Game Development as a Pathway to Information Technology Literacy

Mark Frydenberg
mfrydenberg@bentley.edu
Computer Information Systems Department
Bentley University,
Waltham, MA 02452, USA

Abstract

Teaching game development has become an accepted methodology for introducing programming concepts and capturing the interest of beginning computer science and information technology (IT) students. This study, conducted over three consecutive semesters, explores game development using a gaming engine, rather than a traditional programming language, as a means not only to introduce programming concepts, but also to promote the development of information and communications technology (ICT) literacy skills among first-year business students. The paper argues that in addition to learning programming concepts, completing the steps involved to develop and publish an original game requires students to demonstrate a variety of ICT skills. To be successful, they must be proficient at creating and editing multimedia, interacting with multiple operating systems and mobile devices, performing research online, transferring files from one machine to another, and uploading the files for their games to an app store and the web.

Keywords: game development, ICT skills, information technology literacy, app development, programming.

1. INTRODUCTION

Teaching technology literacy to non-technical students has changed greatly since John Kemeny and Thomas Kurtz, creators of the BASIC programing language at Dartmouth College, introduced the notion of computer literacy in 1964. Kemeny recognized that that "someday computer literacy will be a condition for employment, possibly for survival, because the computer illiterate will be cut off from most sources of information." (Kemeny, 1983) Computer literacy then was achieved by using time-shared teletype terminals connected to a college mainframe to create simple BASIC programs as a way to develop algorithmic thinking. "By making the BASIC environment so friendly, [Kemeny and Kurtz] created a safe place for people to play and explore. The computer game movement came from BASIC. People shared their games, long before there were networks, by printing the programs [in computer magazines] for others to enter in and enjoy." (Claburn, 2014)

Today, the computing tools, languages, and devices have evolved, but creating games as a pathway to achieving technology literacy has remained constant. The author's previous research (Frydenberg, 2015) examines how game development provides an authentic learning experience. This paper examines both programming and general information and communication technology (ICT) concepts that students may learn as a result of developing and publishing their own computer games to an online app store and the web.

Computer literacy has expanded to include digital literacy and ICT literacy. The ability to write programs is only one possible outcome of technology literacy today. Equally, if not more important than knowing how to program is for today's students to have proficiency with

personal productivity and collaboration tools, email, the World Wide Web, social media, mobile devices and the cloud, and to know when and how to use these tools responsibly. (Frydenberg & Press, 2009) (Press, 2011) "Digital literacy comprises effective problem-solving skills, critical thinking and communication skills, creativity, and self-regulation along with an understanding of culturally and contextually based practice in the use of, and engagement with digital technologies." (Burton, Summers, Lawrence, Noble, & Gibbings, 2015, p. 152)

The Partnership for 21st Century Living (2015) describes ICT literacy as the ability to apply technology effectively, including:

- Use technology as a tool to research, organize, evaluate and communicate information
- Use digital technologies (computers, PDAs, media players, GPS, etc.), communication/networking tools and social networks appropriately to access, manage, integrate, evaluate and create information to successfully function in a knowledge economy
- Apply a fundamental understanding of the ethical/legal issues surrounding the access and use of information technologies.

Game Development and Technology Literacy

A current trend in both high school and college introductory computing courses is to incorporate the use of visual development tools to enable students to build apps and games with little or no programming required. (Catete, Peddycord III, & Barnes, 2015), (Abelson, Mustafaraj, Turbak, Morelli, & Uche, 2012) (Thomas & Blackwood, 2010). Popular game development tools include Alice (Carnegie Melon University, 2015), Scratch (Lifelong Kindergarten Group at the MIT Media Lab, 2015), App Inventor (MIT, 2015), and Game Maker (YoYo Games Ltd., 2015). One reason for this interest among faculty is to "better attract and retain students in the computer science major." (Bayliss, 2009, p. 337)

From an information technology educational perspective, creating games presents students with opportunities to be creative, and demonstrate their proficiency as able participants in a world based on technology literacy skills. Games allow students to turn their ideas into real applications that they can play on their own laptops and mobile devices.

While game development in and of itself is a worthwhile exercise for students studying programming, this paper argues that information technology and business students also can develop additional ICT literacy skills and understand IT concepts through the process of creating computer games.

Research Questions

This study shares results after offering a game development experience to beginning information technology students. The study addresses the following research questions:

- How does game development fit within the topics of an introductory information technology course?
- Which ICT skills do students find important while creating computer games?
- Does creating and publishing a computer game allow students to apply their knowledge of IT concepts and develop ICT skills?

2. LEARNING IT CONCEPTS THROUGH GAME DEVELOPMENT WITH CONSTRUCT 2

This study describes the use of Construct 2 (Scirra Ltd., 2015), a game development tool for creating two-dimensional games, among first-year business students in an information technology course. Construct 2 does not require learning a programming language and allows for rapid prototyping of computer games, so it is quite suitable for beginners to create a functional game with no prior programming skills. Construct 2 includes a development environment to design a game and specify its rules of play, and a Game Creator to export completed games as web or mobile apps. Construct 2 was chosen for this study because its Game Creator generates HTML5, CSS3, and JavaScript files that can be published on the web, as well as native app packages for Windows Phone, iOS, and Android, that can be published on the Windows, Apple, and Google Play app stores and marketplaces.

The sections that follow discuss the programming concepts to which students are exposed while developing their games, and the IT skills that they develop while creating the game's multimedia elements, and publishing their games on the web and to an online app store.

Learning Programming Concepts through Game Development

To develop a game in Construct 2, one begins by designing the game's screens, layouts, and objects (characters, platforms, projectiles, sounds, and so on), and their properties. Figure 6 shows the Krazy Kopter game during play. The game was created by a student in this study. The object of the game is to fly the helicopter without hitting an upper, lower, or middle obstacle. The heights of the obstacles are generated randomly.

