Splat! er, Shmup? A Postmortem on a Capstone Production Experience

Adrienne Decker, Christopher A. Egert, Andew Phelps
School of Interactive Games and Media and RIT Center for Media, Arts, Games, Interaction and Creativity (MAGIC)
Rochester Institute of Technology
Rochester, NY – USA
adrienne.decker@rit.edu, caeics@rit.edu, andy@mail.rit.edu

***Abstract*— Developing large-scale software applications in teams has long been a popular feature of capstone courses in computing degree programs, particularly in the fields of computer science and software engineering. Instructors utilize real world problems as well as external clients to motivate the students and to provide authentic experiences. Within the field of game design and development, students have strong desires to make and ship a game or series of games during their collegiate experience. To address this need, we created a production studio course offering that addressed the desires of students to create a game as a culminating experience while focusing upon the production process and challenges that are particular to the game industry. In this paper we describe the process undertaken by students, faculty and staff to ship a game. We discuss the successes and failures of the team and in achieving their goals. We show the similarities and differences between the process of developing and shipping a game and other types of large-scale software design and development projects undertaken in similar courses**

Keywords— game development; game design; game development education; game development teams; software development project

I. Introduction

The School of Interactive Games and Media at the Rochester Institute of Technology recently began offering a semester long 'production studio' course. This curricular experience was designed to satisfy a number of goals.

First and foremost, the production studio course served to expose students to the principles of game production through a number of authentic contexts, whether those contexts came from the instructor, from clients outside academia, or from students wishing to explore entrepreneurial opportunities.

Second, the production studio course was designed to allow students to experience the production process that dealt with issues related not just to large-scale software development, but also to the attention to detail related to products within the games industry.

Furthermore, the production studio course acted as a culminating but unofficial 'capstone' to the degree program. While the production studio course was and continues to be elective, many students within the program and those outside the discipline have taken one or more production studio courses in their last years of study.

In this paper, we will discuss the results of one such offering of the production studio course in which the students worked on a game design and development project. The idea for the project was conceived by the instructor for the course and provided a significant challenge of balancing gameplay fun, educational experience, and novel interaction into one application. The students in the course had to realize the instructor's vision by fleshing out of the idea and implementation, staying true to the overall goal of the project while simultaneous contributing to the overall success of the game application.

II. Background

Within the computing disciplines, the capstone project course satisfies program outcomes related to real world experiences as well as outcomes related to large-scale team-based application development [1, 2, 3]. Guidelines for computing curricula in computer science, information technology, and software engineering address the need for such experiences to anchor theoretical knowledge to a practical domain [4, 5, 6]. The disciplines within the field of computing have recognized the need for a culminating experience in which students explore the development of a product through the lens of clients, users, and other stakeholders that have a vested interest in the overall product lifecycle of a complex software application [7, 8, 9, 10, 11].

There are a number of examples in the computing domain that speak to the design, implementation, and outcomes of individual course offerings [7, 8, 9, 11, 12, 13]. Each of these examples emphasize particular traits that make for an authentic project experience. For example, large-scale project courses often look to problems derived from real-world sources, such as projects that benefit academic community processes [14], projects that speak to educational practice [15, 16], projects that address challenges in the open-source community [17, 18, 19], and projects that assist in the delivery of product in the not-for-profit and commercial spaces [20, 21]. Goals and outcomes for these courses are carefully balanced between the the needs of these particular stakeholders that are involved in the project against the reality of the educational and practical considerations of software architecture. For new projects, careful consideration of needs analysis and software cost estimation is essential [22]. For open-source projects, care must be taken to determine the status and complexity of current

offerings as well as the openness of the community and other contributors [18, 19].

In comparison, there are examples of courses that place emphasis not on the application domain, but rather on the process itself. These courses place emphasis upon educational goals related to the software product lifecycle. Issues related to product lifecycle in terms of planning, development, and maintenance and the ability of teams to work effectively within the development process becomes a major focus of the experience. Such courses may include examination of agile models that help define the development process [9, 23]. They may also include checks and balances for code review, robustness checking, error detection, and user experience [13, 21].

Another important feature of these courses is the concept of student and instructor role. For the instructor, project courses often require several perspectives in terms of interaction with the student population [7, 24, 25]. At one level, the instructor is still acting as the moderator of the educational experience, ensuring that course outcomes are met and that the educational needs are being met for the students. At another level, the faculty becomes the mentor for the student, advising not only on software production but also issues related to the interaction with key constituents. The faculty acts as liaison between clients or takes on a production role within the course. For students, project courses also require that they experience different roles from design, development, architecture, usability design, documentation, feedback analysis, and other roles directly related to the product lifecycle [7, 21, 26, 27].

Within each course design, these areas of exploration are weighted with different levels of emphasis depending upon resource availability, constraints of the course experience, and overall program needs [21]. The experience can range from a simulation of a real-world authentic experience in which the students and the instructor drive project ideas and expected outcomes, to interactions with companies in which the students have to completely manage the situation.

