Lessons from a Class on Handheld Augmented Reality Game Design

Evan Barba, Yan Xu, Blair MacIntyre
GVU Center, School of Interactive Computing
Georgia Institute of Technology
85 5th Street NW, Atlanta, GA 30306
+1 404-894-5224
{ebarba3, yux7, blair}@gatech.edu

Tony Tseng
Interactive Design and Game Development
Savannah College of Art and Design–Atlanta
1600 Peachtree St. NW, Atlanta, GA 30309
+1 404-253-3259
itseng@scad.edu

ABSTRACT
This paper describes the structure, format, and outcome of an experimental class in Handheld Augmented Reality Game Design conducted as an inter-institutional collaboration between the Georgia Institute of Technology and Savannah College of Art and Design. It analyzes several unique elements of the class, especially the need for rapid prototyping of multiple games when exploring game design using novel game technology, and highlights important issues regarding interdisciplinary collaboration and mobile augmented reality gaming. The paper also discusses several approaches taken by students in completing class assignments, which suggest viable design strategies for the creation of future handheld augmented reality experiences.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques–*evolutionary prototyping, user interafaces*; D.2.m [Miscellaneous]; K.3.1 [Computers and Education]: Computer Uses in Education.

General Terms
Design, Human Factors

Keywords
Rapid prototyping, game design, collaboration

1. INTRODUCTION
Augmented Reality (AR) design is a multi-disciplinary endeavor that aims to add rich multimedia content to environments and situations in the everyday world. As such, it requires knowledge of computer vision, graphics and animation, user interface design, and an innate sense of how to combine these in new and compelling ways. Moreover, because augmented reality is contingent upon real world events and objects, AR experiences are more dependent upon their context than other interactive computer experiences. While this fact makes it difficult to identify general design principles which are common across AR experiences, it also makes for a natural connection with the current trend in mobile technology. The growing popularity of mobile gaming and location-based applications on handheld devices offer the promise of bringing AR to the mainstream consumer. However, since it is only recently that AR designers have had mobile devices capable of handling the processor intensive operations these AR applications require, there is still much to learn about designing effective and compelling handheld AR interactions, experiences and games.

The lack of a proven set of game concepts, or well accepted interaction techniques, for AR required us to take a more hands-on approach to teaching students how to properly leverage the capabilities of mobile devices to create AR games. Therefore, we designed a project-oriented class to support students in becoming familiar with the unique concerns of handheld AR games. This also allowed us to explore aspects of game design practices in regard to handheld AR design and education. In particular, we were able to leverage the prototyping process familiar to game designers and the natural emphasis it places on "learning by doing," to gain some insight into how to effectively use this process to explore mobile AR interaction techniques, and apply it more broadly as an educational tool. Multi-disciplinary courses on game design are not new (e.g., [3]), nor are classes in handheld game programming (e.g. [1]), but teaching a class using a novel technology unfamiliar to most of the students and for which no "canonical examples" exist presented some interesting challenges.

The class was segmented into four prototyping cycles, intended to provide both students and instructors defined opportunities to evaluate the effectiveness of their approaches and make changes as necessary. We initially focused the class around the concept of creating "AR board games," traditional tabletop board games with AR content delivered on hand-held devices. The reliance on physical game pieces in board games, as well as the game-boards themselves, are good correlates to the use of markers and multi-marker boards (a sheet of multiple markers in a known and fixed arrangement) in AR; and the social nature of board games capture the kind of play experiences we hoped to achieve.

As a final goal for the course, we hoped to give students an authentic game design and implementation experience by coordinating two classes at the Georgia Institute of Technology (GT) and Savannah College of Art and Design (SCAD), and having the students in those classes work together. This gave students at both institutions the experience of working in multi-disciplinary teams, exposed them to the unique constraints of the various sub-specialties within game design, and introduced them all to AR and mobile device development from a comfortable starting point.

ICFDG 2009, April 26-30, 2009 Orlando FL, USA.

2. PARTICIPANTS AND METHODS

The class consisted of nineteen students in total. Thirteen of these were enrolled at GT and six in SCAD. Fourteen of the students were undergraduates in their third or fourth year and five were graduate students. Seven students were majors in the Computer Science program at GT, five students were majors in the Computational Media program at GT, one student was a Digital Media graduate student at GT, and six students were majoring in Game Design at SCAD. Two participants were female. Pre-requisites for GT students were a senior-level class on game design and architecture, and completion of the standard introductory courses in their respective degree programs. The only pre-requisite for SCAD students was an introductory class in game design.