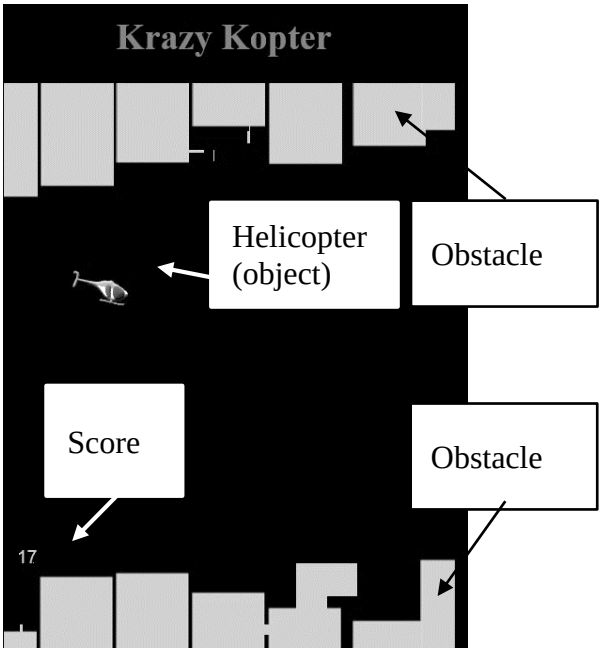


Figure 6. The object of Krazy Kopter is to fly the helicopter and avoid the obstacles.

Appendix 1 Figure 1 shows the game's start screen, and the how the game's play layout is designed using Construct 2. The developer can assign properties to each object in the game.

Expressing objects and their properties in terms of real world objects, such as a helicopter's position or speed, is a natural way to introduce the concept of objects. As with courses introducing objects through Alice and Scratch, Construct 2's environment allows students to develop "a good sense of objects." As Stephen, Dann, and Pausch write, "Everything in a student's virtual world is an object!" (Stephen, Dann, & Pausch, 2003, p. 193). The main characters of their games, whether flying helicopters or flapping birds, are all objects. Figure 7 shows some of the properties of the helicopter object.

Construct 2 uses a visual syntax to describe events that occur during game play (such as screen taps, key presses, and collisions) and the actions to take in response to them. Identifying these steps writing the instructions introduce students to basic input/processing/output and promote algorithmic thinking. Students learn that they must initialize variables at the start of the game (such as setting constant values for speeds or starting values for the game score), and describe the steps to follow when events are detected at each iteration of the play loop, or "tick" of a timer, that continues until the game is over.

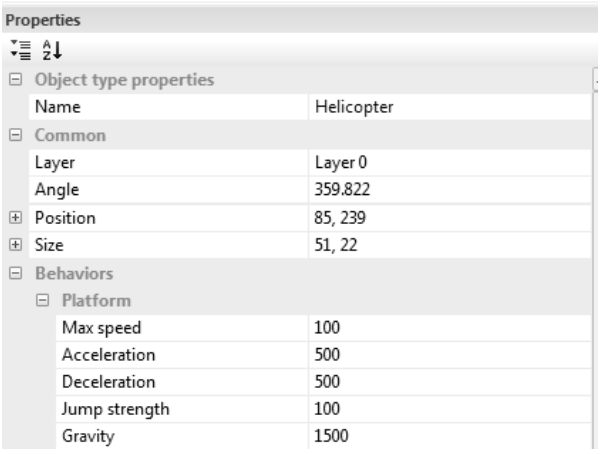


Figure 7. Expressing properties in Construct 2.

Students find that expressing rules in natural language is often helpful before constructing them using Construct 2. For example, one could summarize the conditions for ending the game as "When the helicopter collides with an upper, lower, or middle object, return to the Start screen (because the game is over)." Figure 3 shows how to express this rule in Construct 2.

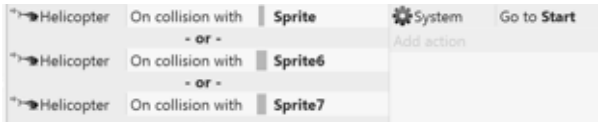


Figure 8. Specifying the end-of-game rule in Construct 2.

Figure 2 in Appendix 1 shows all of the rules for the Krazy Kopter game, annotated with brief comments describing what they do. Table 4 below summarizes the programming concepts that are used the Krazy Kopter game design. Specifying the game play in Construct 2 introduces students to fundamental programming constructs, including creating instances of

objects, constants, program logic, and transfer of control.

Table 4. Programming Concepts Reflected in using Construct 2 to Design the Game shown in Appendix 1 Figure 2

Step	Programming Concept Reflected
0	Initialize constants and variables.
1	Read input (from touch or keyboard)
2	Looping and iteration
3	Create new instances of an object
4	Create new instance of an object
5	Decision logic
6	Destroy objects no longer in use
7	Logical comparison, increment variables, display output

Learning Information and Communications Technology Skills and Concepts through Game Development

In addition to learning programming concepts, students apply their ICT knowledge to this project as they interact with a variety of multimedia content and apps to complete their games. By using Construct 2 to create a game that runs as a web app or a Windows app, students experience the prevalent trend in software development today of creating apps that run on multiple devices and platforms. (de Andrade, Albuquerque, Frota, Silviera, & da Silva, 2015) Students see the relevance of creating software that runs on their computer or mobile device.

Through the process of making games, students demonstrate proficiency with and understanding of several technology-related tasks and concepts.

3. IMPLEMENTATION OF STUDY

The game development study described in this paper took place over three semesters (spring 2014, fall 2014, and spring 2015), in three different sections of a technology concepts course at Bentley University, a business university in the United States. IT 101 (Introduction to Information Technology and Computing Concepts) is a required course for all first year students. IT 101 teaches technology literacy skills including maintaining laptops, using mobile and desktop productivity software and apps, navigating the World Wide Web, developing simple web sites with HTML, interacting with operating systems. The course also introduces IT concepts including storing and communicating digital information, multimedia formats, Internet protocols, digital storage and the cloud, computer and networking components, mobile devices, and building technology solutions.