However, despite the intended outcomes and benefit to students, there is still the issue as to how students engage with these courses. For some students, the reality will be that a complex project is a required part of the curriculum [28]. However, for those with a choice, motivations can include either the desire to address a challenging and complex problem head on or involvement with a specific application domain that has great appeal [29].

III. The Allure of Video Games

For many computing students and their faculty, the digital video game holds a special place in the imagination. From the student perspective, the video game represents a cultural symbol that they are already deeply familiar with, as many computing majors have played or encountered video games and are familiar with some of the demands related to their construction.

For the instructor, the video game represents a system of reasonable complexity with many discrete parts that must work together in the service of the game experience.

For both parties, video game systems possess large and often challenging code bases, require rapid skills acquisition for graphical libraries and other necessary toolsets, require significant thought to the overall architecture of the system, require a fusion of creative and technological concerns, and appeal to our sense of making software that serves a purpose (in this case entertainment and user engagement) [29, 30, 31, 32].

Because of the broad-based appeal of these experiences, a number of computer science and software engineering programs have used video game design and development to appeal not only their own students, but as means of broadening participation with diverse and underrepresented groups [31, 33, 34].

In practice, the use of video games in capstone project courses has been addressed in the literature with mixed results for both students and faculty experiences [33, 34, 35], but despite any particular outcome, their appeal cannot be denied.

IV. The Challenge of Video Games

On the surface, video game applications have the same design and development considerations as any other piece of software. Developers must address issues such as overall system architecture, communication of information between game engine components, maintenance of system states, and processing of input and events [36]. While at the same time, the video game must present the user (or in this case player) with an understandable way of interacting with the system as well as an entertaining experience [37].

However, despite the similarities in some of the processes, video game design and development changes the emphasis related to certain production practices. Such changes require that instructors rethink the nature of development groups and their interactions within these courses.

A. Issues of large scale software development

Like many large-scale software development experiences, video game applications consist of a number of subsystems. At the highest level, these systems control the logic and graphics rendering of the game. But, as the designer and developer explores the code base, these systems must address everything from scene rendering (lighting, object rendering, materials and texture application, shadow, and environmental effects), input processing, state management, collision detection, physics, artificial intelligence, networking, user interface design and implantation, and much more. All of these systems must work in a near real time environment to ensure visual consistency and smooth gameplay [36].

The architectural decisions for such systems are fairly complex. Developers must decide what they are willing to create themselves and what they are willing to integrate into their architecture from other sources. Some choices are straightforward (such as the selection of graphical API that will be used in a game engine) while other choices must be carefully studied for compatibility, ease of integration, cost, licensing restrictions, and extensibility [36, 37]. Poor architectural decisions in games are not easily reversible and

often have far reaching ramifications in the success or failure of a project [38].

B. Role diversity on project teams

All endeavors in large-scale software development require team members to serve specific functions in the software construction process. These roles include software architect, framework designer, implementation specialist for different areas of system construction, user interface designer, quality assurance inspector, documenter, and many other roles based upon the need of the software system. However, despite the range of roles on a larger project team, the responsibilities are often filled with students who come from similar backgrounds with the same perspectives and life experiences [39].

For video game design and development, the number of roles increase and span an incredibly diverse set of backgrounds. Successful game projects require a team that has a combination of technical, artistic, and user experience competencies along with domain experts for the genre of game that is being developed [40].

Although one could argue that the basic challenge between standard software production and game development is similar in nature, the range and depth of expertise needed, including personnel in management roles with reasonable experience, can have profound effects on what is created. For example, there can be a large gulf in player experience depending on whether a game utilizes programmer art, which is functional but often lacks polish, or whether a skilled content developer has contributed to the work.

Due to the range of talents and expertise required for many game design and development roles, students are asked to either take on these roles with little experience, acquire expertise by either purchasing asset libraries for content, third-party libraries for technology, or by appealing to students in other disciplines to join their teams.

C. Team Size

Teams for game development are often large in nature. This is not only to accommodate for the number of active roles within the development team, but also to provide redundancy and overlap for critical development functions.

There have been several studies on the size of teams for software development [41, 42], and as with any software project the concerns of team size is a balance between social behaviors as team size grows, productivity, communication, and other intrinsic factors that speak to a properly functioning group.

Since game design and development roles span such diverse disciplines, team size for larger projects can allow for the formulation of cliques, usually along the lines of content developers, level designers, and engine developers.

D. Role Assignment

A typical project course addresses roles either through some form of an assignment mechanism. Students are either assigned roles by the instructor [25], self-select based upon personal experiences and desired outcomes for the experience [46], rotate roles to ensure a broad perspective [43], or ostensibly interview for the role and are selected based upon how they present themselves to the instructor or client [27, 43]. Evaluation in a given role is often determined through peer feedback, faculty observation, or through other evaluative processes. If an individual fails in his or her role, the consequence is tied to an educational outcome – roles are reassigned to cover deficiencies and faculty handle instances where students withdraw the course.

However, in the game industry, role failure of an individual or group is handled quite differently than the classroom, resulting is reassignment or dismissal if the expected output is not in alignment with the project direction. This results in the acquisition of new talent or the cancellation of projects. In either circumstance, the overall health of the group, product, and company is impacted by such actions.

