Teaching Game Design through Cross-Disciplinary Content and Individualized Student Deliverables

Ursula Wolz
Computer Science and
Interactive Multimedia
The College of New Jersey
Ewing, NJ 08628
+1 609 771 7266
wolz@tcnj.edu

Christopher Ault
Interactive Multimedia
The College of New Jersey
Ewing, NJ 08628
+ 1 609 771 2236
ault@tcnj.edu

Teresa Marrin Nakra
Music
The College of New Jersey
Ewing, NJ 08628
+1 609 771 2759
nakra@tcnj.edu

ABSTRACT
We have created a course structure for Game Design that we use to teach collaborative software engineering principles. A core feature of our approach is the cross-disciplinary collaboration that purposely avoids a balkanized Computer Science perspective. Our multidisciplinary faculty have mentored upper level students from different majors in a large joint game development project. Both the course and the group project serve as a vehicle for a collaborative cross-disciplinary experience in technology development. This paper addresses the issues that we addressed in curriculum structure, student assessment and cross-disciplinary team teaching. We also present our dual model of *cross-disciplinary content* and *individualized deliverables* for personalized student learning outcomes. Our development model also emphasizes principles of Project Management to keep many students with disparate skills and tasks focused on the overall group goal.

General Terms
Software Engineering, Games

Keywords
Game Design, Game Architecture, Project Management, Multidisciplinary computing, Computer Science Education

1. INTRODUCTION
There has recently been a resurgence of interest in using games as a domain of application for teaching computer science fundamentals. Video game design, development and architecture are also gaining credibility as disciplines in their own right within Computer Science, drawing from such areas as Computer Graphics, Artificial Intelligence and Networks. Video game design can also be used as a vehicle to teach software engineering principles in the construction of both game engines and to the creation of games themselves [2, 3, 4]. However, an unrealistic and artificial environment is created when the software engineering experience in game implementation is limited to computer science students. The construction of a fully robust and engaging video game is dependent upon expertise from

disciplines outside computer science including creative writing, music composition, sound technology, theater production, digital 3-D art, cinematography, and character animation. To a large extent game design and implementation is a compelling model for a more global question of how to teach skills in cross-disciplinary technology development. As was eloquently stated in "The World is Flat" [5], the skills necessary for collaborative communication across disciplines will be critical to the continued success of the American workforce. This is coupled with the timely resurgence of interest in video games themselves as tools for education [7]. There is a predicted need for computer scientists with expertise in game development who have practical experience in cross-disciplinary collaboration.

At our institution a multidisciplinary faculty from Communication Studies, Computer Science, Digital Art, Interactive Multimedia, Music and Professional Writing raised the question of how to exploit video game development as a vehicle for a collaborative cross-disciplinary experience in technology development for upper level students from a variety of majors in a single, year-long cohesive course. This paper reports on our experience with this approach, addressing the major issues of curriculum structure (Section 3), individual student assessment (Section 4), and cross-disciplinary team teaching (Section 5).

In particular, we present a dual model of *cross-disciplinary content* and *individualized deliverables*. This supports assignments that allow each student to determine how narrowly or broadly to focus their personal learning within the breadth of disciplines. Furthermore, this approach provides a model for student-centered learning that attempts to dismantle the "silo" effect of undergraduate education that creates balkanization of disciplines.

Our original goal was a suite of courses that would share the objective of creating a single 3-D, multiplayer virtual world with a robust story line supported by high quality sound and music. We envisioned that students would collaborate based on highly developed expertise in their chosen fields.

We anticipated that established models of project management, where team members report through hierarchical organizations of skills-based accountability, would suffice to facilitate design and production.

