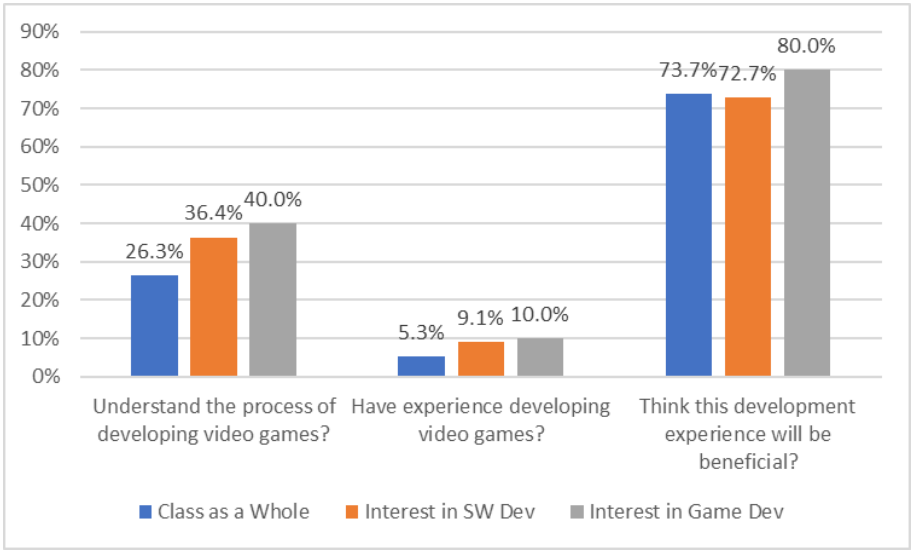

Figure 4: Experience in Game Development

Figure 4 shows that this project had the potential to be of benefit to students in the course across the board. Interestingly, even though 52.6% of the class felt that they had an interest in game development as a career path, 73.7% of the class felt that they would benefit from undertaking this project.

Lastly, a post survey was conducted as students were nearing the final competition date. Students were asked three primary questions that were based on the same 0 to 5 Likert scale. Students were asked if they found the development experience beneficial, if the difficulty of the project was appropriate for their skill level, and if they felt like had gained knowledge by undertaking the project. Figure 5 illustrates the result of this survey.

Overall the results of this survey showed that students both found the development process beneficial and that a majority of them gained knowledge. Responses on the question of whether the project was appropriate for their skill level were a bit less positive. It is unknown exactly why this is, but some analysis is provided below.

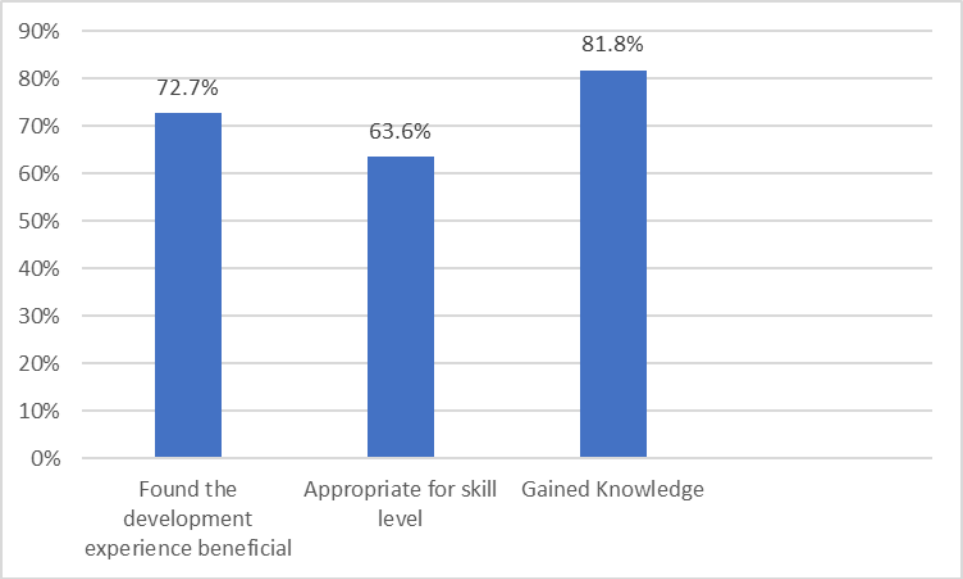

Figure 5: Post Lesson Survey (n = 11)

4 Lessons Learned

In addition to the quantitative data gathered, I talked with students about the process as we were reaching the final competition. Several students mentioned that the development process would have been a struggle if they were given only the description of the goal and told to implement the software. Although this is probably typical of most assignments, the primary goal here was to give students an opportunity to explore a career using knowledge they had attained in the parallel courses.

With the use of the flipped classroom style I was able to observe much of the development firsthand and it seemed that the struggle for students was based around how to make a game loop work with the somewhat limited controls they had available on the ship. For example, several groups tried something like firing the rear thrusters on the ship and then starting a loop to wait until the x value reached a certain point. However, the way the game was developed the student's module had to finish processing for new location data to be delivered from the server, so the x value was never delivered. A more sophisticated client that processed and updated game data from the server independently from the code that constructed outgoing packets might have helped with this issue and allowed student to use more sophisticated repetition structures.

Students also seemed to struggle with seemingly simple actions like turning to a specific angle. The client code was written so you could only fire thrusters in bursts that might be much shorter than the speed you were travelling. This meant students had to determine exactly how many bursts were needed to stop their ship at a given point as well when to start delivering them, which led to some frustration on the part of students.

A frustration that I experienced was that the single server model occasionally led to testing congestions. Matches were limited to 1 minute during the development phase to keep the queue of students from building up. This worked, but there were still frequently 5 minute breaks between when a student compiled their code and when they were able to determine if it worked the way they intended. In the future, I may look to run multiple servers simultaneously.

5 Conclusion

The introduction of this lesson into our Computer Science First Year Experience curriculum has been, overall, a positive experience for our department. Some minor revisions are certainly needed for future iterations of the game to match the skill level required to develop a game combatant, but overall this was a positive experience for both myself and the students. Students could frequently be seen taking breaks from programming their own ships to observe a battle happening at the front of the classroom and adopting their code to overcome a difficult loss. The final tournament was quite entertaining with several high-pressure moments. I look forward to using this lesson again next year and possibly introducing new game concepts in the future.

References

[1] Clyde Herreid, Nancy Schiller. “Case studies and the flipped classroom.” *Journal of College Science Teaching* vol. 42, no. 5, pp. 62-66, May 2013.

[2] Scott Freeman, et al., “Active learning increases student performance in science, engineering, and mathematics”, Proceedings of the National Academy of Sciences of the United States of America, vol. 111, no. 23, pp. 8410-8415, Jun. 2014.

[3] Ivan Beale, et al., “Improvement in cancer-related knowledge following use of a psychoeducational video game for adolescents and young adults with cancer”, Journal of Adolecent Health, vol. 41, no. 3 pp. 263–270, Sep. 2007.

[4] Katherine Sward, et al., “Use of a web-based game to teach pediatric content to medical students”. Ambulatory Pediatrics, vol. 8, no. 6, pp. 354–359, Nov 2008.

[5] Malek Yaman, et al.. (2008). The effects of instructional support and learner interests when learning using computer simulations”, Computers & Education, vol. 51, no. 3, pp. 1784–1794, Dec. 2008