E. The user experience

User interfaces for software applications are designed to make the role of interacting with software easy and predictable [44]. The user experience for software should provide the user with an awareness of what can be done and how it can be accomplished to maximize productivity with minimal action.

Although user interface is the controlling factor for experiences with game systems, the goals of the user interface are different. As video game systems are designed as entertainment, the user experience should still be easy to master and must allow the player to quickly ascertain the status of his or her state within the game and should allow for rapid access to player action.

However, the entertainment factor for video games also dictates that the overriding experience for any interaction with game software should be fun [45]. The technology, art, level design, story, audio, and other components all contribute to creating a state of fun in the game. These individual components are noticed by the player when the sense of fun is absent and become ubiquitous when the game is truly enjoyable.

Unlike standard forms of usability testing [44] that can help us measure and analyze the user experience, measuring fun can be elusive.

F. Product viability and client acceptance

In project courses, the role of the product consumer or client is often taken up by faculty or on-campus groups [46, 47, 48]. When working with open source and not-for-profits, outcomes can be greatly curtailed, and lack of delivery of a product can be justified against the academic experience presented to the students [14, 49, 50].

However, in the games industry, there is always the possibility that a product fails to ship. As part of the product cycle, dealing with the possibility that project is terminated because it does live up to the quality of a company's product is a critical experience needed by game developers.

G. Communication

Many capstone course experiences instill in their students the need to communicate effectively through reports and presentations [20, 51, 52].

It is also important to engage stakeholders and other personnel when developing game systems. This includes clientele such as game testers, the web community, review publications, media personalities, event and show coordinators, and contest organizers. Students also need to be prepared to discuss their work with visitors who often show up with little notice. The skill is to balance spontaneity in discussion with an awareness of how to "stay on message" regarding the project.

V. Production Studio Course

At Rochester Institute of Technology, the production studio course is an elective in the Game Design and Development degree that students typically take in their fourth or fifth year. Our undergraduate program is a co-op based program and students generally miss a semester or two on campus in their sophomore/junior year, which means that some students take five years to finish the degree.

As the production studio course is meant to occur after students return from a co-op experience, one of the intended goals is to leverage their existing knowledge of the workplace with the skills acquired in the workplace. In this manner, production studio can assume that students enrolled in the course are already familiar with some production models that are used in industry and have experienced situations at intersection of personal responsibility and team success.

The goals of the course are to produce a ship-ready game that was developed by the students using their own techniques and project management style. The focus is heavily rooted in the basis of the entire curriculum of the School, namely that it is project-driven and experiential. If we wanted to 'teach production' we felt strongly that it was necessary for students to experience it first-hand from a variety of perspectives. In other courses, smaller projects and prototypes help to build out students' portfolios. But, as a department, we suffer from a typical problem in that many of our computing-oriented games students express a desire to produce quality finished and published work by the time they graduate. Unfortunately, many do not, because class projects, the rigors of campus activities, and the general climate tend to work against the sustained focus needed to produce top-tier commercial-quality work.

The purpose of the production studio course is to

(a) produce a quality, ship-ready indie game title based on either student or instructor designs (and usually a combination of both)

(b) design and implement an appropriate project management strategy that successfully results in the production of the work in question,

(c) leverage multi-disciplinary skills and knowledge sets in the creation of a multi-faceted digital media project, and

(d) successfully document each stage and contribution to the work in question, as a learning tool for participants and outside communities.

B. Fall 2014 Production Studio

The motivation for the project for the fall 2014 section of Production Studio were discoveries about Jackson Pollock's work. There has been some evidence to suggest that the famous 'drip paintings' of Jackson Pollock were not entirely random and that Pollock had a vision of what he wanted the finished product to look like [53]. The combination of a love of Pollock's work, a familiarity with the artists' statements regarding the role of being 'in the zone' when creating such work, and in drawing from similarities between flow theory [54] as it was originally based on the study of artists and more recently applied to game mechanics led the lead course instructor towards this topic for the production studio. This could be fertile ground to create a game that allowed users to construct a work similar enough to Pollock's master-work as to allow reflection and examination of process.

Specifically, the goal was to marry a form and function that gamers were familiar with (a classic twin-stick 'arcade shmup' ('shmup' being early 80's arcade game slang for 'shoot-em-up')) and the basics of a path-driven, drip-oriented design. In this way, the instructor hoped to bridge a world in which the general public was more familiar (i.e. shmups) with the more esoteric elements of 1950s Modern Art. Early tech-tests that were created during the spring and summer leading up to the course seemed to indicate a graphical effect and fidelity that could lend itself to such a concept.

With this general ideal, the supervising faculty instructor hired an industry professional that had experience in software production management and associated processes, namely Scrum, to serve as a second instructor for the course on topics and areas related his expertise. Also, the Assistant Director of the sponsoring research center, who is responsible for public relations, social media, outreach, press, and communications for the center was asked to work with course participants on such issues as they pertained to the development process and release of the game. Additional faculty were identified as expert resources for technical effects pertaining to 'looking like paint' as well as in the game engine that was planned to support game development (Unity).