Table 1: Student Roles in Game Development

Title	Deliverables	Responsibilities
Art Director	Finalized and stylistically consistent art, story and sound	Overseeing the work of the art and writing staff
Tech Manager	Solutions for trouble shooting technical issues.	Overseeing the tech staff and liaison to art staff
Project Manager	All the assets completed on time and in the right format	Managing the workflow of the project as a whole
Prop and Scenery Modelers	All non-character models and background models	Working with 3-D modeling tools
Character Modeler	Two main characters, three secondary characters	Working with 3-D tools to model, and animate all characters
Level Designer	Map of all games levels, identifying interactivity triggers	Creating level maps and trigger points
Texture Designer	Textures for props, characters, and all other surfaces	Working with 3-D tools to create “skins”
Lighting Designer	Lights placed appropriately throughout the levels	Working with game engine tools to insert lighting
Story Analyst/ Writer	Game implementation consistent with established story	Creating dialog and directing voice actors
Game play writer	Actions associated with interactivity triggers	Identifying all trigger points and the action
Documenter	Web site for informational and publicity purposes	Standardize all documentation including tutorials
Support Software Manager	Installation and maintenance of all support software	Maintaining software including exchange server
Composer	All music necessary for all levels.	Create music that enhances the game play
Sound Technician	All sound effects including dialog.	Ensure integration of all sound effects and dialog
AI Technician	All logic and algorithms for game implementation	Implementing logic for the game
Interface Designer	User interface elements	Creating an intuitive player interface
Media coordinator	All assets developed by artists are correctly installed	Responsible for inserting all assets into game

2. INSIGHTS FROM OUR PILOT YEAR

Academic year 2005-2006 (AY05-06) was our pilot year in which approximately 20 students per semester were enrolled in courses that supported this enterprise. A once per week four-hour workshop allowed students to participate in the collaboration through specific roles (summarized in Table 1). This was intended to model the professional game development environment if not the 24/7 intensity of the industry. In 30 weeks we produced of complex high resolution and one rough cut level of a modified “first person shooter” with full sound and high quality music. We used the ValvE “Source” game engine. We also produced a story bible and sample assets for a full 3-D multiplayer game (see http://www.tcnj.edu/~Games.)

Limitations of a Product-Centered Model

Based on analysis of student work throughout the year, we concluded that our initial assumptions about course structure and organization were off the mark for a pedagogic environment. Our mission is to educate students. The goal of producing a fully robust game was merely the vehicle through which to teach concepts and skills. Our pilot showed that the kind of narrow task assignment we envisioned, with students segregated by skills (e.g. by major from the contributing disciplines), severely constrained students' ability to grow, learn and simply communicate in unanticipated ways.

Furthermore, we confirmed work by Constantine and Gillard [1, 6] that traditional hierarchical models of team organization with linear models of time management are insufficient. They do not support the cross-disciplinary communication necessary to create software as complex as a video game. We initially identified "programmers," "artists," "writers" and "sound composers" whose work would be integrated through production leaders (who emerged from the ranks.) In practice we saw, not unexpectedly, that individual artists needed to collaborate with individual programmers and writers. Clean boundaries between groups clustered by expertise thwarted the development process, reinforcing balkanization and discipline-based prejudice. Our simple models of collaboration and group expectations based on skill sets for deliverables were insufficient. They did not provide students with the rich immersive experience required to meet our initial pedagogical goals of collaborative, multidisciplinary software development.

This also made assessment problematic. We had clear and established criteria for judging art work, writing and programming. However, this also reinforced balkanization. Traditional assessments of specific deliverables discouraged students from collaborating because their role in producing a specific outcome wasn't always clear. It also discouraged students from taking risks and trying their hands at developing skill sets outside their area of expertise. Why produce a terrible 3-D model that wouldn't be used in the game anyway if you can make a clear and valuable contribution with a novel algorithm?

Furthermore, how does an instructor measure the worth of a deliverable when everyone is making a unique contribution?

Our original course framework envisioned a fall semester experience in which students learned the theory and craft necessary to spend the second semester in a large group, single project collaboration. For the fall semester we anticipated at least five “courses,” one each in of the contributing disciplines (Digital Art, Communications, Computer Science, Interactive Multimedia, Music and Writing.) Given the size of the student population from which to draw (we are mid-sized primarily undergraduate college) as well as a realistic expectation for managing a single class project, we could not support faculty load or student enrollment at reasonable levels with so many courses. Also, conceptually we were balkanizing the disciplines, and needed to create a truly multidisciplinary experience that would meet load constraints.