Table 5. ICT Concepts and Skills Evident in Game Design and Publishing Tasks.

Development Task	ICT Concept / Skill
Design and develop a 2-D game using Construct 2.	Programming logic and languages, objects, instances, variables, transfer of control, iteration
Create or locate sound and image files for use in the game.	Multimedia files, downloading, resizing, file compression, Creative Commons, Online tools, search.
Export the game from Construct 2 for web and Windows Store.	Files and Directories, Compile.
Publish HTML version of game on website.	HTML5, create pages with <embed> code, FTP files to a web server
Play game on smartphone or tablet.	Native vs web apps; Shortcuts.
Copy game files from user's (Windows 7) laptop to a Windows 8 computer to submit to Windows Store.	Copy files from one device to another. Flash memory, cloud storage. Interact with multiple operating systems.
Prepare images for game's logo, store logo, and splash screen.	Basic image editing: resize, crop
Digitally sign files for submission to the Windows Store.	Digital Signatures
Test game in emulator as part of submission process to Microsoft store.	Emulator, Software deployment.
Respond to errors in certification reports if game does not pass submission.	Software development life cycle, debugging, software revisions

In-Class Training

Students enrolled in three different sections of the course taught by two different instructors participated in the project each semester:

- An Honors section, for students enrolled in the honors program, chosen because of their high scholastic abilities,
- An Accelerated section, offered to students who self-selected to be this section because of their interest in technology, and
- A standard section open to all students.

Each section met for two 75-minute sessions per week.

Students installed Construct 2 on their laptops prior to the first training session. At the first training, they followed a demonstration to create a simple 2-D platform game. Completing this exercise introduced game elements such players, shooters, and layouts, user interface elements for keyboard and touch input, detecting collisions of objects, and creating and destroying objects. During the second training session, students learned to enhance their games with music and sounds, keeping score, and end-of-game logic.

Microsoft sponsored the gaming project and provided support from an academic technology evangelist to lead the training sessions in each class, offering consistent delivery across all sections. The instructors, student tutors, and former IT 101 student volunteers who had completed the gaming contest in prior semesters, assisted students during the demonstrations and exercises in their classrooms. Student tutors and the Microsoft evangelist provided additional support outside of class during designated hours in the CIS Sandbox, a campus tutoring and social learning space.

As an additional incentive, Microsoft donated prizes to award to the creators of the best games at a games party held at the conclusion of the project.

Student Assignment

The gaming project spans a period of four weeks during the semester, so students have time to gain basic proficiency with Construct 2, design their own games, and get lots of help from tutors and peers. This is one project where the process of completing it is as important as the final result, if not more, when determining a grade. Dividing the project into smaller steps seems to make the process more manageable.

The gaming assignment required students to create an original game, or adapt the game they worked on in class and add new features. They could collaborate with a partner in the design process, but had to create their own individual games (which could be similar in game play) for this assignment. Students had to submit their games to the Windows Store, and publish them to their websites created earlier in the semester. Appendix 2 Table 1 lists several online resources shared with students for use in completing their assignment. The complete description of the assignment is provided in Appendix 3.

Tutors and former IT 101 students played and reviewed all of the games the weekend after they were due, evaluating them based on their originality, playability, complexity, and aesthetics. The top seven games were selected from these reviews. All students then were invited to a games party where they could play all of the games and vote for their favorites. The winners were announced, and prizes awarded.

4. STUDENT SURVEY

Approximately 269 students enrolled in 9 sections of IT 101 over a three-semester period participated in this study. At the end of each of the three semesters, a survey was offered to each class participating class. Responses were received from 239 students; 64% were male, 36% female. Not all students replied to all questions used in this study.

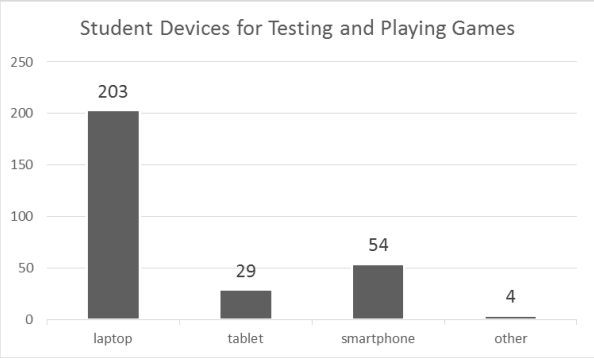


Figure 9. Devices used for testing and playing games.

When asked if they created any type of software applications in the past, 85% of the 208 responses indicate that they had not; 15% had created applications before, including simple games, web pages, Java, or C++ programs. These results suggest that creating games to play on their computers and mobile devices was a new experience for most of the students, and certainly, deploying them to two platforms was new for all of the students. Figure 9 shows the devices students used to test or play their games. It is no surprise that 98% used laptops (since

each student has one), followed by smartphones (26%), tablets (14%), and other devices (4%).

Appendix 1 Figure 3 shows attitudes toward technology among the students enrolled. As enumerated in Table 2 above, creating a game allows students to experience many IT concepts beyond programming, and demonstrate ICT skills as they complete it.

Figure 5 summarizes student perception of the importance of several ICT skills. Students used a scale from 1 (not important) to 5 (very important) as they rated the importance of each to the gaming assignment. Values shown are the average ratings for each skill.

