Introductory Game Development Course: A Mix of Programming and Art

Chao Peng
Department of Computer Science
University of Alabama in Huntsville
Huntsville, AL, USA
chao.peng@uah.edu

***Abstract*—Video games have grown to a major industry. Many universities have started new programs concentrating on game development. This paper describes an introductory course for video game development, which is used for initiatives to develop a gaming minor in the undergraduate Computer Science curriculum. This course is designed to teach students all aspects in a game production cycle. We anticipate this course will engage students in computing and bring collaborative opportunities between art and computer science students.**

***Keywords*-game development; interdisciplinary education; computer science course**

I. Introduction

Video games are a successful application of computer technology enjoyed by players of all ages. Video games have expanded beyond being simply an entertainment medium, to a medium that many researchers and educators believe can help people engage with, learn, and retain content that they otherwise might struggle with.

Game development courses are accepted into the academia. Many college-level institutions offer game development bachelor's degree or minor. Game software as an artistic content has brought collaborations between software engineers and artists. Software engineers write low-level codes to solve technical issues. They rarely design games. Artists are visionary of games and promote how a game should play, but they usually have little experience in coding or any sort of programming. Students interested in the game development program are expected to learn both areas and able to become gameplay programmers who make the design into a reality by writing codes directly for gaming experience. Many existing game curriculums contain programming and art courses but does not have a course for students to combine the two areas in their junior year.

Game curriculums can be categorized based on two different types of concentration: (1) design concentration and (2) programming concentration. A design concentrated curriculum exposes students to story-telling, media and game element organization intensively in their early years. General programming courses may be required for freemen but not oriented by gaming. Junior students may select studio courses but they are not required to be fluent with programming languages. They are introduced to software tools and create games in a drag-drop manner. A programming concentrated curriculum exposes students to technical topics, such as object-oriented programming, data structure, theory of computer graphics and artificial intelligence, prior to the assignments of game implementation. Neither approaches represent games as a mix of art and programming. In the early years of study, students should take courses that allow them to use both programming and artistic skills, so that they gain a greater understanding about games as a cross-disciplinary subject.

Novelty. In this paper, we present an introductory game development course for junior college students. This course is a part of Computer Science curriculum and it is also listed in the curriculum of digital art program for art students who are interested in game production. This course intends to allow students to touch on every aspects in a game production cycle so that they can obtain an understanding of art-programming interdisciplinary concept. This paper discusses the challenges we encountered, the decisions we made for course specifications and actual experience.

II. Background

Pioneering instructors created game development courses in late 1990s. Jones [1] presented an upper-division computer science course for game implementationIt was for students to have an environment where they integrated a broad of computing knowledge to create an interactive product. Game development is more than a programming endeavor [2]. It is a mixture of multidisciplinary interactions and the result of team efforts.

Before designing a game, students need to critique a game similar to what they want to design. The methods of game critique were proposed in [3], [4], [5]. Aarseth [6] addressed the purpose of such a critique is "a methodology for the aesthetic study of games". Games are not necessarily to be overly technical. Students should learn why an existing game attracts people to investigate their time to play [7]. Guimaraes et al. [8] and Baytak and Land [9] claimed that students are more engaged in game development courses than traditional programming courses. They found that some students even continued working on the game projects after the conclusion of the course.

To help students fast prototype games, scripts languages have been commonly used in game development

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.152

courses [10], [11]. Daniel Webster College introduced game development to freshman Computer Science majors [12] using scripting languages, with a focus on small, team-based and innovative projects. Spronck et al. [13] used "dynamic scripting" to generate rules for game AI on the fly.

III. Challenges and Solutions

The artistic and programming topics are often offered in separate courses, ranging in focus from game engine programming ([14], [15]) to asset creation [16] to story narrative ([17], [18]). The course we developed intends to engage students to experience the entire workflow of a video game production. In this section, we describe the challenges we had at the time to propose the course and the suggestions to solve those challenges.

A. Parent and Student Concerns

People may think developing games is not a serious learning process. One misconception is that game development is equivalent to game design, which is to narrate a story and draw concept art, and then ask other people (e.g., software engineers) make the games. When talking to prospective students and their parents, we clarify the difference between *development* and *design*. Game development requires students to learn a great deal of programming and many aspects of computing technologies. Students interested in becoming game developers also have other options after completing the game development degree. They will be qualified to work in data visualization, modeling and simulation, or be software engineers in general.

B. Game Engines

Students majored in Computer Science are usually proficient in programming prior to making their first games. This course intends to teach those students to write codes that directly touch on gameplay mechanics. They are required to use scripting languages within a game engine so that their efforts will be spent on creating game behaviors and character actions without diving into the low-level codebase.