With these details and preparations, the authors constructed a syllabus that set forth these goals and objectives, and planned a course outline that followed a basic arc of:

(1) Identifying roles and responsibilities for the project

(2) Establishing a series of milestones (in roughly weeks 3, 6, 9, and 12 prior to the final presentation in week 16) and critiques for the work as it progressed

(3) Established a workflow model based on the Scrum methodology that would result in the completion of the project

(4) Created a series of play-tests and critiques during the middle and end of the semester

(5) Provided for peer feedback and reflection at the end of the process. In this manner, the instructors hoped to both guide the class through the construction of the game, as well as reflect on the process and the final product.

C. Course Interactions

Along with the progression described in the previous section, there were a number of ways in which the faculty, consultants, and students worked together. Each class day provided students with a central focus for project completion, centered around themes of design, implementation, integration, workflow management, critique, and presentation.

On days dedicated to design choices, the faculty and consulting leads would provide their video to the students, focusing upon the "big picture" view of the experience as a means to describe smaller design choices. Simply speaking, the smaller design goals had to support the overall vision of the project.

On coding days, the faculty and consultants would provide student leads with appropriate tasks. The student leads would decide how to implement a particular direction. Leads would communicate with each other to make sure that design and development concerns would integrate properly. Upon execution of required tasks, team leads would report back to the faculty and consultants regarding their accomplishments.

For days focused on the critique process, faculty and other game development personnel would provide feedback. As the critique process was predominately focused on the overall experience, comments would deal with issues of aesthetic, game mechanic implementation, game balance, and playability.

Another important class function involved days related to communication. For these class days, our communication personnel would address subjects such as building and developing online community as well as how to talk to media, distributors, and the public in general. Students were also exposed to professionals in the field and were responsible for communicating information about their game.

Finally, some days were focused upon the playtest process. Students learned the process of observing and recording information around the player experience. Students were also responsible for analyzing playtest data and developing schemes to address critical challenges with the game application.

VI. Observations

With this preparation, we launched the production studio to create what became *Splattershmup: A Game of Art & Motion*. Right away we were faced with a number of obstacles and issues that were not expected. We assumed, based on the fact that we were in essence 'giving games students what they said they wanted' that they would be both excited, motivated, and dedicated to the cause. For various reasons and issues we discovered that this was not always the case.

A. Inexperience

Several students simply weren't ready for a production level experience. They were still lacking in either skills, theoretical background, or both. A contributing factor to this is that for students to be classified as 4th-5th year, a pre-requisite for the course, the student must have completed a certain number of credit hours. Thus, a student with significant transfer- or AP-credit can be considered 4th year much before their actual coursework and practice would consider them as such.

For those students who were really in their 4th or 5th year, while in earlier courses, they had used the tools or languages incorporated in our production course, they had not always done so in an open-ended, applied way. Students would proclaim knowledge of a given package, technique, or application, but then fail to apply it to the problem at hand in a meaningful way. They had, in essence, become excellent at solving toy problems, but not good at applying their knowledge at a larger scale.

B. Team Size

Students had not, in some instances, worked with a project team larger than 3-4 people prior to this course, where they were asked to work on a team of 15 students. As such, failings of communication, scale, and roles became apparent as the semester progressed with some students trying to break off into smaller units to complete their work and ignoring the task of integrating with the larger group.

C. Motivation and Ego

Several students blatantly exhibited a desire to consider the course not as a special opportunity to create production level content, but rather as simply 'one of their several courses' overall, and 'one requirement to graduation'. Such failure to understand the role and nature of the opportunity confounded the course instructors on several occasions. We feel that we have identified two root causes: ego and a decidedly student view of "good enough."

Students failed to appreciate a role in creating content in which they were not the sole party with creative control. As the class progressed, a set of approximately 3 students failed to engage to an increasing degree as their ideas for the game were jettisoned based on player feedback, and/or timing constraints. Student feedback after the course indicated that some students felt that unless there was a distinctive feature of the game that was of their design that somehow their role/responsibility would not be recognized. This was in direct contrast to statements made by the instructors both verbally and in writing, which focused on getting the core game mechanic and 'painting experience' working before anything else and that contributions would be recognized as pertained to the project, not to a particular feature or snippet of code.

At the same time, we also discovered that students in the course were actively comparing their work only to other student work. So much so that they were forcibly excluding from comparison professional work even if it was produced by a similar team-size and a similar timeline to their own. One student repeatedly stated that what they were building was "good enough for student competitions". However, what the instructors set out to encourage and cultivate was to make an experience that was a completed game that was good enough to

ship to the general public.

D. Playtesting and Critique

The process of playtesting and outside critique wound up being flawed despite the best intentions. In seeking expert feedback, the instructors recruited several industry experts to review the game (and associated concept) at several points during the production process. The first of these, Michael Gough, VP of User Experience at Adobe, was intrigued by an early prototype, and the 'drawing by moving mechanic'. The second reviewer, Anna Sweet was at the time of her visit the senior business development manager for the STEAM service offered by Valve, Inc. Ms. Sweet was a visitor to the lab in week 6, which was approximately the mid-term for the course (the course instructors expected a lull after her visit, as well as a crunch leading up to it, and as such focused mid-term priorities around this visit).