A significant anticipated problem was how to recruit the right balance of students with diverse expertise and then exploit the emerging technical and leadership skills of the group that came together. An added complication is that we could not assume that all students who registered for the fall class would continue in the spring. The course offerings were constrained by upper-level in-major requirements and thus we could not necessarily expect a full year commitment from all students.

The “Silo” Model Creates Balkanization

The segmented or “silo” model of contributing disciplines was problematic with regard to faculty load, course scheduling, and assignment of student seats. The topic foci were not equal in required breadth and depth. Faculty availability to teach within the suite was not consistent, nor was there a balanced need for faculty expertise. To implement our idealized model we would overburden some faculty (e.g. supporting 30 animators in one class, while supporting a single music composer in another.) Furthermore our assumptions about domain expertise did not account for subfields within a discipline.

For example a networks expert might require extensive professional development to teach the requisite knowledge in Artificial Intelligence. Similarly, a cinematographer would need training to teach 3-D animation.

We sought a coursework model in which the content and skills presented could be taught with broad strokes appropriate to students with a range of expertise while requiring students to delve into content and develop skills commensurate with their background and interest. Put differently, student assessment would be based on a metric that balanced generic accountability (e.g. everyone completes weekly journals on reading assignments) with highly individualized targeted measures of skills development. Furthermore, to de-balkanize the disciplines we needed to reward students who were willing to leave the comfort-zones of their major, and who took risks by completing assignments from other disciplines. Their experience out of their major should create an enthusiastic respect for the work of others and give them a vocabulary through which to communication across disciplines. Such communication is key to effective cross-disciplinary collaboration, which in turn is key to reducing the silo effect.

3. CROSS-DISCIPLINARY CONTENT

In May 2005 seven faculty members, representing six undergraduate majors (Art, Communications, Computer Science, Interactive Multimedia, Music and Writing) and including CS faculty in Artificial Intelligence, Interface Design and Networks, participated in an intensive workshop to design a yearlong experience in collaborative cross-disciplinary, 3-D, multiplayer video game development. We articulated the need for content teaching in five overarching areas: game genre, interactive storytelling, game engine architecture, production management and the social and ethical impact of games. We also identified three primary technical areas: character animation, interactivity and artificially intelligent agents. Furthermore we identified three secondary technical areas: Audio (including dialog, sound effects and music composition), Theater Production (including staging and lighting), and Computer Networks (for massively multi-player games.)

Table 2 shows our resulting course structure that is a mix of (1) formal lectures of cross-disciplinary content, (2) studios for technical skills development, (3) demonstrations/practice in support software (e.g. Maya, Reason, XSI), and (4) workshops for product development. We organized the two-semester experience to frontload content and skills development. Table 2 provides an overview.

The class is scheduled as one four-hour block per week for two fifteen-week semesters. Lecture topics typically occur during the first 90 minutes. On lecture days, the remaining post-lecture time is used for workshop, tutorial and studio. Whole group demo days (social ethics days, demonstration, focus group/product testing) have a structured agenda with assigned responsibilities.

Significant course content and skills development occurs outside of class through (1) assigned readings, (2) small group meetings, (3) small group and individual tutorials (with both faculty and students as tutors), and (4) individual and small group design sessions, graphics development, and sound/music recording sessions. We also use the Microsoft Sharepoint server as a conduit and repository for materials development. Students develop all game assets, most technical tutorials, as well as documentation and time lines. Faculty materials are primarily lecture notes and assignments.

