A Strategy of Integrating Game Programming into Computer Science Curriculum

Yen-Hung Hu and Jean Muhammad
Department of Computer Science, Hampton University, Hampton, Virginia 23668, USA

Abstract—This paper describes a strategy to integrate game programming into the computer science curriculum. It is a well-known challenge that nationally computer science/computer engineering enrollment is at its lowest in at least a decade. Many approaches have been implemented during past few years and game programming is one of those showing positive feedbacks. In this paper utilizing the game programming approach to increase student’s interest and retention, we have provided the platform and game components adopted by the Department of Computer Science at Hampton University. Outcomes and performance studies from actual class trials are also included.

Keywords: Computer Science, Game Programming.

1. INTRODUCTION

National enrollment of Computer Science/Computer Engineering programs have decreased continuously for many years [1]. To improve falling undergraduate enrollment in computing, several approaches have been adopted. Crenshaw et al. [2] proposed a case study, which is based on new programs with improved student-community interactions. Their results showed that student retention could be improved by the following recommendations: (1) Increase community interactions; (2) Increase early research opportunities; (3) Create multiple and diverse mentoring programs; and (4) Harness existing social networks to advertise mentoring programs.

Rafieymehr [3], on the other hand, proposed another approach to attract more high schools graduates to enroll in computer science and the relative majors. This approach, building on a combination of strategies to spark an interest in computing in middle and high schools, could be categorized into two efforts: (1) Provide enough and right level programming courses for middle and high school students; (2) Create attractive and inspired hands-on projects for middle and high school students.

Other approaches were more focused on introducing new courses or revamping existing courses. Herath et al. [4] implemented an interdisciplinary accounting forensics course in the existing curriculum to improve computer science enrollment. Whitehead [5] at the University of California, Santa Cruz offered a large enrollment game design course to attract non-major students. Meanwhile, several other teams also revealed their ways of integrating game components into their curriculum to draw students back to the computer science major [6] [7] [8] [9]. In general they found that students faced challenges of learning and understanding fundamental courses. For instance, the difficulties of mastering early programming courses may tend to scare many students away [6]. Research results in [9] showed that the strategy of adopting computer games have been meaningful and feasible in student recruitment and retention.

“... roughly 1/3 of our first year class has been attracted by the game courses ... K. Becker et al. [9]”

The Department of Computer Science at Hampton University recognizes the serious issues concerning the decrease of enrollment and retention and have proposed several courses of actions in their attempt to circumvent this trend. The department has increased connections and interactions with local and regional high school students, gained more external funds to retain major students, and revamped its existing curriculum to attract more high school graduates and non-major students. We as university educators in the field of computer science are all battling this challenge and must be willing to conquer it no matter the cost.

In this paper, we propose an approach of integrating game components into the computer science curriculum. To optimize our limited resources, we adopted Microsoft XNA [10] as the game platform. And then built exercises from basic to advanced topics upon this platform. XNA game studio has enabled developers to easily create video games for Microsoft Window systems, Xbox 3601, and Zune digital media player2 by using optimized cross-platform gaming libraries based on the .Net Framework3.

The objectives of this project are to intrigue and retain computer science major students. Therefore, a visualized and interactive interface that could graphically display users’ customizations would be very helpful. In our implementation, we chose well designed topics that would enhance the student’s understanding of high-level programming concepts as well as encouraging them to tackle projects of increasing levels of difficulty.

The remainder of the paper is organized as follows. In Section 2 we describe game components and our course implementation in detail. Section 3 discusses assessment and our performance study. Section 4 concludes the paper.

1 Xbox 360 is a video game console produced by Microsoft.
2 Zune digital media player is a media play produced by Microsoft.
2. GAME COMPONENTS AND IMPLEMENTATION

In this section, we describe game components adopted by this project and also explain how we implement such components into our curriculum.

To intrigue students, we have selected game components that could eventually reflect the fundamental importance of basic programming skills as well as bringing attentions and challenges to the more experienced student. To examine our approach, the Department of Computer Science at Hampton University offered its first gaming programming course in the fall semester of 2008.

2.1 Game Components

We have selected several game concepts from Chad Carter’s, “Microsoft XNA Unleashed - Graphics and Game Programming for Xbox 360 and Windows” [11], as the outline of this course. Such concepts include: Installation and Introduction of XNA game studio, C# programming, Introduction, XNA basics, 2D in XNA, High Level Shader Language (HLSL), Physics and artificial intelligence (AI), and 3D in XNA.

![Game Components Diagram]

Figure 1: Game Components

Figure 1 shows the selected game components, which the gaming course builds upon from the bottom (i.e., Installation and Introduction of XNA Game Studio) to the top (i.e., 3D in XNA).