Initially, we conducted a two page individual survey which was completed by seventeen students during the final class period, and focused on individual experiences of each of the unique features of the class. Interviews were conducted with two focus groups, one of GT students and one of SCAD students after the completion of their fourth assignment. These probed collective and individual feelings regarding each of the unique aspects of the class; including, the dynamics of their collaborations, their understanding of AR, their experience with mobile development, and the overall structure of the class. Finally, group interviews were conducted with each of the five teams during the last three weeks of the class. These focused primarily on the group dynamics and design process.

3. RELATED WORK

Beyond the usual game design classes taught by us and others, and the various books on rapid prototyping of game ideas, the Experimental Gameplay Project at Carnegie Melon University particularly influenced us. Students in the ETC MS program began the Experimental Gameplay Project as an attempt to see how fast they could prototype novel game ideas[1]. The initial project lasted for one semester and yielded numerous interesting findings about the game design process [3]. At the center of this endeavor was the one-week rapid prototyping cycle required for the creation of each game. The apparent success of the EGP in regard to both the popularity of the games created and with the apparent learning that such a hands-on approach produced was a significant motivation for the original idea of this class. However, given the novelty of the technology, and the fact that we had both undergraduates and graduate students who were not experienced game designers or programmers, we extended the time frame for completed prototype to four weeks rather than one, and emphasized background skills and knowledge where appropriate.

4. CLASS DESIGN

4.1 Materials

The primary software package used for this class was Studierstube ES (STBES), provided by Imagination Computer Services. The software is written in C++ and provides all the basic APIs needed for completion of a mobile augmented reality game, including: marker tracking, model loading, scenegraph utilities (implemented in XML format) and an open game loop. Despite still being in development at the time of the class, this was the best choice for our needs due to its clean implementation, robustness, and the excellent support provided by the developers. Two additional software components were given to SCAD students to form a pipeline for content creation. One was a *model exporter* which functioned as a plug-in to Maya and allowed students to export the models and keyframe animations they created into the XML format readable by STBES. The second was a program, the *modelviewer*, which allowed for unit testing of art assets on the mobile devices without incorporating them directly into an STBES application.

The mobile device used for this class was the Gizmondo a handheld gaming device that contains a 0.3 megapixel camera, Bluetooth connectivity, and standard game controller controls (two shoulder buttons, four face buttons and a d-pad) arranged around a 640x480 screen. The Gizmondo also comes equipped with a Nvidia GoForce graphics processor, making it well-suited to basic graphics applications. It runs Windows CE 4.2 on an ARM9 processor, which does not have floating point capability. Each student was supplied with his or her own personal device for play-testing and content creation. All games and assets were stored on SD cards provided by the students. Additionally, all GT students were provided with Visual Studio 2008, and SCAD students were provided with Maya and other content creation tools. The class was not supplied with version control for their code, although some of the groups implemented this on their own. GT also provides T-Square, an implementation of the web-based collaboration suite Sakai[2], available to students, and SCAD students were given their own accounts in this system for the duration of the class. All students were encouraged to use this for blog posts, forum discussion, and code sharing, although very few took advantage of that functionality.

4.2 Class Schedule

The class was structured with one two hour meeting on Monday and one one hour meeting on Friday on the GT campus. In addition, SCAD students held a three hour meeting on Wednesdays on their own campus. Students from both schools regularly attended the Monday session together, while students attended sessions on their own campuses Wednesdays and Fridays. Additionally, students were encouraged to attend the sessions held on Wednesdays and Fridays at the other campus, but this was not a requirement. Students were required to meet with their teams outside of class at their convenience.

The first four weeks of the class consisted of lectures on basic topics in AR, including overviews of graphics and computer vision, related work, and historical context. Additional lectures included some topics which were relevant to mobile gaming in general, such as technological specifications for various mobile devices, and others that were relevant to general programming techniques including an overview of STBES. Additional lectures were added during the semester to address topics that arose in the course of game development. These were meant to address the concerns of students who felt deficient in some area of knowledge that was relevant to the completion of their projects.

[1] http://www.etc.cmu.edu/projects/experimentalgameplay/

[2] http://sakaiproject.org/portal

An important and unavoidable scheduling issue was that GT has a sixteen week semester, while SCAD has only a ten week quarter. This meant that SCAD students were not present during the first and last three weeks of the class, as their institution was out of session. Although we were aware of this while designing the course, and tried to adjust the syllabus accordingly, the impact of this discrepancy manifested itself in both obvious and subtle ways. We discuss these effects, and what they suggest about inter-institutional collaboration below.