Scripting languages support fast prototyping. Scripts are interpreted by the script interpreter component of the game engine for script-to-codebase communications [19]. It is a challenging task to identify the appropriate game engine for our course: some game engines require a basic understanding of programming (e.g., Unreal, Unity), while others do not even ask you to write a single statement (e.g., GameMaker, Stencyl). The question of which game engine to use is difficult to answer because there is no best or right answer. None of them are perfect. They all have strengths as well as problems. Some are fast for development, some are easy to distribute the finished games, some may have performance issues, and some come with developer-friendly interface. We evaluated a few popular game engines shown in Table I.

We decide to adopt Unity 3D since it provides a unified asset pipeline and great tweaking support. Unity 3D can publish games to a variety of platforms. It has a free version and a powerful scripting feature with good and helpful documentations. Scripting in Unity 3D can be done in Javascript, C#, or Boo. Game behaviors can be easily extended by adding new scripts. Adding a script is trivial, and Unity 3D comes with useful examples.

C. Depth of Course Materials

Video games are an example of interactive computer applications [20], which is for players to inhabit. We do not lay out the set of comprehensive aesthetic approaches to develop interactive systems. Students are expected to define simple play rules that sufficiently guide them to identify implementation goals. The prerequisite is a programming language course, which should cover the knowledge of syntaxes, variables and flow of controls.

Art assets are necessary for games. In the class, students obtain hands-on experience of asset creation but not really go into depth. Students are allowed to borrow art assets from publicly available resources and use them in their games under end-user notices.

Students are required to develop a design document for each of game development assignments . The design document should cover game stories, level designs and gameplay mechanics. They re also required to develop an art list clarifying the number of characters, types of animations and environmental game objects needed for their games. Students are also challenged and conduced to solve the problems related to graphics, AI, user interfaces, character controls and gaming physics. Techniques such as 3D transformation, path-finding algorithms, geometry rendering, etc. are also exposed to the students.

IV. Course Design

The course description in the syllabus is shown as the follows:

" In the process of game production, gameplay programmers create extensions for game applications that directly touching on gaming experience. They support game design needs. This includes game logic, character behaviors, motion systems, camera logic, collision systems, rendering, particle effects, etc. Many game studios rely upon scripting languages to enable gameplay programmers to create application extensions and build game prototypes. In this course, students will acquire theoretical as well as applied aspects of extensible application architectures. Existing and emerging scripting languages are discussed extensively. Students will explore and utilize current applications and must create extensions to these applications by programming in scripting languages."

The experience of being as part of a team is important to the students, not only working with other programmers

Table I
Evaluations of game development tools.

Engine Name	Game Type	Development Environment	Programming Required?	Features
CraftStudio	2D and 3D	Windows, Mac and Linux	No	In a collaborative way, good for online game development
Construct 2	2D	Windows	No	Drag-and-drop actions, exporting in HTML5
Game Maker	2D	Windows	No	With drag-and-drop actions, OK performance
Stencyl	2D	Winodws, Mac and Linux	No	flash-based nature, no built-in gamepad support
Torque	2D and 3D	Windows	Yes	TorqueScript, a big group of users, steep learning curve
Unity	2D and 3D	Windows and Mac	Yes	Multiple scripting Languages, graphically not as good as Unreal
Unreal	2D and 3D	Windows and Mac	Yes	Not well supporting mobile developers, very good for 3D, not so good for 2D
XNA	2D and 3D	Windows	Yes	No interface, used to be popular, but now killed off by Microsoft.

but also knowledgable to nontechnical subjects. Students are given an option to form a team with up to two personnel. For the students working in a team, their projects must contain a document explaining the contributions of personnel. The grade of a team member is upon the overall quality of the project and the amount of individual contributions.

A. Textbook

Although online tutorials are available for students to learn game development tools, textbooks are still the workhorse of education. As beginners with passion, students are ambitious, and often, they bite off more than they can chew. They might already play triple-A video games and are eager to develop a game with similar quality and complexity. But they should understand that they need to start with the development of simple games. Thus, we would like to choose a textbook emphasizing on small and simple gaming ideas. One difficulty is that many game development books give intensive theories and algorithms that do not engage students to practical workflows. To serve our intro-level course, the book should discuss a wide variety of concepts and demonstrate crucial functioning elements that students can adopt in their implementation. The textbook we chose is *"Game development with Unity"* [21], published in 2012. This book guides students to work in Unity well and emphasizes on the object-oriented design with example codes. We also employed a second textbook entitled *"Introduction to Game Development"* [22]. It covers the overall subject and explains the whole of the game industry and development process from a high viewpoint.