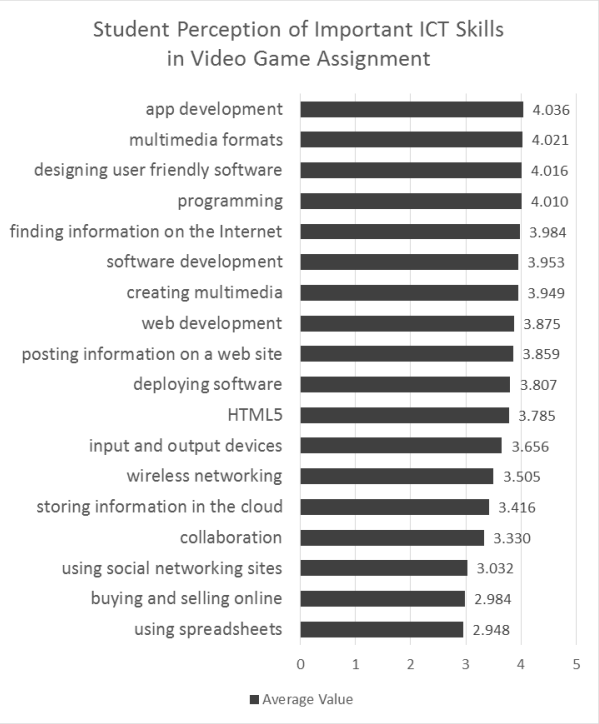


Figure 5. Student perception of important ICT Skills.

Students recognized that creating multimedia, knowledge of multimedia formats, programming, software, app, and web development, HTML5, deploying software, and designing user friendly software as more important to creating their games. These were skills they had to apply while completing their games. Students found the remaining ICT skills to be less applicable to this project.

Appendix 1 Figure 4 summarizes how students experienced the game project as a way to apply their IT knowledge and skills. 78% of the 208 students responding to this question students agreed or strongly agreed that the project enabled them to learn about app development; 55% agreed or strongly agreed it helped them develop skills in logical thinking; 41% agreed or strongly agreed it let them use IT 101 skills; 42% agreed or strongly agreed that it captured their interest in CIS; 63% agreed or strongly agreed that the gaming assignment was an interesting way to apply IT concepts. These results suggest that the project was an effective way for students to develop ICT skills and apply their IT knowledge.

Said one student: "This project introduces very important programming concepts to IT 101 students. It may not be for everyone, especially those who are not interested in developing applications. " Another student commented, "It is obvious how [creating a game] developed my skills working with multimedia as that is how I added graphics and such, but it helped me understand how a digital game interacts with a website and then both of those interact with a server, when I posted it. A very helpful lesson!" A third student commented on the business skills required. "The main thing I kept running into was problem-solving and collaboration with others. These are two very helpful real-world skills, so it was good that they were integrated in the project."

5. CHALLENGES AND LESSONS LEARNED

The process of deploying a game to two different platforms (the web and the Windows Store) was a challenge for some students.

Publishing Games to the Web

Most students were able to follow the instructions provided to export their games from Construct 2 for deployment to the web. Students added pages to their web sites to include screenshots and descriptions of their games, the embed code provided from Construct 2, and a link to a web page for playing the game full screen.

To simplify the process of publishing games to their websites, students were instructed to save the HTML5, JavaScript, CSS3, and other files generated by the Game Creator in a Game folder within their website. Some students forgot this step, and saw that their website's index.html file was unexpectedly overwritten with the game's index.html file because it, and the other game files were not saved in the correct location.

Once published to their websites, students could navigate to their game's web address using a browser on their laptop or mobile device to play the game. By adding this location as a shortcut on their devices' home screens, their games became original, fully-functioning mobile web apps. Students were able to "experience mobile computing as creators, not just consumers. This goes beyond the issues of teaching programming or computational thinking, to fundamental ideas about how a citizenry can be empowered in the age of information." (Abelson, Mustafaraj, Turbak, Morelli, & Uche, 2012, p. 39)

Publishing to the Windows Store

Students also exported their Construct 2 games as native apps to submit to the Windows Store. While most students did not have their own Windows 8 laptops or mobile devices on which to run their games as Windows 8 apps if published successfully, the process of submitting games to the Windows Store still has great educational value.

Games must pass strict tests in order to be published in any vendor's app store. The submission process verifies that the correct files are present, that the game or app runs without crashing, and that the submission contains the required documentation. When submitting to the Windows Store, Microsoft testers play each game, testing for touch input and other features.

Publishing a game to the Windows Store was by far the biggest challenge to students each time the project was offered. The hardest part for many students turned out to be following the multi-step instructions provided. Even though screen-by-screen instructions showed what to do in Visual Studio, many students were frustrated quickly. This is likely because they students had no previous familiarity with Visual Studio and because they had to user computer lab machines running Windows 8 to do so. Most student laptops and all university computer labs were still running Windows 7, but the version of Visual Studio required for submitting their games to the Windows Store ran on Windows 8. Windows 8 was installed on four computers in the lab, but that still was not sufficient. The instructors increased this to eight computers the following semester, which was an improvement, but still caused long waiting lines to access a computer from which students could submit their games to the Windows Store.

Once their games were accepted, however, many students took pride in their accomplishments, some noting that it is a resume booster to have a game of their own creation published to the Windows Store.

In some cases, interpreting comments on a report if a game did not pass certification also was challenging. For example, a content and age rating must be specified at the time of submission, but a student neglected to do so. There was nothing wrong with the game itself, but the information needed for publication was incomplete.

From an IT literacy point of view, following the process of publishing an app to an online store gives students a better appreciation of the steps that developers of the apps they use every day must follow, and adds a real-world learning lesson, beyond programming skills, to the project.

Microsoft requires a free Windows developer account to submit games to the store. Credit card verification was required for identifying a user's country of origin in order to submit a game to the store. Not all students had their own credit cards, which added a level of complexity. Purchasing a few pre-paid Visa cards was an awkward, but sometimes successful workaround. Microsoft has since removed this requirement.