4.3 Assignments

With SCAD operating on a quarter schedule, those students were not present during the first three weeks of class. We took this opportunity to focus on programming in STBES. In an approach similar to the hybrid one described in [4], we had these students complete individual skills building exercises that were intended to ground them in the techniques they would need to complete the group projects. This also gave the GT students, who would presumably be doing most or all of the programming, a chance to acclimate to the programming environment without the added complexity of game development, content creation and group dynamics. GT students were asked to individually complete seven tutorials, provided with the STBES package, on their own and demonstrate competency through the creation of an original program which incorporated elements of one or all of these tutorials.

The remaining four assignments were completed by teams of three to five people comprised of one or two SCAD students and two or three GT students per team. Teams were selected by the students with the only restriction that students with previous experience in AR (from a previous class or through our lab) could not work together. This was done to try and keep all the teams on a level playing field. For the second, third, and fourth assignments the teams created a playable game prototype and accompanying demo videos, which exemplified key elements of the gameplay. Each project was assigned as a three or four week exercise in rapid prototyping. The first week consisted of brain-storming game ideas and concepts, the second (and possibly third) weeks were devoted to feedback and implementation, and the final week consisted of presentations with feedback aimed at last minute tweaks and fixes.

As with the first three weeks of the class, SCAD students were not in session during the final three weeks. Consequently, we decided to have the final project cycle devoted to "remaking" a previous project; adding additional functionality and polish to a previous prototype, rather than having GT students make an additional game on their own.

5. DISCUSSION

5.1 Rapid Prototyping

5.1.1 Physical Prototyping

The use of rapid and paper prototyping in game design is well established. Even if the desired end product is an entirely electronic game, most designers will paper prototype throughout the process. There are numerous good reasons for doing so, and books such as [2] discuss these in detail. Underlying this practice is the assumption that higher-fidelity prototyping is replaceable by cheaper, faster, paper prototyping when designing the game mechanics and rules. However, this assumption does not necessarily hold for designing handheld AR games and experiences, because the students have little to no experience with these kinds of games. In contrast to traditional console or PC games, where the game interface is typically well defined (e.g., the player uses a controller or a keyboard and mouse to effect actions in the virtual world) and the game designers have a wealth of examples and personal experiences to draw on, in mobile AR games, the traditional input controls are secondary forms of interaction; the primary form of interaction is embodied physical interaction.

Figure 1. Collateral Damage game board and bomb card in action. The markers represent buildings and items in the game, which can be destroyed by the bomb card being thrown.

This embodied interaction between player and objects in the physical world, mediated through the mobile device, is the most critical element of AR gameplay. If this interaction is not properly designed, the player will simply not be able to play the game. It is therefore critical that these physical interactions be tested as quickly and thoroughly as possible, as they will have a major influence on game design. Comparing this again to more well established computer games, in AR one cannot simply assume that a physical interaction will "work" as part of a game, and design the game around this assumption, as one could for typical "1st person shooter controls" or "2D platformer controls". Beyond simply testing certain interaction ideas, it is important that the physical interaction be allowed to evolve with other aspects of the game design, which implies that the game exist as a prototype that can change and evolve with the design.

In general, we found that groups who prototyped live AR interactions the earliest were the most successful. At times these mid-fidelity prototypes provided the feedback that something was simply not going to work, while at other times they hinted at how something which was working might work much better. Either way, getting to these live prototypes as quickly as possible was a major indicator of the overall success of the project and quality of the game. Invariably, groups who spent too much time paper prototyping game design before beginning implementation failed to get functional games working on time. At their best, these attempts exhibited only a skeleton of a particular experience, rather than a completed game prototype.

Most teams engaged in a style of paper prototyping that we later dubbed *blind prototyping*, in which the designer essentially mimes his way through a game interaction without any live AR content. This proved to be a useful starting point, especially as

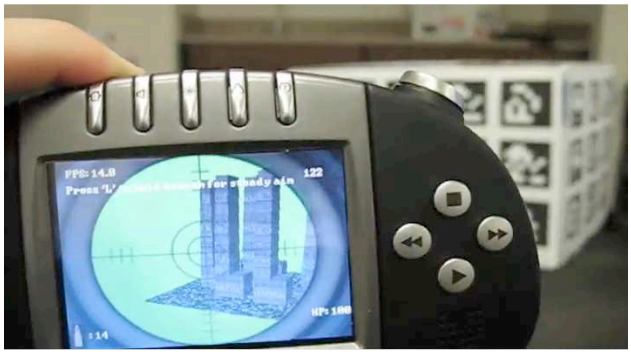

Figure 2. Joe Warpin played on multi-marker cube

the students gained more experience with the technology. However, particularly for earlier projects, this form of prototyped physical interaction can be misleading, as it does not capture the unforeseen intricacies and subtleties of active AR content. For example, in the game "Collateral Damage," (Figure 1) one of the most successful games produced early in the class, players take turns placing cards inscribed with AR markers onto the playing surface, effectively building a game board of resources as the setup progresses. Then players toss "bomb" cards which destroy all cards within their "blast radius," earning the player points.