B. Course Schedule, Assignments and Grading

Lectures are prepared to guide students through the phases of design and implementation in a 16-week semester. Table II shows the details of the course schedule.

Four projects are assigned to students. The first project requires students to design and implement a 2D game. Students first critique an existing 2D game (any platform) focusing on the uniqueness of the game that makes the game fun to play. Students then develop a design document to illustrate their own games. The second and third projects require students to create a 3D model, rig it (bind it with a skeleton) and animate it. Students learn the process of 3D asset creation and are able to use it in the game. The last project is creating a 3D game. Students critique a 3D game and develop the design document prior to the implementation.

A design document is a word file with texts, images, concept arts or other medium that help to narrate the game design. The exact formatting is up to students, though it must

Table II
Course Schedule

#	Topics
1	Course Introduction: syllabus, textbook, grading Video Game History: 1950s-now
2	Game Critique: what features of a game do attract you? Design Elements: story, goal, play challenges, prototypes Game Engine: focusing on scripting
3	Sprite: image and color system Coordinate System: world, object and screen systems 2D Transformation: translation, 1-axis rotation and scale Text: how to draw text on screen
4	2D Physics: gravity, force Sprite Sheet: 2D Animation and motion graph
5	Interaction: collision detection and events User Input: mouse, keyboard and joystick
6	Scene Management: multi-scenes and camera panning Game Publishing: gaming platform
7	3D Fundamentals: 3D coordinate systems and assets GUI Development: inventory implementation
8	3D Modeling: vertices, triangles and normals
9	Texture: UV coordinates Material: phong, blinn and reflection shaders Lighting: light sources and self-glowing
10	Skeleton: bone hierarchy, binding pose and skin weights 3D animation: key framing and motion data format
11	3D Math: vector, matrix and ray-surface intersection 3D Character Control with scripts
12	First Person Camera: mouse look, launcher and scope Third Person Camera: auto Camera
13	Sound effects: audio signal and blending
14	Particle Effects: water, fire and cloud
15	AI: random, A-star algorithm and AI for NPC
16	Ragdoll Physics: death motion Destructible and Trigger Zone: collapsing walls

contain a *title* and *sections* with headings. As an example, the following shows the requirement of 2D game design:

- **Concept statements:** This should be the first section of the document describing the core vision of the game. It should include the game story, what the player could do and could not do.
- **Design goals:** This should include an explanation of what experience the player could have through the playing. For example, students could write in the design document like, "*I would want the player to experience how gravity works*" or "*I would want the player feel slipping sensation while the character is walking on the ice.*", rather than just saying "*I would like make the game very interactive*? or "*I would make the game fun and challenging.*". The document should also address the target audience of the game (e.g., kids, teenagers, dog lovers, hunting fans, etc.). The length of the game (time) should also be included.
- **Gameplay and decision-making:** This should include the key gameplay features. In the design document, students should make connections between the gameplay of their game design and that in the game they critiqued. The types of character actions should be specified in the document. Students should also describe the victory and loss conditions. An gameplay strategy is required to explain how to promote the player to make decisions. Students should detail how each decision advances the player towards either the victory or the loss.
- **Game environments:** Students should illustrate the look of virtual world of the game. They could itemize any game objects or sprite assets that they would need to construct the world. Students should describe the level design with images as appropriate.
- **Presentation:** Students are required to prepare a 10-minute presentation. Video clips showing the physical prototype of the game are not mandatory but are recommended to be included in the presentation to help us better understand your game!

The requirement of 2D game implementation is shown as the following:

- **Player:** Students should create at least one player character that contains at least two types of sprite animations: walking and jumping. The character should be controlled by keyboard or joystick. Students could earn extra credits if the character could fire bullets or similar actions as attacking.
- **Gaming environment and Camera Setting:** Students should create a virtual world that allows the player to explore. The world should have foreground objects (those colliding/interacting with the character) and background objects (those the character can move through freely such as trees, walls, clouds, etc), and they should be appropriately placed in the world to serve the gameplay. The game should also have a camera, either moving to follow the player, or fixed but allowing the player to move between multiple scenes/locations.
- **Items/Game Objects:** Students should create at least 2 items that the player could interact with (e.g., picking/dropping items, moving items, etc.)
- **NPCs:** Students should create at least one NPC (Non-Player Character). The player should be able to interact with the NPC (e.g., talking to, accepting quests from, turning in quests, learning information, etc.) Animations for the NPC are not required.
- **Help Menu:** In the game, an on-screen instructional tutorial is required. It should teach users the controls and how to win or lose the game. Imagining friends would play the game, they should be able to teach themselves to play the game with this help menu.
- **Game Demo:** Students should prepare 10-minute live demonstration in class on the demo day. Students are expected to play their games during the demonstration.