In subsequent semesters, the instructors created one "class account" for each section for students to use when submitting their apps. Using a shared account for the entire class simplified the instructors' task of tracking the status of the games that students submitted. By signing to the Windows Dev Center at http://dev.windows.com with the class credentials, the developer's dashboard shows the current status of all apps associated with the account, as shown in Figure 6.

Figure 6. The Windows Dev Center shows the status of submitted apps.

Microsoft's lifting the credit card requirement combined with the shared class account dramatically increased the number of students who were able to publish their games on the Windows Store successfully, as shown in Figure 7. The first time through, the instructors modified the assignment's publication to the Windows Store requirement to become extra credit because it was such a burden to do. By the third iteration of the project, attempting to publish to the store was required.

Figure 7 summarizes the percentage of students each semester who reported that their games were published successfully during each semester of this study, along with those who tried and gave up, and the small number who didn't bother to try.

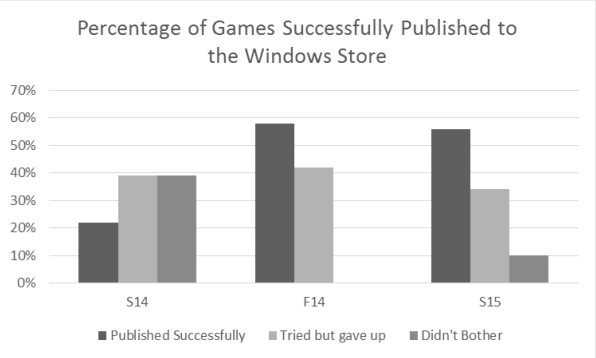


Figure 7. Games successfully published to the Windows Store.

Student Attitudes

Despite the fact that many students found creating their games to be challenging, and publishing it to the Microsoft Store to be frustrating and overwhelming, students took pride in the games they created. Over 100 of the 130 students who responded affirmatively to the question "Did you show your game to anyone else?" also added comments, such as "they thought it was cool that I could do that…," "they were impressed," or "my friends were amazed that my IT class was making games." Students worked hard at the assignment, 50% spending more than 6 hours on it, as shown in Figure 8.

Said one student, "While I created my game, I gained knowledge in the software development process and understood how game designers built their own games."

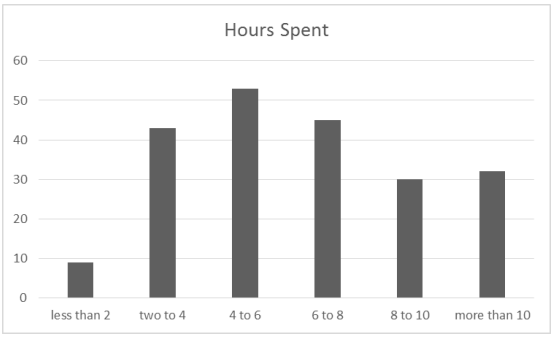


Figure 8. Hours Spent on the Game Assignment.

6. OBSERVATIONS AND CONCLUSIONS

While students were required to create their own games, they were encouraged to work in pairs to share ideas and review each other's work. This proved to be effective as students were able to support each other in their learning and show each other how to accomplish various tasks they remembered but their partners did not. Students used appropriate online ICT tools to share screens, graphics, or other files with each other.

The games that students created were completely functioning apps that they could run on their computers and mobile devices. Many featured touch input, music, score keeping, animations, and other characteristics found in real games. Not only did students create games, they published their games on two different platforms so others could access them on their devices. Doing so required aptitude to identify and solve basic tasks on their computers and devices. All students, regardless of their academic designation as characterized by the section of IT 101 in which they were enrolled, completed at least a basic game.

Completing the steps to publish a game successfully on the Windows Store required a level of support from student tutors, faculty, and the Microsoft evangelists that was not sustainable, and led to frustration among several students. During a later semester, the instructors replaced the requirement of publishing games as native apps to the Windows Store with the simpler task of publishing them as web apps to the Microsoft Azure cloud platform. By publishing to the cloud, students had to configure a virtual FTP server, make use of cloud services, and interact with the Azure portal. In some cases, students saw how the geographic location of the server on which their game was stored impacted performance. They learned that the consumer cloud storage tools with which they were familiar,

such as Google Drive, Microsoft OneDrive, and Dropbox, were useful to store and publish files to the web, but a different solution was needed to publish applications to the web.

Quality of games was mixed across all sections. While all students completed the same training, the games they created were very different. Even variations on the same in-class example generated a variety of results and outcomes. Many students adapted online tutorials to create simple adventure, platform, shooter, and other genres of 2D games. A games contest offered added incentive to produce a high quality final product.

7. ACKNOWLEDGEMENTS

The author acknowledges Michael Cummings and Gavin Baumann, Microsoft Academic Evangelists for their continued dedication to this project. Thanks to Bentley University student Brandon O'Connell for sharing screenshots of his Krazy Kopter game.

8. REFERENCES

Abelson, H., Mustafaraj, E., Turbak, F., Morelli, R., & Uche, C. (2012). Lessons learned from teaching App Inventor. *Consortium for Computing Sciences in Coleges, 27*(6), 39-41.

Bayliss, J. D. (2009). Using games in introductory courses: tips frm the trenches. *41*(1), 337-341.

Burton, L. J., Summers, J., Lawrence, J., Noble, K., & Gibbings, P. (2015). Digital Literacy in Higher Education: The Rhetoric and the Reality. In M. K. Harmes, *Myths in Education, Learning and Teaching: Policies, Practices and Principles* (pp. 151-172). Pallgrave Macmillan.

Carnegie Melon University. (2015). *Alice*. Retrieved from http://www.alice.org/index.php

Catete, V., Peddycord III, B., & Barnes, T. (2015). Augmenting introductory Computer Science Classes with GameMaker and Mobile Apps. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 707-709). New York, NY: ACM.