In the beginning, students will learn how to install XNA into their systems. This installation consists of 3 processes: (1) installing Visual C# express, (2) installing the DirectX Runtime, and (3) installing XNA Game Studio Express. After the installation process, students have to practice and experience C#, which is the main language of XNA. Once the students have acquired satisfactory skills in C#, intensive exercises of XNA basics are imposed. 2D exercises are then provided which have been designed to bring out the student’s interests and produce further challenges. Another important game component to be included is HLSL. HLSL is a standard high-level language created by Microsoft to allow developers to talk at a high level to the graphics card therefore not needing to use assembly code to access the hardware. However, on the Xbox 360 developers could still use assembler directly inside of the HLSL syntax. In addition this course also covers properties of physics and artificial intelligence. Physics properties including kinematics, force, and collisions are used to reflect and mimic movements of objects. AI has been introduced into the gaming industry for quite some time. Since the field is broad, we only touch a very few AI algorithms. At last, 3D game concept is introduced, making the game characters come to life.

2.2 Implementation

We plan to bind our course implementation to course objectives seamlessly. We designed topics for each game component (see Table 1). We then design course exercises (see Table 2), which are able to support every game component (see Table 3). From our perspective, this game programming course should be hands-on project oriented and the exercises should be able to help students to understand course components as well as produce an interest for students to practice further as intensive game professionals.

We have depicted a framework to explain our implementation. The details will be unveiled in the following sections.

<table>
<thead>
<tr>
<th>Components</th>
<th>Topics</th>
</tr>
</thead>
</table>
| Installation and introduction of XNA game studio | • XNA and the Xbox 360.  
• Performance considerations |
| C# programming | • Introduction of C#  
• Implementation of C# |
| XNA basics | • Creating 3D objects  
• Input devices and Cameras |
| 2D in XNA | • 2D basics  
• 2D effects  
• Creating a 2D game |
| High Level Shader Language | • HLSL basics  
• Advanced HLSL |
| Physics and artificial intelligence | • Physics basics  
• AI algorithms |
| 3D in XNA | • 3D basics  
• 3D effects  
• Creating a 3D game |

Table 1: Game Components and Topics

---

4 C# is a programming language developed by Microsoft.
2.2.1 Game Component and Topics

Table 1 shows the design of the game components and their associated topics.

To introduce XNA to students in detail, we have integrated the topics of XNA installation, XNA features, and performance consideration into the first game component (i.e., Installation

<table>
<thead>
<tr>
<th>Exercises</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW-1</td>
<td>Familiar with C# GUI (Calculator, Alarm Clock, Radio, Calendar and Appointments)</td>
</tr>
<tr>
<td>HW-2</td>
<td>Familiar with C# code (GUI and Functionality of Calculator, Alarm Clock, Radio, Calendar and Appointments)</td>
</tr>
<tr>
<td>HW-3</td>
<td>Familiar with XNA and modification of existing games (3D space-war game)</td>
</tr>
<tr>
<td>HW-4</td>
<td>Familiar with XNA interfaces and 2D facilities (2D object with random movement)</td>
</tr>
<tr>
<td>HW-5</td>
<td>Familiar with XNA 3D movement (translation, scaling, rotation, and projection)</td>
</tr>
<tr>
<td>HW-6</td>
<td>2D/3D game development (2D/3D aspects, keyboard/mouse/game pad controller, players could be upgraded while reach the required points, background or texture could change according to different game scenes, music/sound/voice associate with different plays/scenes)</td>
</tr>
</tbody>
</table>

Table 2: Game Exercises and Objectives

and Introduction of XNA game studio). Both Windows and Xbox 360 systems are imposed. In the second game component (i.e., C# programming), the topics of introduction of C#, and implementation of C# are adopted to give students deeper and broader programming skills. These skills will enable students to handle more advanced challenges in game programming.

XNA basics is the third game component that covers the topics of creating 3D objectives and input devices and cameras.

The next game component is 2D in XNA, which is the first component in this course that actually involves implementation of complicated objects designed by students. To achieve the goal of creating this 2D game, the topics of 2D basics, 2D effects, and creating a 2D game are added.

One very important aspect of game programming is to utilize the graphics card. This involves steps of accessing the graphics card. To simplify these steps, XNA introduces the HLSL, which provides a syntax that enable developers to program their needs into the Graphics Processing Unit (GPU) and display on the screen. A set of customized vertex data and parameters are passing through the GPU, in which such data and parameters are calculated and reformed by the vertex shader and the pixel shader. At this point the customized scene is displayed on the screen. To cover this game component, the topics of HLSL basics and advanced HLSL are implemented.

In the next game component, the topics of physics properties of the objects and artificial intelligence of game strategies are embedded. These two concepts give the game an opportunity to mimic and reflect real world experiences. For instance, through adopting the information of a real bug’s behaviors, a virtual bug could demonstrate activities simulating the real one. Although not touched in depth by this course, this component still includes the topics of basics of physics and AI algorithms.

The Last component designed for this course is 3D in XNA. This includes the topics needed to understand 3D basics, effects, and how to create a 3D game.

2.2.2 Game Exercises and Objectives

To enhance the student’s understanding of course materials, we implant 6 exercises. Objectives of these exercises are listed in Table 2. In this section, we will describe each exercise and its objectives in detail. For each exercise, we also select some outcomes from students in this course.