During the initial blind prototyping stage, the designers assumed that the board would be "memorized" by the software and the "collateral damage" calculated would not be dependent on the player's location. However, once the online prototype was constructed, the designers realized that it was both easier to build the game and more engaging for the player, if the blast radius was dependent on how the player "framed" the board with the hand-held camera (i.e., controlled what other markers the camera could see and recognize while seeing the "bomb" card). Thus, the strategy of the game changed from simply trying to be accurate with throwing the bomb cards to one which asked the player to account for their "framing ingenuity" with the mobile device. Subsequently, the designers re-designed the game to leverage this new strategy, changing the method of score-keeping, and re-balancing other elements. It proved extremely difficult for the designers to get a feel for how the interaction in this game depended on the live AR content without their "mid-fi" prototype, and the realizations it provided had a radical effect on the end product.

5.1.2 Designing to Force Good Interactions

Another common adjustment that was made after live prototyping of physical interactions was the addition of certain constraints to the game design to force players to "interact properly." These adjustments worked to restrict the player's physical movements in such a way that accurate, smooth and consistent AR content is guaranteed. An example of this is the game "Joe Warpin" (Figure 2) which depends on the player's proximity to a multi-marker cube for gameplay.

The player assumes the character of Joe Warpin, an army sniper who is charged with the task of flying in a helicopter around a virtual city, and dispatching terrorists with sniper shots. This game is one example of a sub-genre that emerged in the class, that we refer to as *anchored virtuality*. In these types of games, the physical world is not visible through the screen on the handheld. Instead, objects in the real world act only as anchors for the interactions of the player. While rendering live video was used in some games, other teams decided that the jump in frame-rate that comes with not having to render a video image (a technical issue with the Gizmondo), was worth this trade-off. Still, this technique also amplifies the risk of losing tracking altogether, as the player cannot see markers on the screen, and tends to make the game feel more detached from the physical world (seemingly against the spirit and goals of AR gaming). However, in this case, the fact that the virtual world is "contained" in the area of the box, the motion of the player has them looking at markers in a manner that makes the tracking work.

To conserve on ammunition (the most vital resource in the game), players exhibited a natural tendency to try to get as close as possible to a target before firing. Because they could not see the physical world through the camera, they could not be sure they were keeping one of markers on the multi-marker cube in view at all times (a requisite for tracking). This resulted in a loss of tracking when they moved in too close during the aiming process, precisely when accurate tracking was needed the most. On the other hand, moving in too close made the game easier, requiring the game be balanced for the close distance. To solve both of these problems, the designers altered the game rules to inflict damage on the player, in the form of "enemy fire", if the player got too close to the game cube. This served to keep the player at a reasonable distance most of the time, and made them very conscious of when they were getting too close, assuring that the player's own physical movements would yield consistent marker tracking, and thus a smooth and natural game experience.

5.1.3 Getting it Right (or wrong)

The decision to focus the course on rapid prototyping provided constraints on the development of the students' projects in two important ways. First, it forced them to get their projects to a playable state as quickly as possible, giving them the experience with AR games and interactions that they needed to be successful. Second, it gave students the opportunity to try new ideas and possibly, to fail. The speed at which students had to turn an idea into a playable game sometimes meant sticking with a substandard design and making it work the best it could. Even so, students invariably learned from these experiences and were able to apply what they learned to future projects.

In a new domain where there are no design best practices and few examples to use as templates, giving students the opportunity to test their intuition and get feedback quickly in an iterative fashion provided the necessary traction for learning. Discovering the subtleties of a design decision in an early project often provided those designers with intuition about what might work for future designs, and what games might best take advantage of these particulars.