Claburn, T. (2014, 04 30). *When BASIC Was Young: Great Memories*. Retrieved from http://www.informationweek.com/strategic-cio/executive-insights-and-innovation/when-basic-was-young-great-memories/d/d-id/1234939

de Andrade, P. R., Albuquerque, A. B., Frota, O. F., Silviera, R. V., & da Silva, F. A. (2015). Cross platform app: a comparative study. *International Journal of Computer Science & Information Technology, 7*(1), 33-40.

Frydenberg, M. (2015). Creating Games as Authentic Learning in the Information Technology Classroom. *Proceedings of the International Conference E-Learning 2015* (pp. 113-120). Gran Canaria: IADIS.

Frydenberg, M., & Press, L. (2009). From Computer Literacy to Web 2.0 Literacy: Teaching and Learning Information Technology Concepts Using Web 2.0 Tools. *Infomation Systems Education Journal, 8*(10), 18.

Kemeny, J. G. (1983). The Case for Computer Literacy. *Daedalus, 112*(2), 211-230.

Lifelong Kindergarten Group at the MIT Media Lab. (2015). *Scratch*. Retrieved from https://scratch.mit.edu/

MIT. (2015). *MIT App Inventor*. Retrieved from http://appinventor.mit.edu/explore/

Partnership for 21st Century Living. (2015). *Framework for 21st Century Living*. Retrieved June 1, 2015, from Partnership for 21st Century Living: http://www.p21.org/about-us/p21-framework/350-ict-literacy

Press, L. (2011, October 24). *IT literacy – evolution, curriculum and a modular e-text.* Retrieved March 3, 2015, from http://som.csudh.edu/fac/lpress/presentations/mahyar.docx

Scirra Ltd. (2015, 06 01). *Construct 2*. Retrieved from Scirra: https://www.scirra.com/construct2

Stephen, C., Dann, W., & Pausch, R. (2003, January). Teaching Objects-first in Introductory Computer Science. *ACM SIGCSSE Bulletin, 35*(1), 191-195.

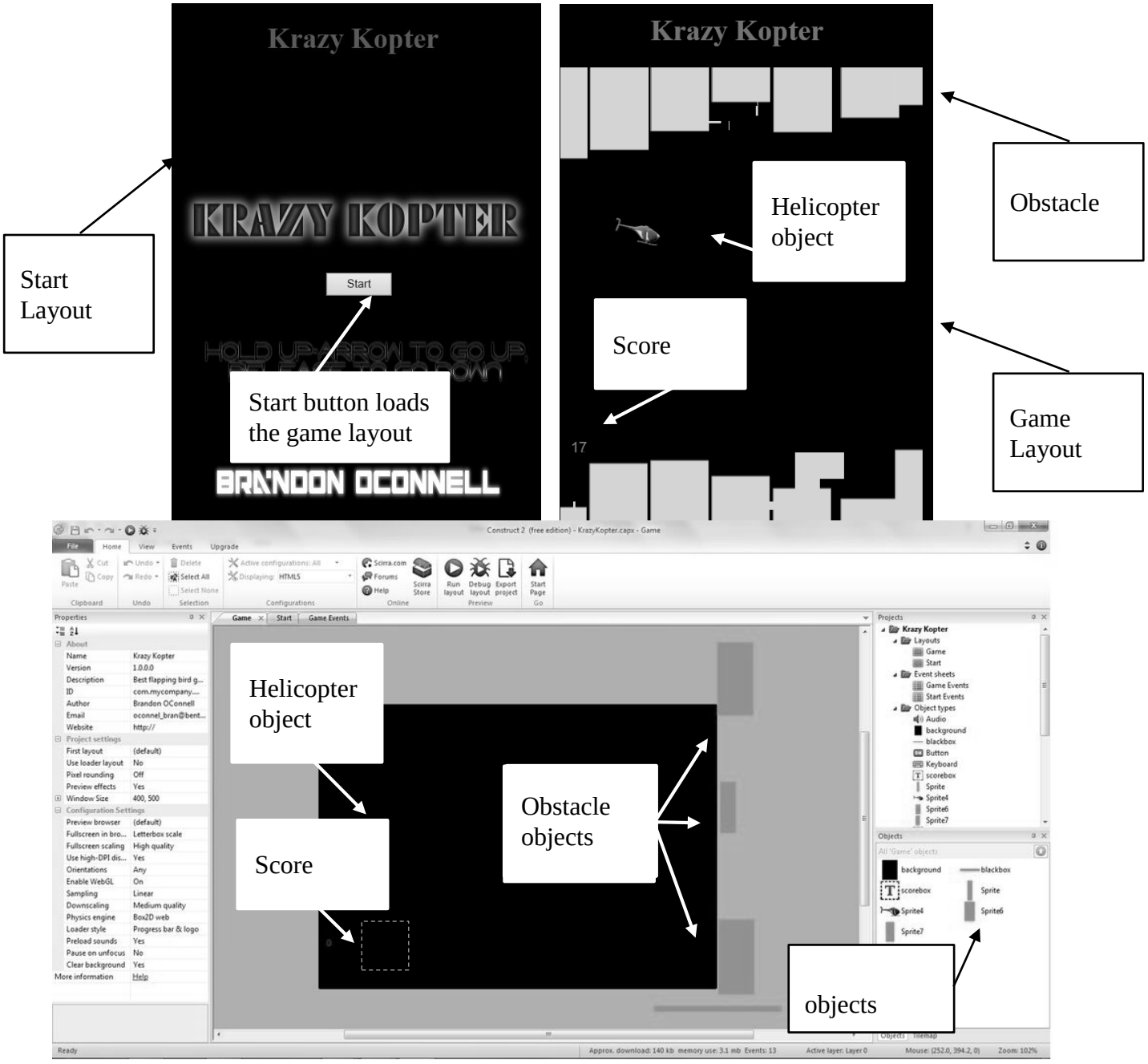
Thomas, J. D., & Blackwood, M. (2010). Computer Literacy and Non-IS majors. *Information Systems Education Journal, 8*(58), 3-12. Retrieved from http://isedj.org/8/58