Table 2: Year-long Syllabus and Faculty Roles

Week	Presentation Topic	Assignment Type	Student-centered	Faculty Participation
1	Game genre		P	IR
2	Interactive storytelling	1. Art	P	G
3	Game Architecture	2. Tech	P	IR as G
4	Project Management	3. Mix	P	IR as G
5	Animation	4. Art	P	IR as G
6	Interactivity	5. Mix 6. Tech	P	G
7	Agents	7. Tech	P	G
8	Workshop: Story & game design		F	G & IR
9	Social & Ethical Impact I		N	IR
10	Sound, Dialog, Music	8. Art	P	IR as G
11	Theater Production Design	9. Mix	P	G

12	Networks for MUDDS	10a. Art 10b. Tech	P	G
13	Social & Ethical Impact II		N	G & IR
14	Workshop: deliverables review		F	IR
15	Focus group: deliverables demo		N	G & IR
		SEMESTER	BREAK	
16	Story presentation, roles discussion		P	IR
17	Time line, responsibility articulation		P	IR
18	Workshop		F	IR
19	Social/Ethics Impact Analysis		P	G & IR
20	Workshop		F	IR
21	Focus group: Low res demo		N	G & IR
22	Time line, deliverables review		P	IR
23-25	Workshop		F	IR
26	Focus group: High res demo, Social ethics impact analysis		N	G & IR
27	Time line, deliverables, product review		P	IR
28-29	Workshop		F	IR
30	Product unveiling		N	G & IR

Legend: F – fully student-centered G: Guest lecturer
P – partially student-centered IR: Instructor of Record
N – not student-centered

The formal lectures give an overview of essential topics based on assigned readings from a mixture of textbook genre (one computer science, one digital art/writing, one reflective). The lectures are designed to make the material accessible to students with little background, for example so that a programmer can appreciate the complexity of animating a character. The lectures are also designed to give a novel perspective to majors in that field. For example, a Computer Science major who has taken Artificial Intelligence learns to appreciate the impact of agent technology on game play.

We fully integrate the social and ethical impact of video games into the curriculum through class discussions of story line, character development, visual images and stereotypes. We also focus on these issues in two full sessions in the fall semester. The students create a forum in which timely issues are discussed. One session is a closed dry run. The second session is well publicized and open to the public. We return to these issues in the spring semester as we evaluate the implementation stages of the game.

4. INDIVIDUALIZED DELIVERABLES

Table 2 lists the degree to which learning in class is individualized. A "fully" individualized session occurs on studio/workshop days. A "partial" session occurs on lecture days when approximately 1.5 hours are devoted to lecture and 2.5 to studio/tutorial/workshop. The presentation days contain no individualized instruction. Only 19% of instruction over the year is lecture based, while 20% is student presentation and 61% is studio/workshop. If all students were developing similar skills and submitting similar deliverables this would be a traditional art studio. However, the studio/workshop time may involve a variety of tasks from the contributing disciplines. This requires a novel approach to support and assessment of student deliverables.

Student work in AY05-06 led us to both constrain and loosen course requirements. Undergraduates tend to cram for tests, and deliver less than optimal results for project deadlines, hoping for extensions. This style is a severe detriment to projects with heavy task interdependency. Time analysis surveys, administered both at the middle and end of AY05-06 showed that half of the students had poor abilities in managing deliverables and a third had insufficient time management skills for successful collaboration.

In reflective essays, students asked for help in time management and more direct accountability of weekly deliverables. The faculty concluded that students needed to be explicitly taught benchmark and dependency analysis skills, and needed carefully guided practice in fulfilling a weekly action item.

These results led us to redesign student assessment in both semesters. We defined a new methodology of *individualized deliverables* that provides a highly personal set of expectations. We can assist students in constructing a set of expectations that meet their personal learning goals (e.g. everyone doing something different), while fostering task dependencies that enhance collaboration in a safe way (e.g. one student's grade is not critically dependent upon another's work.) In AY06-07 we are combining generic expectations with an individualized contract of student deliverables. Final grades are assigned as follows:

Fall Semester

10% Journal entries: acceptable, insufficient, or not completed
20% Final take home exam, 20 questions based on journal entries
20% Lecture follow up assignments choose 5 out of 10
50% Project deliverables: percentage effort on 3 of 4 projects

Spring Semester

15% Journal entries: acceptable, insufficient, or not completed
20% Final take home exam, 20 questions based on journal entries
15% Benchmark recording and Weekly Action Item Report:
50% Project deliverables: percentage effort on 3 of 4 projects

4.1 Generic Assignments for All Students

All students are required to complete reflective writing assignments via journaling and a take home essay final exam on (1) course content, (2) personal skills development, (3) social and ethical impact, and (4) collaboration and communication.