For some teams, the opportunity to create multiple projects gave them the ability to explore a particular mechanic in great depth and evolve an expertise in building it into their games. Figure 3

shows the three projects from one team. Each uses an essentially identical mechanic, placing round markers at various points on the playing surface, possibly in addition to other markers that formed the game board (in the case of the first two games). In each case, virtual game objects move to or from one circular marker to another. In the first example, "AR Puzzle Pacman," the markers act as "beacons" which attract a pac-man-like character and direct his movements around the board, or as gates whose orientation allows each of these characters to move in a particular direction. The center picture is of the game "Terrain," in which the circular markers act as either spawning points or beacons for different kinds of ants which form competing armies. In the third game, "Candy Wars," the circular markers are now magnets, stuck on a magnetic white board. These widgets also function as spawning points, this time for virtual candy whose trajectory you can change by adding additional widgets and setting their orientation.

These games represent a distinct progression in the way the circular markers are incorporated into the gameplay. In the first, they are the movable objects on a stationary multi-marker board. Both single and multi-marker tracking are standard approaches in AR, and although they showed great proficiency in the use of these, the game showed little novelty in how they are integrated with the gameplay.

The second project, Terrain, shows a leap in design logic; both the board and the objects are constructed of markers, like the previous game, only now the markers that form the board itself are movable and constructed "ad hoc" by the players (an idea inspired by the Collateral Damage game discussed above). Although the game mechanic is essentially an evolutionary step from AR Puzzle Pac-man, the overall complexity and fluidity of the interaction has increased. Where the player's placement of markers previously only determined the direction and final destination of virtual objects in AR Puzzle Pacman, she now has control over the starting point of this action as well. Moreover, where the board was previously pre-defined and required a large prebuilt gameboard, the player now has control over this spatial structure and builds it up during gameplay from the game pieces.

The third game in this series demonstrates another significant increase in the complexity of design and proficiency with the technology. Much appears the same when is compared to the previous games: for example, the placement of the now-familiar circular markers is still the primary mechanic and these still provide spawn points and goals for path finding. However, the multi-marker board has disappeared and the markers are freely placed anywhere on the playing surface. In the previous games the board functioned as the boundary for play. It provided the walls which directed the motion of pac-man, or the terrain the army of ants could march over. The user was given the task of getting these characters to their destinations unscathed. This is still the goal of Candy Wars, only now the boundaries are supplied by gravity and enforced by a 2D physics engine.

Candy Wars is intended to be played on a refrigerator or similar upright magnetic surface. Gravity, then, is an intuitive real-world constraint which is translated into the virtual world and bounds the possibilities for play in the same fashion as the board did in the previous examples. By integrating physics, the virtual and augmented worlds are more tightly and intuitively bound together. Despite its seeming simplicity, Candy Wars is technically sophisticated. Aside from the technical difficulty of incorporating external software APIs to create the necessary physics, the game takes the concept of "board-building" seen throughout these projects to a new level. Without the physical and virtual board constraining the actions of the player, this game is much more free-form than the others, and can even be used as more of a toy than a game if the user so desires.

As with each group, this team was required to iterate repeatedly, and forced to create a different game each time, even if it was based on a similar mechanic. Through this iteration, they developed a deep understanding of what was possible with their core mechanic, and could develop an intuition about what a good player experience in this domain would be like.

5.2 The Role of Board Games

While much game design education focuses strongly on video games, most game design programs also incorporate the analysis other types of games, such as board games, sports, and playground games into the curriculum as well. The reason is that these are good examples of game design principles in action and provide historical and cultural grounding for the analysis of the games students are expected to create. Our class focused on board games as discussion tools and for design inspiration, for two reasons. First, the technical aspects of the current generation of handheld AR hardware and software are amenable to the "boards and cards" kind of games we had the students play. Second, we are very interested in encouraging multi-player, social games as a design target.

For the first two prototyping cycles we spent the first two-hour class period of the brainstorming weeks playing and analyzing

Figure 3. Three games with the same physical mechanics. From left to right: AR Puzzle Pacman, Terrain, Candy Wars.

board games. Groups chose one of several dozen board games which they played for the first half of the class period. In the second half of the class, each group was asked to analyze and describe the formal aspects of the game (rules, mechanics, genre, etc.) and to discuss what was most compelling or effective about it and how those aspects might be transferred into a mobile AR game in a *meaningful* way.

The original plan was to begin each of the prototyping cycles with this brainstorming activity. However, after two cycles, we felt that the usefulness of this activity had reached a point of diminishing returns. Students had already demonstrated familiarity with game design concepts and an ability to identify aspects of game systems suitable for AR applications. Additionally, because many students had limited game design experience, we noticed a tendency in many of the groups to adhere too strongly to the elements of the games they were exposed to, at times almost mimicking game-play exactly. What had originally functioned as a tool to inspire creativity was now acting as a constraint on it. Further evidence for this interpretation appeared in interviews and surveys with the participants, many of whom echoed these sentiments exactly.