YoYo Games Ltd. (2015). *GameMaker: Studio*. Retrieved from Yo Yo Games: http://www.yoyogames.com/studio

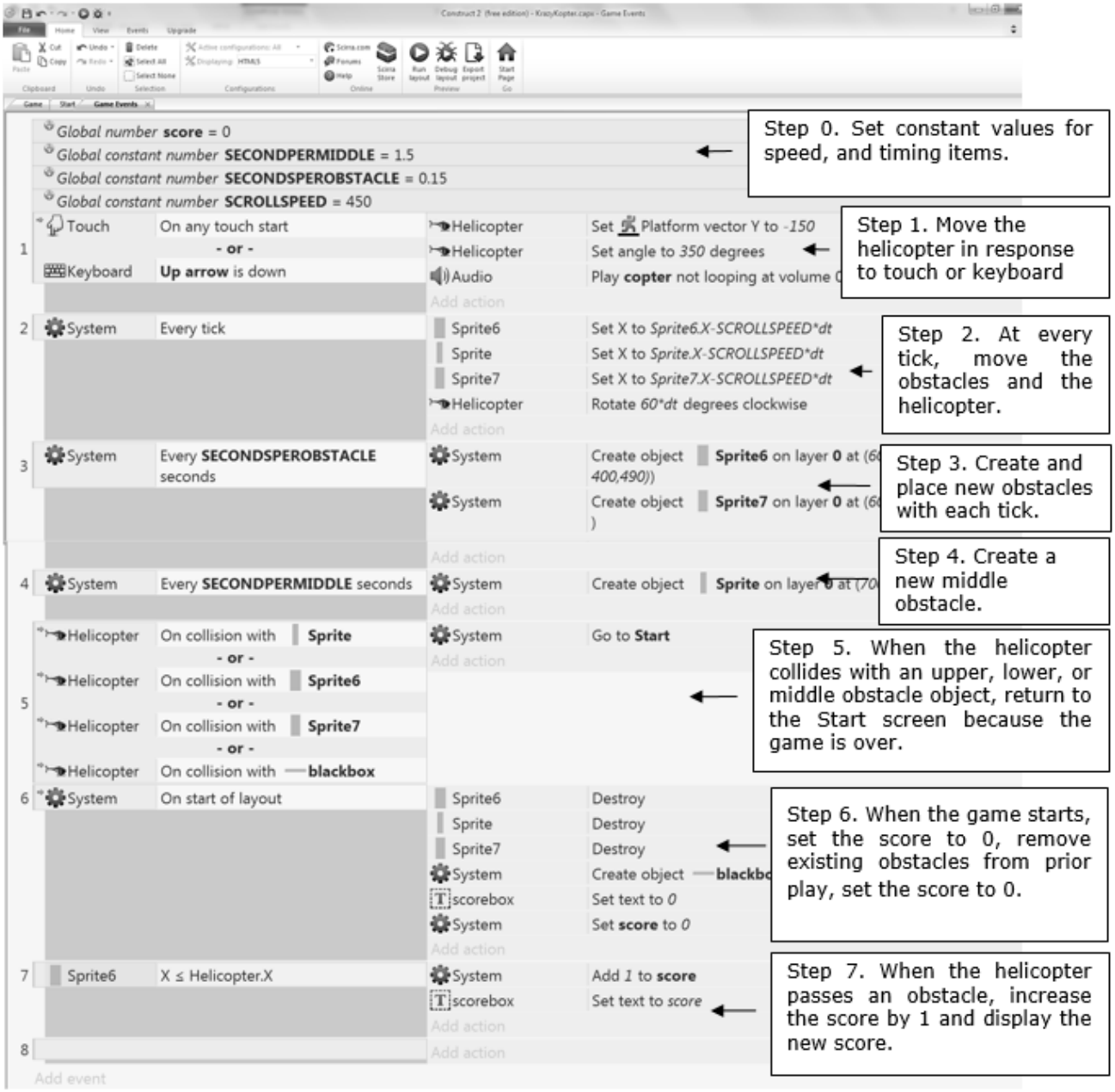
Editor's Note:

This paper was selected for inclusion in the journal as a EDSIGCon 2015 Meritorious Paper. The acceptance rate is typically 15% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a paper in 2015.