All students are also required in both semesters to present an individual deliverables contract. This is an evolving document that includes a percentage breakdown of (1) selected assigned exercises, (2) contribution to large projects, and (3) time line and dependency analysis.

In the Fall semester, correspondence on individualized deliverables occurs through documents submitted to a course management system "drop box", through email correspondence and face-to-face meetings with the instructors of record.

In the second semester we use an accountability technique developed at the end of the AY05-06. Each workshop session begins with a review of action items from the previous week. Each student checks in by reporting on the status of the item and whether (1) a deliverable is ready for full group demonstration and evaluation, or (2) is ready to be included in the next version of the game. At the end of the workshop each student checks out by identifying his or her action items for the week to come.

4.2 Breadth vs. Depth of Skills Development

In the fall semester, which is more content based, students must choose 5 of 10 lecture summary assignments from the contributing content areas. Table 2 shows them broadly categorized as “technical”, “artistic” or “mixed.” These exercises create opportunities to de-balkanize perceptions of skill sets. Students can select assignments that let them remain safely within their general area of expertise. But they must complete at least one exercise at the edge of their safety zone. For example a Computer Science major could play it safe by selecting only “tech” or “mix” assignments (e.g., 2, 3, 6 ,7 and 10b) staying well within the bounds of Computer Science. A more adventurous technical student might select 1, 2, 3, 5 and 10b, adding a few exercises that are “mix” or “art.” A technical student willing to significantly broaden her background might select all of the exercises outside her safety zone (e.g., 1, 4, 8, 9, and 10a).

Table 3 show the results of Fall 06 exercise submission. Students are identified as “T” for tech or “A” for art based on their registration in one of two co-listed courses. The “T” students were primarily Computer Science majors; the “A” students were primarily Interactive Multimedia (IMM) majors. Grading was done on a 4-point scale: exceeds (4), meets (3), almost meets (2), and does not meet (1) expectations. There is no significant difference in grades between the two groups, although, as expected, IMM students did slightly better than CS students on art. Surprisingly, CS students did better than IMM students on mixed assignments. Three interesting phenomena emerged:

Table 3: Distribution of Assignments

Assignment	**% of Media Students**	**% of CS Students**	**% of All Students**	**Avg grade Media Students**	**Avg grade CS Students**
1. Story (art)	56%	11%	33%	3.4	3

4. Animation (art)	33%	22%	28%	3.7	3
8. Sound/music (art)	78%	56%	67%	4	3.9
10A. Networks (art)	44%	22%	33%	3.5	3
3. Project Management (mixed)	0%	33%	17%	N/A	3.3
6. Interactivity (mixed)	33%	33%	33%	2.7	3.7
9. Theater (mixed)	44%	78%	61%	3.3	3.3
2. Architecture (tech)	0%	33%	17%	N/A	4
5. Interactivity (tech)	0%	0%	0%	N/A	N/A
7. AI agents	0%	11%	6%	N/A	3
10B. Networks (tech)	0%	44%	22%	N/A	3.3

First, none of the Multimedia students attempted the "tech" activities, and they remained for the most part safely within the art activities. This appears to be an indicator that the "tech" activities were perceived as well beyond the ability of the Multimedia students, despite the fact that they were required to read a minimal amount of C++ code and were not required to write any code. (All Multimedia students have had, as a prerequisite, the equivalent of a CS 1 course).

Second, the Computer Science students were willing to take risks outside their area of expertise, and in fact did a proportionately larger number of "art" and "mixed" activities than "tech" activities. Viewed differently, the tech activities were perceived as hard by all of the students. Looking closely at Table 3, their bias was toward the music (56%) and theater (78%) activities that did not require extensive drawing or design expertise. They fully appreciated that the IMM students could outperform them on the art/writing (animation/story/design) activities because of the latter groups' more developed expertise.