Instead, we decided to switch to a more traditional form of idea generation. Now that the students had experience with AR, we had them do project pitches as we would in our other design classes. Therefore, for the third project brainstorming period we had each student (not group) do an "individual pitch" in which they gave a two minute presentation of an original game idea to the class, allowing students to choose which ideas they thought were best and wanted to work on for their next project. This exercise also served the additional purpose of forcing students to present their ideas publicly, as some tended to fade into the background during group presentations.

As mentioned above, in all design pitches and project work, we asked that students focus on the *meaningful* incorporation of AR technology into the games they created. This was done specifically to counter a common trend seen generally in the use of AR and other novel technologies, to use it as a gimmick or "eye candy" rather than creating experiences or games that leverage the technology in a deep and meaningful way. In our own work, and in our early discussions in this class, we have noticed this tendency among the students; that is, to imagine experiences and games that put 3D graphics and animations "into the world" but that add nothing to the gameplay beyond what you would get with a desktop or console game. While such eye-candy works well for demos and garnering interest in the subject matter, a primary aim for the class was to have students contemplate the value of AR more deeply. In classroom discussion and feedback, the instructors were constantly asking the question "Why AR?" to prompt students to focus on the aspects of their games to which the use of AR actually added something meaningful to the game experience. In general this was effective, as students learned to expect this question and have a well-thought out answer prepared.

Ironically, some of those students who were taking the class with the intention of generating visually striking work to include in their portfolio reported becoming gradually more annoyed by this question, as they were sometimes asked to give up on ideas that would have produced rich, stylized content in favor of ideas

Figure 4: Friggin' Lasers. Virtual lasers reflect off of virtual objects

that may not have been as visually compelling, but had the potential to reveal something novel about AR gaming.

One project, "Friggin' Lasers," (Figure 4) could be said to have fallen victim to some of the problems outlined above. The designers drew their inspiration from the game "Khet," a chess-like game where players orient mirrored pieces on a grid in an attempt to create a reflected path for a laser light fired from their home base to their opponent's. The game works well in the real world because it uses lasers and optics in a new and interesting way, while leveraging many of the strategies common to games like Chess and Go!.

While the game did eventually turn into an interesting and playable prototype, the use of AR felt hollow. The designers used the "laser and mirror" metaphor too literally (almost piece for piece in their early designs), and failed to find a good way to integrate AR in a meaningful way; either as an interesting new mechanic for physical interaction or through leveraging possibilities in virtualizing the lasers. In contrast to the successful integration of virtual gravity in Candy Wars, a game with a similar mechanic, Friggin' Lasers did not benefit through virtualizing the physics of optics because the interactions were not novel, but merely digital analogs of interactions already present in the board game.

5.3 Inter-Institutional Collaboration

5.3.1 Materials Feedback

Overall, students were able to design and program effectively with STBES and the Gizmondo, and by the fourth assignment they all demonstrated an evolved understanding of what the software and hardware were capable of doing. The games themselves also exhibited increasing sophistication and complexity in their designs and interactions. In this section, we give an overview of our experiences, good and bad, with a cross-disciplinary/cross-institutional collaboration such as this. Many of these points will sound familiar to those who have attempted such collaborations, but we hope will prove useful to those considering such an endeavor.

Both GT and SCAD students expressed frustration with the lack of documentation for STBES; contrary to their more traditional programming and design classes that use well-established software packages, students were unable to simply search the web for STBES programming and content creation examples and tutorials. This meant they would often have to spend time

implementing low-level functionality or experimenting with the content pipeline, and had to scale back their ambitions in order to deliver a functional prototype in time for the deadlines. However, students did learn to re-use some of the low-level functions from previous games. Groups also swapped code with each other to get functions they wanted for new games, and in the interviews it was generally suggested that there be a more formal process for doing this, as most found it helpful.

The lack of documentation and occasional instability of STBES and its components was more pronounced for the SCAD students. Since the educational focus of SCAD is on the creation of creative content using well-established tools, using software that is still in development sometimes damped the enthusiasm these students began with. Fortunately, a number of bug fixes from the developers were released to the SCAD class and these cleared up many of the issues with the content pipeline. Once this pipeline was firmly in place, SCAD students were able to focus their time on content creation rather than troubleshooting, and became more engaged in the process overall.