Needless to say this visit was of significant interest to the student body. Her feedback, which was largely positive had an interesting impact on the students. While the instructors took her remarks to mean that the game concept and associated development for the moment was commendable, the students took her feedback to mean that the prototype was, as presented, worthy of praise. This then presented a dichotomy that never ultimately resolved itself during the lifespan of the course: students felt that the project was near-complete, whereas the instructors felt that the project was at best a rough sketch. The associated levels of effort to complete these mismatched goals was thus similarly out of alignment, which led the supervising faculty to present Figure 1 in a post-mortem presentation of the project.

The resulting sense of expectation and performance is clear, in that in the wake of feedback that validated students felt they received. In the act of recruiting expert feedback, we biased the resulting discourse in such fashion that the course was unable to recover properly. This is in contrast to the fact there were other playtests and experiments conducted in weeks 9-16 that were of equally critical import (including the university President and Vice President for Research), but none captured the student interest in a manner similar to outside games-industry professionals. While this is a well-known trope within the faculty of games-related programs, it bears repeating: the weight and clout of the industry seems to have almost unmitigated impact, but it is rarely as unforgiving in its commercial realities to academia as it is in general practice.

E. Course Overload

One of the concerns regarding this course experience was that of student overload. The faculty involved in designing this course experience readily and freely admit that the goal was to create an experience that was an analog of an industry production scenario, with similar objectives and pacing. Before and on the first day of the course, the faculty explained to the students that this process was radically different than other courses, and clear examples were given as to the student role and expectation for the project. Similar to industry, contracts were signed not just to protect student rights, but to outline the gravity and responsibility of the commitment involved with this particular course.

Even with such warning, students felt they had a sense of what they were getting into and professed a number of regulating behaviors and activities that would help in the management of a higher than expected workload. During the execution of the class, it was readily apparent that most student strategies for coping with and balancing overload had not been field tested.

In many cases, students relied on the same strategies and behaviors they engaged in the normal classroom. Although these strategies proved effective for academic scenarios, the strategies commonly failed with the higher level of team interdependency and the demands of the course. The faculty observed that students were reluctant to change strategies to those that prove useful in commercial settings, and in cases where students withdrew into the academic mode of overload management, the team experienced communication and execution problems.