Third, both groups overwhelmingly (67% sound/music and 61% theater design) chose the two assignments outside the area of expertise of both groups. None of the students had extensive formal training in either, although many were amateur musicians. Especially on the music assignment, the students consistently exceeded expectations. Given their lack of formal training, they created surprisingly good compositions.

4.3 Large Project Collaboration

Large project collaboration provides the focus of both content mastery and skills development. Such work is crucial to de-balkanizing the constituencies and providing an environment that fosters cross-discipline communication. In the context of highly individualized roles (see Table 1), we ask students to choose their participatory role and identify deliverables in a highly personal way. In order to prepare them for the second semester when their contribution will be critical to the whole, we ask them in the first semester to split their personal deliverables between at least 3 of 4 projects. They tell us what percentage of their project grade will come from their contribution to each project.

Table 4 summarizes the commitments made at the beginning of the Fall 06. All students are required to commit at least 10% to the "ethics" forum (and a maximum of 80% to any one task). The "story bible" requires creating the backstory, interaction framework and characters for the game to be implemented in the spring. The "enhance last game" project develops skills in our SDK, pipeline processes and support tools. The "toy engine" project provides in-depth experience in augmenting a game engine. As shown in Table 4, students approached the deliverables from many perspectives ranging from deep commitment to a problem (e.g. story bible at the maximum of 80%) to even distribution (all four projects at 25%). Table 4 shows the expected self-selection, where "T" students tended toward implementing the toy engine and "A" students tended toward the story bible. However, the table also shows significant risk taking on the part of all students.

Table 4: Percentages of Individual Deliverables

Student	Ethics	Toy Engine	Enhance Last Game	New Story Bible
A1	10		40	40
A2	10		70	20
A3	20	10		70
A4	30	10		60
A5	80		10	10
A6	10		60	30
A7	10		10	80
A8	10	20	70	

A9	**10**		**10**	**80**
A10	20		40	40
T1	15	50		35
T2	10	45	45	
T3	**25**	**25**	**25**	**25**
T4	25	40	35	
T5	10	60		30
T6	10	60	20	10
T7	20		50	30
T8	10	50	40	
T9	10	60		30

Table 5 shows the performance outcomes at the end of the semester. Note that students were allowed to renegotiate their percentage distribution on projects, with a minimum of 10% for ethics and keeping at least two other projects. As expected, IMM students with rigorous presentation skills training outperformed their CS counterparts in the Ethics Forum. However, overall despite selection that biased media students toward story and CS students toward the toy engine, there was no significant difference in grades between the two cohorts.

Table 5: Performance Outcomes on Deliverables

	% of IMM	% of CS	% of All Students	Avg Grade IMM	Avg Grade CS
Ethics	100%	100%	100%	3.22	2.89
Story	89%	56%	72%	3.75	3.00
Toy	22%	89%	56%	3.00	3.38
Source	67%	56%	61%	2.83	3.00

Finally, it is worthwhile to consider how the individualized deliverables impacted final class standing. Table 6 summarizes the averages for the two cohorts for the four major requirements for the course. With the exception of the journal submissions (which are skewed due to one media student failing to hand in most of these assignments), overall grades for both groups are remarkably close, with a slight advantage for CS students. But the entire group met our expectations (rank 3) within 2/10 of a point, an in fact exceed our expectations on serious project work in the form of collaborative and individual deliverables (e.g. the workshops).

Table 6: Summary of Grading on Individual Deliverables

	Avg. IMM	Avg. CS	Avg. Overall	% of total grade
Total	3.15	3.31	3.23	--
Journal	1.88	3.64	2.81	10%
Workshop	3.49	3.56	3.52	20%
Take Home	2.85	3.07	2.96	20%
Deliverables	3.39	3.24	3.31	50%

5. FACULTY ROLES

A single instructor cannot begin to manage a course sequence such as this. Fortunately our institution has moved toward a transformed curriculum in which team teaching and multi-disciplinary collaboration are encouraged.