A different issue was the difference between the mobile and desktop platforms. SCAD students universally expressed surprise with the limitations of the supplied software and hardware, which they referred to collectively as "problems using Gizmondo." For example, they had to create low-polygon count models in order to ensure a frame-rate that made their games playable; this is a common limitation on current mobile platforms that will be ameliorated to some extent in the coming years by new hardware. Other technical limitations included limited animation support, and a 256x256 maximum texture size. Some students felt that it was an interesting challenge to create high quality content given these constraints, while others felt that their hands were tied and their modeling and animation skills were not being fully utilized. Educationally, we feel this particular aspect of the experience was more beneficial than students at both SCAD and GT realized, as such problems with adopting new technology are extremely common in "the real world." Particularly, those students who expressed an interest in entering the video game industry will likely be reminded of this experience when a new console, OS or SDK is released, and many of the issues they faced with this software and hardware re-emerge on a different scale.

5.3.2 Schedule Feedback

The most significant factor in the design of our class was that of "home-field advantage," which in this case translated into SCAD students feeling as if they were "guests" in a GT class and not full-participants. This attitude manifested itself mainly as SCAD students reporting that they sometimes did not feel ownership of their projects. Their lack of technical knowledge also seemed to contribute to this feeling. Since STBES was intended primarily as a programming tool made by programmers, SCAD students initially had a difficult time intuitively understanding what its limitations were. However, they did eventually learn how to think "inside the box" the software and hardware provided, as evidenced by major design contributions in the third prototyping cycle.

This problem should not be ignored, and based on our past experiences and conversations with others who have attempted cross-institutional collaboration, we do not believe it was simply a result of our particular circumstances. Care should be taken in leveling the playing field, and it is not always easy to accomplish. For example, due to institutional restrictions, GT students could not be required to attend sessions held on the SCAD campus; however, as the semester progressed, it was clear that having them do so would have been very useful. Those GT students who did attend sessions at SCAD reported that their SCAD partners were much more engaged in the process overall. Likewise, SCAD students whose GT members "made the trip" to their campus also reported feeling more ownership over design decisions. even though there was no evidence that changes had actually been made during this time.

The second major effect of inter-institutional collaboration was less specific and is best understood as a lack of "quality time." Although each team was required to meet at least once a week outside of class, only a minority of teams reported doing so consistently, and most coordinated solely over email. GT students typically programmed on their own; adding content created by SCAD students at the last possible minute. On the SCAD end, this often meant that they had no input in any design changes that needed to be made due to technical limitations that manifested during the implementation stage. The lack of time spent together is common in group projects within one campus, and the distance only exacerbated the usual practical concerns, such as scheduling issues, the availability of computing resources to program or model, and the frequently referenced "six dollars for parking."

5.3.3 Moving Forward

As we plan for a repeat class next year, we feel that a combined "studio-session" (i.e., lab hours with required attendance) held in a facility with resources for every step of the process from brainstorming and design to programming, content creation and playtesting, could conceivably address most of the major concerns. During a studio session instructors would be available to answer technical and design questions as they arose, therefore addressing relevant issues when they are needed most. Common complaints with delivering feedback on progress after in-class presentations were: students would rather have been working on their projects than preparing the presentations; the feedback occurred at the wrong time and it was too late to incorporate without radical design changes; feedback was not focused enough on what they felt were the "real issues," by which they meant technical concerns that were eating up their time and thus taking time away from refining gameplay. Getting this feedback from instructors in real-time in a studio setting would eliminate all these issues. This would also, presumably, speed-up the prototyping cycle, as solutions would come more quickly and be available to everyone at the same time.

Another benefit of running the class with more mandatory lab hours is that it would allow content creators and programmers to work simultaneously, creating a positive feedback loop that solves the problem of each sub-specialty not being familiar enough with the issues facing the others. In this model, those focused on content creation are exposed to mundane technical issues and gradually become more familiar with the technical constraints that affect design. Likewise, programmers would be able to understand the capabilities of content creation tools that they have little exposure to in classes. Everyone could still do what they do best, but they would be working together and

engaging with the same tools and artifacts simultaneously. In principle, this structure could allow for much tighter cohesion within the team and provide the opportunity to integrate individual units more frequently, giving them a sense of the "look and feel" of the game much earlier in the process. This also jibes nicely with the above discussion on the need to move from paper prototype to interactive prototype as quickly as possible.