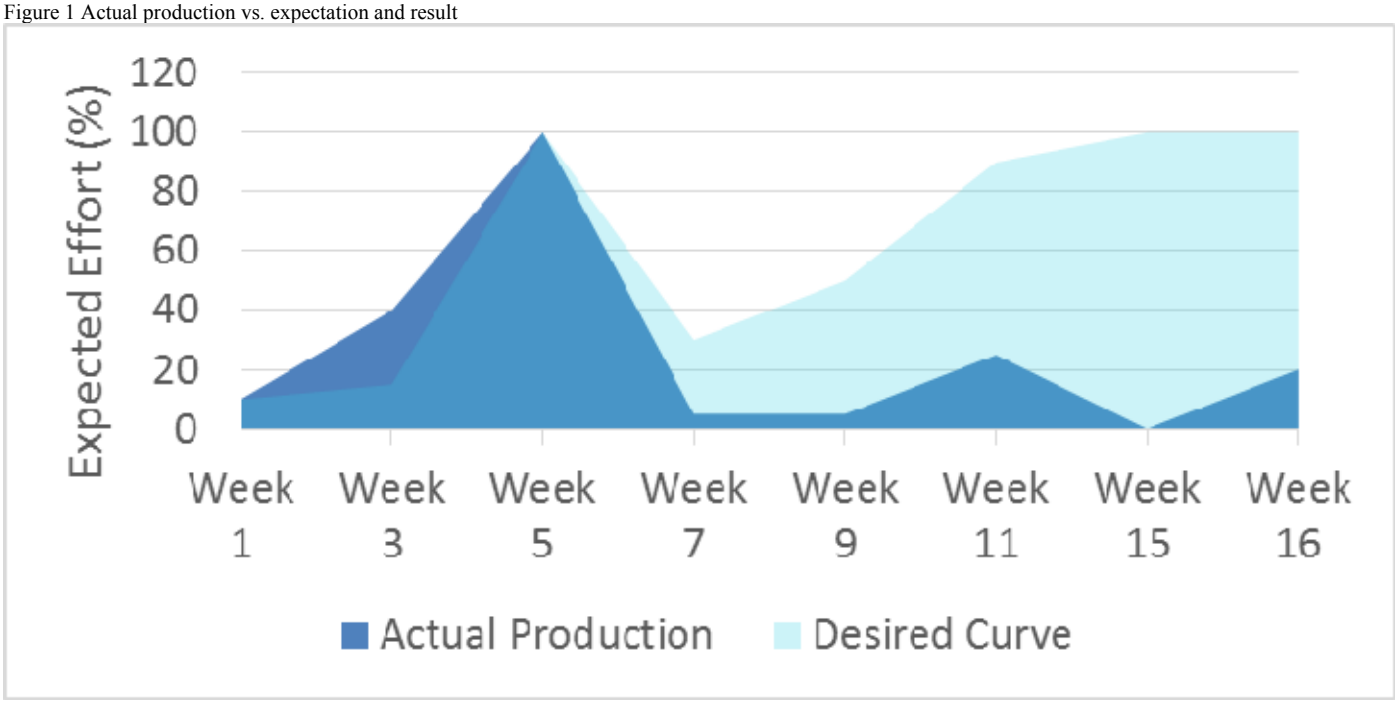
VII. Student response and Aftermath

At the end of the course, the faculty collected data from institute mandated course surveys. For the fifteen students that completed the course, six provided feedback. While there was not enough data to provide a quantitative analysis for the course, the written comments demonstrated the dichotomy how students perceived the experience.

On a positive note, there were at least two students that recognized the value of having an experience that was modeled after actual industry processes. Unfortunately, there were also comments that demonstrated that students were uncertain as to the relationship between the faculty, consultants, and their peers. Some responders wanted explicit direction while others felt they wanted more autonomy and control of the process. For the latter, it was clear they were uncomfortable following the leadership of the vision team and would have preferred to work on projects of their own design (which would be very unlikely fresh out of college).

Beyond the course, students involved in the project were part of a national campaign to promote the project. Students were involved with its presentation at local venues such as the ImagineRIT annual innovation festival as well as national venues such as the Game Developer's Conference. Currently, the game has been featured in several prominent social media outlets and has its own distribution channel.

Figure 1 Actual production vs. expectation and result



VIII. What We Learned

In addition to the several insights noted in the preceding section, there are a few other large-scale observations that are worth noting with respect to what we learned through this process. The first is that students in our programs need to stop separating what they perceive as "their work" and "student work" from "real work" or "professional work". This came up repeatedly during the course and was used by students as a way to duck standards or responsibilities, namely that by comparing their work to other student work they sought to avoid direct critique and comparison with best practices and production standards as they are defined in the commercial field.

It was also apparent that students had divorced, in both their practice and the way that they conceived the work for the course, game design from engineering and construction. This is an emerging problem in that our curriculum focuses heavily on game design and core theory in the sophomore year, but these courses often use paper-prototyping techniques, board and card games, and other tools to allow students to focus on design concepts and not get bogged down in technology. But this produced an interesting outcome in that several students in the production studio course saw themselves as 'designers' or wanted to 'focus on game design'. This is fine, in principle as in industry, there are roles for designers. However, it is still the case that there needs to be a corresponding role for implementing the features of that design, reviewing and testing gameplay in progress, and working directly to fix issues as they arose. This kind of 'design divorced from implementation' mindset is something that will need to be examined carefully and corrected if the curriculum is to continue to be as relevant and practical as desired.

It was also clear that the students in the course failed to ultimately make the transition in terms of thinking less about building the game 'for' the faculty and staff supervising the courses, as opposed to 'with' the faculty and staff supervising the course. The project, given both the way it was conceived and the desire to ship publicly/commercially as a portfolio driven activity, held relevance and investment not just from the students but from everyone involved and across several constituencies at the university. Some students in the course recognized this and engaged with it, while others clearly did not.

IX. Conclusion

The process of making large scale software (games or otherwise) in courses is difficult. Even when preparations are made and the course is structured around the shared goal of completing and shipping the end product, it does not guarantee buy-in from the students or a term that is free of trials and tribulations. Perhaps some of the more surprising aspects of the failures are those of engagement. It was surmised that these students especially, would want to make and ship a commercial-quality product. Some clearly did not embrace that goal. Further, upon hearing one good critique of the product, much of the momentum was lost for the term and the game was never fully realized and never lived up to the potential that the reviewer from industry saw. Further, students even at this late stage in the curriculum have yet to see themselves as budding professionals and still see themselves as students. As such, they believe that the quality of their work didn't need to meet industry standards, even if releasing to the industry was the goal.

These challenges inform our next efforts at this type of production studio and inspire us to structure the course and

content differently to better motivate and engage the students in the process so that we achieve the success of producing a shippable title.

ACKNOWLEDGMENT

The authors would like to extend our thanks to the students of the Interactive Games and Media Production Studio course offered in the Fall Semester of 2014. The authors would also like to thank Aaron Cloutier for his expertise as design lead and project coordinator. His contributions were instrumental to the success of the course.

REFERENCES

[1] C. L.Bullard,I. Caldwell, J. Harrell, C. Hinkle, A. J. Offutt, "Anatomy of a software engineering project," in *Proceedings of the nineteenth SIGCSE technical symposium on Computer science education (SIGCSE '88)*, Atlanta, GA, pp. 129-133, 1988.

[2] J.G. Meinke, "Augmenting a software engineering projects course with oral and written communication," in *Proceedings of the eighteenth SIGCSE technical symposium on Computer science education (SIGCSE '87)*, St. Louis, MO, pp. 238-243, 1987.