In the fall 2005 course, a single instructor of record supervised the projects and gave only two of the formal lectures. Guest instructors presented the other lectures with externally funded stipends. In fall 2006, two faculty shared faculty load of a single section. The guest to instructor of record ratio decreased as reflected in Table 2. Other models of instructor of record to guest lecturer are certainly possible. However, broad representation of contributing disciplines is needed to prevent balkanization.

In spring 2006, three faculty members shared responsibility for two separate sections, thus doubling the total allotment of faculty hours. Our rationale for offering a single section in fall and two in spring was the significant increase in instructor of record participation as seen in Table 2. A problem remaining at our institution is how in the future to adequately compensate without external funding the guest instructors in both their roles as deliverer of instruction and formal evaluator.

Table 2 also highlights the novel set of responsibilities of the instructors. The primary instructors are not over-arching content experts, but rather production managers responsible for accountability during the workshop sessions that comprise 60% of the total contact time. Responsibility for articulating individual learning is a collaborative exercise between instructor and student. Assessment shifts from traditional grading of standard deliverables (including test answers) to analysis of time management, skills development and highly personalized demonstration of content mastery. The payoff for instructors is that this style of grading is far more satisfying.

6. SUMMARY

Analysis of student final exams from AY05-06 as well as journals and final exams from fall 2006 suggest that we are successfully integrating course content across disciplines in a manner that de-balkanizes the disciplines critical to video game development. Student deliverables for the AY05-06 game demonstrate significant cross-disciplinary contributions and consequent skill mastery.

Over 80% of the students in the fall 2006 class successfully answered the reflective questions on specialized topic lectures, related assigned readings to the lectures, as well as to their assigned project work. There is evidence in their writing that they see the complexity and contributions of the various disciplines.

Of more significance are the reflective writings with regard to collaboration and communication. Last years' students demonstrated deep understanding of the critical need for good communication across disciplines, and the value of at least a superficial understanding of disciplines outside their own major.

Game design and development will never be a field exclusively within the domain of computer science. Nor will it become a field entrenched in digital art or interactive storytelling. Our two-year experience in collaborative multidisciplinary teaching suggests that game design is indeed a field for the 21st century that requires truly global, diverse communication skills.

7. ACKNOWLEDGMENTS

Our thanks to our colleagues in the Game Design Project at The College of New Jersey: Kim Pearson, Phil Sanders, Terry Byrne, Miroslav Martinov, JiKai Li, Anita Allyn, and Robert McMahan. Thanks also to the students in AY05-06 who put their hearts into the first run and provided invaluable constructive criticism of our methods. And to AY06-07 team: we've just gotten started. We are also grateful to Microsoft Research, and especially John Nordlinger for the gift that made this work possible.

8. REFERENCES

[1] Constantine, L. L. Work organization: paradigms for project management and organization, Communications of the ACM, Volume 36 Issue 10, October 1993, pp 35-43

[2] Claypool K., and M. Claypool, Teaching software engineering through game design, ACM SIGCSE Bulletin, Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, Volume 37 Issue 3 June 2005, 123-127

[3] Coppit, D. and J. Haddox-Schatz. *Large Team Projects in Software Engineering Courses*. Proceedings of SIGCSE 2005, (St. Louis, Missouri, February 2005), 137-141

[4] El-Nasr, M.S.and B. K. Smith, Games: Learning through game modding Computers in Entertainment (CIE), Volume 4 Issue 1, January 2006, pp 1 -13

[5] Friedman, T. L. The World is Flat: a brief history of the twenty-first century. Farrar, Straus and Giroux, New York, 2005.

[6] Gillard, S., Managing IT projects: communication pitfalls and bridges, Feb. 2005 Journal of Information Science, Vol 31 (1) 37-43

[7] Federation of American Scientists, *Summit on Educational Games, Harnessing the power of video games for learning*, http://www.fas.org/gamesummit/, October 2006.

[8] Wolz, U. and M. S. Pulimood, *An Integrated Approach to Project Management through Classic CS III and Video Game Development*, to appear in the SIGCSE 07, March 7 -10, 2007, Covington, KY