5.3.4 Assignment Feedback

Naturally, the individual abilities and skills students arrived with vastly affected the overall quality of their work; and some students had more difficulty with the open-endedness of the assignments than others. Our objective with the first assignment was to get all students to the same basic comfort and skill level with STBES. The technical students ranged in ability from those who were able to integrate the tutorials and deliver tight and clever programs, to those who had no significant experience with C++ and Visual Studio. Although, the majority of students successfully completed this assignment on their own, they also felt that being asked to creatively combine tutorials into something new as a first assignment was too confusing and open-ended. They suggested, instead, that spending the first few weeks completing shorter, more prescribed assignments that could later be used as the low-level building blocks of the larger projects would have helped them get a better grasp up front and possibly even saved time, or allowed for more functionality, in later projects.

Although the decision to give students the chance to remake a previous project as their fourth and final assignment was partially made for pragmatic purposes, the students universally reported that it was advantageous in a number of ways. First, it allowed some to fix what was broken in previous games, and gave them a sense of "closure." One of the inherent limitations of rapid prototyping is that you do not have time to re-design what isn't working immediately before a deadline. In some cases projects met with unforeseen difficulties during implementation and work-arounds needed to be implemented. This sometimes left students feeling unsettled about the quality of their work, and they appreciated the opportunity to re-examine their failures. Knowing that they would have this opportunity later made it easier to let go of a project and move on to the next.

Second, students were able to add features that they did not have time for in the first iteration. In some cases, students really liked their games and wanted them to be perfect, but did not have time to add subtle but valuable features the first time around. The ability to refine their favorite game was particularly important to senior and graduate students who wanted to have high-quality work to include in their portfolios. Third, by the time the final prototyping cycle came around, students reported that they were burnt out from the pace of the class and effort required to complete earlier projects and did not have the energy or ambition to begin an entirely new project from scratch.

6. CONCLUSIONS

The instructors all saw evidence that students had gained considerable knowledge of issues in game design, AR and related disciplines, and project management. Clearly, everyone developed a more sophisticated intuition about what makes good handheld AR games, and we were able to distill some generalizable design practices and game categories from common features found in the games. Many of the sixteen game prototypes produced in the class exhibit more natural and substantial AR interactions than are commonly seen in current research prototypes or product demos, and we felt that some even have a chance at commercial success.

Despite typical technical, logistical and social sticking points, we believe that we have the groundwork for more successful inter-institutional collaborations in the future, and intend to implement the changes mentioned above in the next iteration of the class. Additionally, we plan to widen the scope of these collaborations into classes related to film, animation, games, and other areas of entertainment.

7. ACKNOWLEDGMENTS

We would like to that Georgia Tech and SCAD-Atlanta for giving us the opportunity to teach this class, and to the students who took it and worked so hard to make it a success. Additionally, we would like to thank the Media Power Group for support, especially the donation of Gizmondo's for the students to use, and for Imagination and the Technical University of Graz for the use of, and support for, the STBES programming environment.

8. REFERENCES

[1] Boudreaux, Hollie Jim Etheredge, Timothy Roden. "Adding Handheld Game Programming to a Computer Science Curriculum." Proceedings of the 3rd international conference on Game development in computer science education (GDCSE'08). Miami, Florida, 16-20.

[2] Fullerton, Tracy, Christopher Swain, Steven Hoffman. 2004. Game Design Workshop: Designing, Prototyping, and Playtesting Games . CMP Books, San Francisco, CA.

[3] Gabler, Kyle, Kyle Gray, Matt Kucic, Shalin Shodhan. "How to Prototype a Game in Under 7 Days: Tips and Tricks from 4 Grad Students Who Made Over 50 Games in 1 Semester." Gamasutra, October 26, 2005. [http://www.gamasutra.com/features/20051026/gabler_01.shtml]

[4] Wolz, Ursula. "Teaching Game Design through Cross-Disciplinary Content and Individualized Student Deliverables." Talk presented at Microsoft Academic Days Aboard the Disney Wonder Cruise Ship Orlando, Florida February 23, 2007. Available from [http://www.academicresourcecenter.net/curriculum/pfv.aspx?ID=6936#Download]

[5] Roden T., and Etheredge, J. "Educating Game Programmers." In Proceedings of the 2007 Microsoft Academic Days Conference on Game Development (GDCSE'07) (Orlando, Florida, February 22--25, 2007), 510--514.

[6] Settle, Amber Joe Linhoff, Andre Berthiaume. "A hybrid approach to projects in gaming courses." In Proceedings of the 3rd international conference on Game development in computer science education(GDCSE'08). Miami, Florida, 36-40.