Table III
Grading Assessments

Attendance	3%
Quizzes (2)	6% (3% each)
Midterm Exam	12%
Project 1 (2D game)	16% (6% critique+design, 10% implementation)
Project 2 (3D Modeling)	9%
Project 3 (3D Animation)	9%
Project 4 (3D game)	23% (6% critique+design, 17% implementation)
Final Exam	22%

We also create one midterm exam, one final exam and 2 quizzes. Those exams and quizzes alllow us to have evaluations on students' learning, particularly on the knowledge that are not easy to be exercised through projects. Table III shows the assessment rubrics.

V. Evaluations and Results

The course was offered at undergraduate junior level in Computer Science curriculum. 36 students were enrolled, including 33 males and 3 females. All of them have experience in programming and have already taken a data structure course or equivalent prior to registering this course. 3 students withdrew early due to the complications of the projects. The remaining students successfully completed the course. 18 students formed 2-person teams, so 9 games for each game development project were done by teams. 26 students completed a specialized survey at the end of the course, which is a 78.8% completion rate.

The specialized survey was designed to collect the information including student's previous programming background, skill development, gameplay interests and career

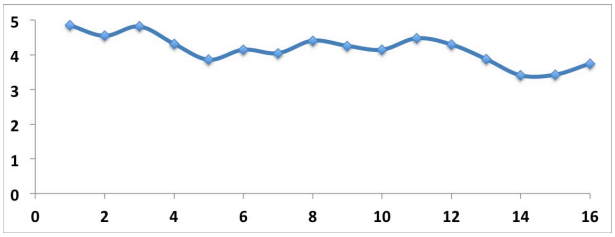

Figure 1. Weekly changes of students' enthusiasm on game development during the 16-week semester. The enthusiasm is measured with a value between 1 (not interested at all) and 5 (very interested).

expectations. The survey was administered to students at the end of the 16-week semester, along with university standard evaluations. The survey shows students were already comfortable to write complex computer programs prior to registering this course. Some students commented that the reason they registered this course is because they want to develop their own games to be similar to what they enjoyed playing, but they could not achieve that due to the time restriction and lack of experience. Most of students expressed that their interests in game development were increased after this course.

Another survey was administered weekly online to evaluate the changes of students' enthusiasm during the time of developing games. The result is shown in Fig. 1. Their enthusiasm was influenced by the type of game they worked on and the time and efforts they put in. Students were at low points of enthusiasm during week 4-6 and week 13-16, because they were under the pressure to finish 2D and 3D game projects. Some feedbacks indicate students preferred working on 3D games rather than 2D games. Some students expressed that they understood that knowledge of 3D math is required to develop video games, but they did not realize a deep understanding of 3D vectors and transformation operations is so important for 3D games until Project 4. Students also mentioned that they spent much time to debug camera and character action codes, and felt 3D animation is more difficult to be integrated into the game than other gameplay features. Overall, students maintained a fairly high passion during the semester, though they experienced the complexities and challenges to write gameplay codes. Some students were even overworked on implementations of the two game projects in order to meet their design goals.

Although students knew the importance of design document, they were not really active on it and did not spent much time to write the document. Some students held the opinion that writing words about a non-existing game feels maddening. They would prefer putting fingers to keyboard and refining the gameplay mechanics while they were typing actual codes. As instructors, we do not want students to do such unusual developing activities. Most of students changed the attitudes and delivered a well-prepared design document for the 3D game project. In the survey, one student commented that, *"I had to get something on paper otherwise I was developing in the dark."* For developing a design document, we suggested students craft the design using not only words but also images. We told them that words are good but usually not enough to document a game. We encouraged them to draw images themselves to illustrate the ideas. One studnet commented in the survey that, *"I am not good at drawing, but what I drew helped me figure out my level design."*. Fig. 2 shows some of 2D and 3D games developed by the students of this course.

For Project 2 and 3, students were required to use Autodesk Maya to model a 3D character they designed and then animate it. About a third students used their own characters in Project 4. Project 2 and 3 did not give much challenges to the students as the two game projects. Students commented that modeling and animating a 3D character made them have a good understanding of the process of 3D asset creation.They learnt that 3D modeling is a time-consuming process.