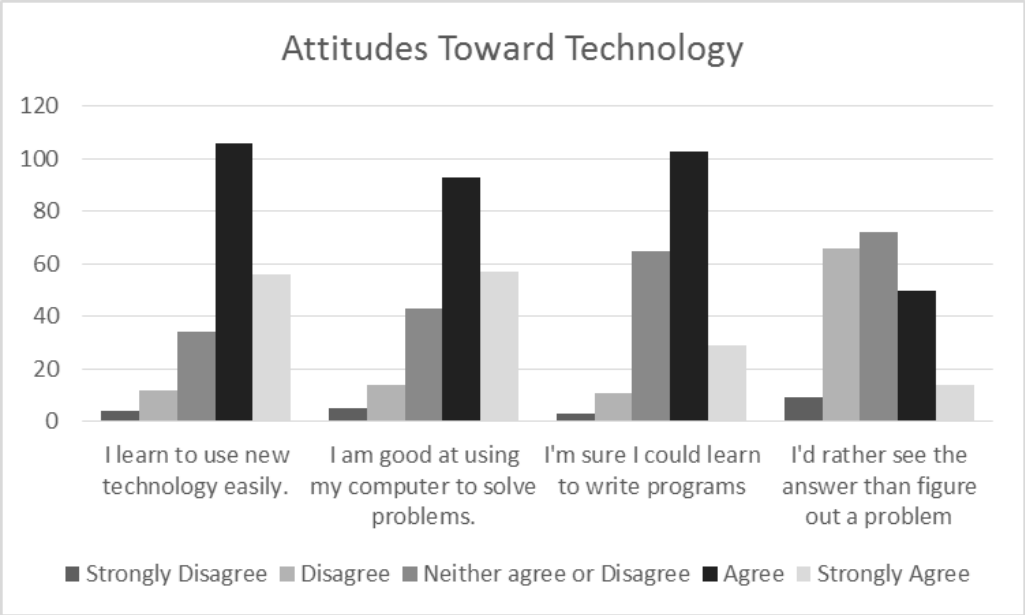
Appendix 1. Additional Figures



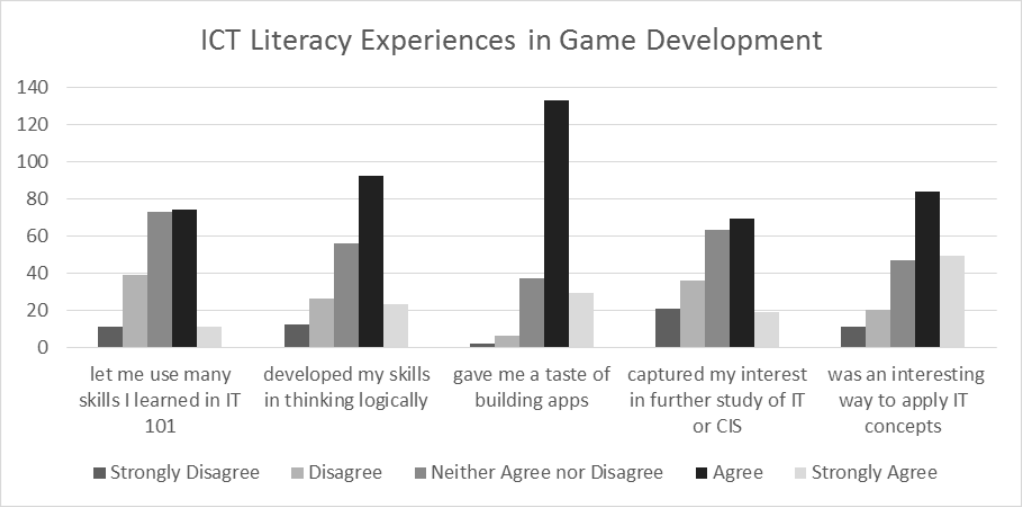
Appendix 1 Figure 1. (a) The start screen and game play screen when the game is playing on the web or a mobile device. (b) Designing the game play screen in Construct 2.



Appendix 1 Figure 2. Specifying rules of game play in Construct 2. Steps shown demonstrate programming concepts listed in Table 1.



Appendix 1 Figure 3. Attitudes toward technology among participants.



Appendix 1 Figure 4. ICT Literacy Experiences in Game Development.

Appendix 2. Gaming Resources

Table 1. Online Gaming Resources

Title	Web Address	Description
Construct 2	https://www.scirra.com/construct2	Creation engine for 2-D games that generates web and native applications.
Construct 2 Tutorials	https://www.scirra.com/tutorials/top	Tutorials for learning Construct 2.
Best Sound Effects Ever	http://www.bfxr.net/	Web app for creating and previewing sound effects for computer games. Sounds can be exported as WAV files.
Open Game Art	http://opengameart.org/	Open source collection of graphics for use as backgrounds, characters, shapes, and objects in computer games. Online forum for game developers to discuss and share their games and artwork.
Pixel Prospector	http://www.pixelprospector.com	Links to websites providing graphics for use in games without royalties, tools for creating sound and music, and additional resources for game developers.
Video Game Name Generator	http://videogamena.me/	Generates names for video games by randomly selecting unlikely word combinations.

For complete assignments, descriptions, and handouts for the gaming assignment, visit http://cis.bentley.edu/sandbox/ and search for Gaming Resources.

Appendix 3. Gaming Assignment

Working individually or with a partner, create your own 2-D Platform Game using Construct 2 sized to play on a mobile device. You might model it after the game we created in class, or look at one of the tutorials on the Construct 2 web site and model it based on one of those tutorials, or create something entirely original.

Basic Features
Your game should include several of these basic features:

- An attractive home screen with a button to play the game and rules describing how to play
- A game screen, that when the game is over, returns you back to the home screen
- Keyboard and touch input
- Spawned objects
- Music or sound
- Attractive graphics

Advanced Features
Your game should include several of these, or other advanced features:

- Allow user to turn on or off background music
- Score keeping
- Collision Detection
- Randomization of speed, placement of objects, etc.

Publish your Game to your Website

- Create a page called game.html and link to it from your website's home (index.html) page.
- Take a screenshot or two of your game, write a brief description of how to play, and describe which basic and advanced features above you included (and how).
- Include a link to a "full screen" version of your game.
- Place a link from your game.html page to a page named play.html that displays your game in full screen, ready to play.

Publish your Game to the Windows Store
Follow the instructions provided for how to publish your game to the Windows Store. This is an involved process, which you will need to complete in on a computer running Windows 8.
Grading:
Grading is based on CIS Sandbox tutor reviews and the instructor's comments. We will look for originality, playability, complexity, and aesthetics.

- **Acceptable:** Your game is very similar to the tutorial we did in class, but you changed the graphics or sounds. Not so original, but you get the idea.
- **Better:** Your game has a different play model, but it's not very playable. For example, you always win or lose immediately, the game play is not intuitive, or it's just not fun to play.
- **Best:** Your game is original, uses several basic and advanced features, and is highly addictive.