[3] L.M. Northrop, "Success with the project-intensive model for an undergraduate software engineering course," in *Proceedings of the twentieth SIGCSE technical symposium on Computer science education (SIGCSE '89)*, Louisville, KY, pp. 151-155, 1989.

[4] The Joint Task Force on Computing Curricula, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, New York: ACM, 2004.

[5] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) & IEEE Computer Society, Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, New York: ACM., 2013.

[6] B. Lunt, J. Ekstrom, S. Gorka, G. Hislop, R. Kamali, E. Lawson. R. LeBlanc, J. Miller, H. Reichgelt, Curriculum Guidelines for Undergraduate Degree Programs in Information Technology, New York: ACM, 2008.

[7] D. Coppit, J.M. Haddox-Schatz, "Large team projects in software engineering courses," in *Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE '05)*, St. Louis, MO, pp. 137-141, 2005.

[8] J.G. Flowers, "Improving the capstone project experience: a case study in software engineering," *in Proceedings of the 46th Annual Southeast Regional Conference (ACM-SE 46)*, Auburn, AL, pp. 237-242, 2008.

[9] S.M. Hadfield, N.A. Jensen, "Crafting a software engineering capstone project course," *Journal of Computing Sciences in Colleges*, vol. 23, no. 1, pp. 190-197, October 2007.

[10] J.A. Preston, "Utilizing authentic, real-world projects in information technology education," *ACM SIGITE Newsletter*, vol. 2, no. 1, art. 4, April 2005.

[11] J. Vanhanen, T.O.A. Lehtinen, C. Lassenius, "Teaching real-world software engineering through a capstone project course with industrial customers," in *Proceedings of the First International Workshop on Software Engineering Education based on Real-World Experiences (EduRex '12)*, Zurich, Switzerland, pp. 29-32, 2012.

[12] D.E. Krutz, S.A. Malachowsky, T. Reichlmayr, "Using a real world project in a software testing course," in *Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14)*, Atlanta, GA, pp. 49-54, 2014.

[13] C. Szabo, "Student projects are not throwaways: teaching practical software maintenance in a software engineering course," *in Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14)*, Atlanta, GA, pp. 55-60, 2014.

[14] Z. Alzamil, "Towards an effective software engineering course project," in *Procedings of the 27th international conference on Software engineering (ICSE '05)*, St. Louis, MO, pp. 631-632, 2005.

[15] S. Ludi, "The benefits and challenges of using educational game projects in an undergraduate software engineering course," in *Proceedings of the 1st International Workshop on Games and Software Engineering (GAS '11)*, Waikiki, Honolulu, HI, pp. 13-16, 2011.

[16] M.M. Rohde, M.A. Toschlog, "Toward the fusion of serious simulation & video games," in *Proceedings of the 2009 Spring Simulation Multiconference (SpringSim '09)*, Society for Comp Simulation, art. 71, San Diego, CA, 2009.

[17] B.R. Krogstie, "Power through brokering: open source community participation in software engineering student projects," in *Proceedings of the 30th international conference on Software engineering (ICSE '08)*, Leipzig, Germany, pp. 791-800, 2008.

[18] R. Marmorstein, "Open source contribution as an effective software engineering class project," in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (ITiCSE '11)*, Darmstadt, Germany, pp. 268-272, 2011.

[19] T.M. Smith, R. McCartney, S.S. Gokhale, L.C. Kaczmarczyk, "Selecting open source software projects to teach software engineering," in *Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14)*, Atlanta, GA, pp. 397-402, 2014.

[20] G. Alkadi, T. Beaubouef, R. Schroeder, "The sometimes harsh reality of real world computer science projects," *ACM Inroads*, vol. 1, no. 4, pp. 59-62, December 2010.

[21] C.K. Hobbs, H.H. Tsang, "Industry in the classroom: Equipping students with real-world experience - A reflection on the effects of industry partnered projects on computing education," in *Proceedings of the Western Canadian Conference on Computing Education (WCCCE '14)*, art. 7, Richmond, BC, Canada, pp. 1-5, 2014

[22] E.R. Carroll, "Estimating software based on use case points," in *OOPSLA '05 Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, San Diego, CA, pp. 257-265, 2005.

[23] S. Krusche, L. Alperowitz, B. Bruegge, M. O. Wagner, "Rugby: an agile process model based on continuous delivery," in *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (RCoSE 2014)*, Hyderabad, India, pp. 42-50, 2014.

[24] S. Kurkovsky, "Four roles of instructor in software engineering projects," in *Proceedings of the 13th annual conference on Innovation and technology in computer science education (ITiCSE '08)*, Madrid, Spain, p. 354, 2008.

[25] B.K. MacKellar, "A software engineering course with a large-scale project & diverse roles for students," *Journal of Computing Sciences in Colleges*, vol. 26, no. 6, pp. 93-100, June 2011.

[26] T. Rishel, "An innovative project structure for teaching software engineering," *Journal of Computing Sciences in Colleges*, vol. 28, no. 2, pp. 232-237. December 2012.

[27] M.V. Stein, "Using large vs. small group projects in capstone and software engineering courses," *Journal of Computing Sciences in Colleges*, vol. 17, no. 4, pp. 1-6. March 2002.