VI. Conclusion

We presented the design and implementation of an introductory game development course that has shown a great integration of many concepts of art and computing disciplines. The course used a project-oriented pedagogy. Students were enthusiastic to the topics and lectures and enjoyed working on the projects. This course was an experimental course at the moment it was offered for the first time as a special topic. It is going to appear as a regular course in the curriculum of Computer Science and be added to the curriculum of the game production minor in Art department. We expect this course to feature collaborative work with art students in next one or two semesters. This course attracts non-CS majors to become interested in computing as well as working in interdisciplinary environments. In the future, we would like to involve industry professionals to provide students with any sort of concrete supports, working experience and necessary skills.

References

[1] R. M. Jones, "Design and implementation of computer games: A capstone course for undergraduate computer science education," in *Proceedings of the Thirty-first SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '00. New York, NY, USA: ACM, 2000, pp. 260–264.

[2] N. R. Sturtevant, H. J. Hoover, J. Schaeffer, S. Gouglas, M. H. Bowling, F. Southey, M. Bouchard, and G. Zabaneh, "Multidisciplinary students and instructors: A second-year games course," *SIGCSE Bull.*, vol. 40, no. 1, pp. 383–387, Mar. 2008.

[3] L. Konzack, "Computer game criticism: A method for computer game analysis." in *CGDC Conf.* Tampere University Press, 2002, pp. 89–100.

Figure 2. Games developed by the students in this course. The first row are 2D games; the second row are 3D games.

[4] D. Thomas, J. P. Zagal, M. Robertson, I. Bogost, and W. Huber, "You played that? game studies meets game criticism," in *DIGRA Processing*, Tokyo, Japan, 2009.

[5] A. Waern, "Game analysis as a signature pedagogy of game studies." in *FDG*, 2013, pp. 275–282.

[6] E. Aarseth, "Playing research: Methodological approaches to game analysis," in *Proceedings of the Digital Arts and Culture Conference*, 2003, pp. 28–29.

[7] K. Peppler, M. Warschauer, and A. Diazgranados, "Game critics: Exploring the role of critique in game-design literacies," *E-learning and Digital Media*, vol. 7, no. 1, pp. 35–48, 2010.

[8] M. Guimaraes and M. Murray, "An exploratory overview of teaching computer game development," *J. Comput. Sci. Coll.*, vol. 24, no. 1, pp. 144–149, Oct. 2008.

[9] A. Baytak and S. M. Land, "A case study of educational game design¡ i¿ by¡/i¿ kids and¡ i¿ for¡/i¿ kids," *Procedia-Social and Behavioral Sciences*, vol. 2, no. 2, pp. 5242–5246, 2010.

[10] P. Sweetser and J. Wiles, "Scripting versus emergence : issues for game developers and players in game environment design," *International Journal of Intelligent Games and Simulations*, vol. 4, no. 1, pp. 1–9, 2005.

[11] M. Cutumisu, C. Onuczko, M. McNaughton, T. Roy, J. Schaeffer, A. Schumacher, J. Siegel, D. Szafron, K. Waugh, M. Carbonaro, H. Duff, and S. Gillis, "Scriptease: A generative/adaptive programming paradigm for game scripting," *Science of Computer Programming*, vol. 67, no. 1, pp. 32 – 58, 2007, special Issue on Aspects of Game Programming.

[12] T. Goulding and R. DiTrolio, "Complex game development by freshman computer science majors," *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 92–99, 2007.

[13] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game ai with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.

[14] R. Coleman, M. Krembs, A. Labouseur, and J. Weir, "Game design & programming concentration within the computer science curriculum," *SIGCSE Bull.*, vol. 37, no. 1, pp. 545–550, Feb. 2005. [Online]. Available: http://doi.acm.org/10.1145/1047124.1047514

[15] E. Sweedyk, M. deLaet, M. C. Slattery, and J. Kuffner, "Computer games and cs education: Why and how," *SIGCSE Bull.*, vol. 37, no. 1, pp. 256–257, Feb. 2005.

[16] A. Ward, *Game character development with maya*. New Riders, 2005.

[17] H. Jenkins, "Game design as narrative architecture," *Computer*, vol. 44, p. s3, 2004.

[18] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining "gamification"," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, ser. MindTrek '11. New York, NY, USA: ACM, 2011, pp. 9–15.

[19] A. Calleja and G. Pace, "Scripting game ai: An alternative approach using embedded languages," *Proceedings WICT*, 2010.

[20] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*. The MIT Press, 2003.

[21] M. Menard, *Game development with Unity*. Cengage Learning, 2012.

[22] S. Rabin, *Introduction to game development*. Cengage Learning, 2010, vol. 2.