[28] M. Buckley, H. Kershner, K. Schindler, C. Alphonce, J. Braswell, "Benefits of using socially-relevant projects in computer science and engineering education," in *Proceedings of the 35th SIGCSE technical symposium on Computer science education (SIGCSE '04)*, Norfolk, VA, pp. 482-486, 2004.

[29] K. Shaw, J. Dermoudy, "Engendering an empathy for software engineering," in *Procedings of the 7th Australasian conference on Computing education (ACE '05)*, vol. 42, Darlinghurst, Australia, pp. 135-144, 2005.

[30] J. Peckham, D. Rosen, "Multidisciplinary teams to fulfill apprenticeship requirements in industry," *Journal of Computing Sciences in Colleges*, vol. 18, no. 5, pp. 123-127, May, 2003.

[31] E. Sweedyk, "Women build games, seriously," in *Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11)*, Dallas, TX, pp. 171-176, 2011

[32] E. Sweedyk, R.M. Keller, "Fun and games: a new software engineering course," in *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '05)*, Caparica, Portugal, pp. 138-142, 2005.

[33] K. Claypool, M. Claypool, "Teaching software engineering through game design," in *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '05)*, Caparica, Portugal, pp.123-127, 2005.

[34] T. Smith, K.M.L. Cooper, C.S. Longstreet. "Software engineering senior design course: experiences with agile game development in a capstone project," *in Proceedings of the 1st International Workshop on Games and Software Engineering (GAS '11)*, Waikiki, Honolulu, HI, pp. 9-12, 2011.

[35] A.I. Wang, "Extensive Evaluation of Using a Game Project in a Software Architecture Course," *Transactions on Computing Education*, vol. 11, no. 1, art. 5, February 2011.

[36] J. Gregory, Game Engine Architecture, 2nd ed. CRC Press: Boca Raton, FL, 2015.

[37] J. Schell, The Art of Game Design: A Book of Lenses, 2nd ed. CRC Press: Boca Raton, FL, 2015.

[38] A. Grossman, Postmortems from Game Developer: Insights from the Developers of Unreal Tournament, Black and White, Age of Empires, and Other Top-Selling Games. Focal Press: Burlington, MA, 2013.

[39] M.V. Stein, "Using large vs. small group projects in capstone and software engineering courses," *Journal of Computing Sciences in Colleges*, vol. 17, no. 4, pp. 1-6, March 2002.

[40] M. Mencher, Get in the Game!: Careers in the Game Instustry. New Riders: Boston, MA, 2003.

[41] D. Rodriguez, M. A. Sicilia, E. Garcia, R. Harrison, "Emperical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562-570, March, 2012.

[42] H. Barki, S. Rivard, J. Talbot, "Toward an assessment of software development risk," *Journal of Management Information Systems*, vol. 10, no. 2, 1993.

[43] N. Clark, "Peer testing in Software Engineering Projects," in *Proceedings of the Sixth Australasian Conference on Computing Education*, vol. 30, Darlinghurst, Australia, pp. 41-48, 2004.

[44] B. Shneiderman, C. Plaisant, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 5th ed., Addison-Wesley: USA, 2009.

[45] R. Koster, A Theory of Fun for Game Design, Paraglyph Press, Inc: Scottsdale, AZ, 2005.

[46] B. Turhan, A. Bener, "A template for real world team projects for highly populated software engineering classes," in *Proceedings of the 29th international conference on Software Engineering (ICSE '07)*, IEEE Comp Society, Washington, DC, pp. 748-753, 2007.

[47] J.D. Tvedt, R. Tesoriero, K.A. Gary, "The software factory: combining undergraduate computer science and software engineering," in *Proceedings of the 23rd International Conference on Software Engineering (ICSE '01)*, pp. 633-642, 2001

[48] T.P. Way, "A company-based framework for a software engineering course," in *Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE '05)*, St. Louis, MO, pp. 132-136, 2005.

[49] E. Brannock, R. Lutz, N. Napier, "Integrating authentic learning into a software development course: an experience report," in *Proceedings of the 14th annual ACM SIGITE conference on Information technology education (SIGITE '13)*, Orlando, FL, pp. 195-200, 2013.

[50] A. Pletch, A. Agajanian, "A software engineering project that looks like the real world," *Journal of Computing Sciences in Colleges*, vol. 22, no. 6, pp. 92-99, 2007.

[51] V.W. Pine, M.L. Barrett, "What kinds of communication are required on the job?" *Journal of Computing Sciences in Colleges*, vol. 21, no. 2, pp. 313-321, 2005.

[52] J. Whitehead, "Collaboration in Software Engineering: A Roadmap," in *Future of Software Engineering (FOSE '07)*, Washington, DC, pp. 214-225, 2007.

[53] MoMA. (2015). *MoMA's Jackson Pollock mystery* [Online]. Available: http://www.phaidon.com/agenda/art/articles/2013/may/29/momas-jackson-pollock-mystery/

[54] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience. Harper Perennial Modern Classics: New York, 2008.