Exercise HW-1 (see Figure 2 for one of the outcomes) plans to introduce C# GUI1 to students. In this exercise, four C# graphics (i.e., calculator, alarm clock, radio, and calendar and appointments2) are included.

Exercise HW-2 (see Figure 2 for one of the outcomes) is an extension of exercise HW-1. Every function in each of the various graphics must be coded properly. Since widgets have been added into games by being one of the major roles or accessories of such games, these two exercises will be very beneficial to the young developer. Through these exercises, students will be able to design and implement C# graphics.

Figure 2: C# Graphics Design and Implementation

Exercise HW-3 (see Figure 3 for one of the outcomes) will provide the students with the ability to touch the structure and

---

1 GUI: Graphical User Interface.
2 These four exercises are adopted from Chapter 2 of [12].
3 This exercise adopts the Spacewar game from the Windows Starter Kit.
have a sense of XNA games. In this exercise, students have to install the Spacewar game from the Windows Starter Kit\(^8\) and modify at least one feature of this game.

**Figure 3:** Modification of Space War. Original (left) vs. Modified (right)

Exercise HW-4 (see Figure 4\(^9\) for one of the outcomes) and HW-5 (see Figure 5\(^10\) for one of the outcomes) require students to implement 2D and 3D objects with movements, respectively. Students will learn various movement concepts and HLSL coding through these two exercises.

**Figure 4:** 2D Object with Movements

**Figure 5:** 3D Object with Movements

Exercise HW-6 (see Figure 6\(^11\) for one of the outcomes) is the final project used to demonstrate an ability of combining all knowledge learned from this course. Students have to make their own 2D or 3D games with certain requirements such as: (1) this game should be controlled by either keyboard, or mouse, or gamepad, (2) background or texture of this game should be able to change when players reach other game levels or scenes, (3) music or sound effect of this game should be able to change when players reach other game levels or scenes. This exercise will provide students the opportunity and challenge of implementing their first semi-professional game with many features, which are very difficult to be implemented by a traditional programming language (e.g., JAVA, C++, etc.) without a supportive game platform.

**Figure 6:** A 2D Game with the Capabilities of 1) Keyboard Control, 2) Background Change, and 3) Music/Sound Change.

### 2.2.3 Game Component and Exercises

In previous sections, we have mentioned the game components and the topics of this course as well as described the game exercises and the objectives in detail. In this section, we would like to demonstrate that the exercises prepared for this course will actually provide a sound implementation of the game components (see Table 3).

As mentioned earlier, exercise HW-3 requires students to install the Spacewar game from the Windows Starter Kit, and then modify at least one feature. This means, students have to explore the architecture of the specified game to find codes that could actually affect the outcome when they are modified. It is a time consuming job. During this procedure, students would acquire sufficient knowledge to support the first game component, Installation and Introduction of XNA game studio and XNA basics (i.e., Component #1).

Exercise HW-1 and HW-2 are focused on the practice of C# GUIs and implementations. Therefore, they are naturally designed to support the second game component, C# programming (i.e., Component #2).

Exercise HW-4 and HW-5 are the preliminary practices for engaging students in actual game programming. They cover

---

\(^8\) This Window Starter Kit is included in Microsoft XNA Game Studio Express 1.0.

\(^9\) The 2D image in Figure 4 is adopted from [11].

\(^10\) The 3D object in Figure 5 is adopted from the Spacewar game from the Windows Starter Kit.

\(^11\) This is an outcome from the final project of the game programming course. It is conducted by the following students: Mr. Julian A. Strothers (team leader), Mr. Timothy A. Brown, Ms. Candace N. Kizzie, and Ms. Ashley R. Thornton. All of them are students of the department of Computer Science at Hampton University.
the game components of 2D in XNA, HLSL, and 3D in XNA (i.e., Component #4, #5, and #7).

Exercise HW-6 is a semi-professional project that reflects the students' comprehensive understanding of this course. It covers the game components of 2D in XNA, HLSL, Physics and artificial intelligence, and 3D in XNA (i.e., Component #4, #5, #6, and #7).

<table>
<thead>
<tr>
<th>Components</th>
<th>Exercises</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation and introduction of XNA game studio</td>
<td>HW-3</td>
</tr>
<tr>
<td>C# programming</td>
<td>HW-1, HW-2</td>
</tr>
<tr>
<td>XNA basics</td>
<td>HW-3</td>
</tr>
<tr>
<td>2D in XNA</td>
<td>HW-4, HW-6</td>
</tr>
<tr>
<td>High Level Shader Language</td>
<td>HW-4, HW-5</td>
</tr>
<tr>
<td>Physics and artificial intelligence</td>
<td>HW-6</td>
</tr>
<tr>
<td>3D in XNA</td>
<td>HW-5, HW-6</td>
</tr>
</tbody>
</table>

Table 3: Game Components and Exercises

3. ASSESSMENT AND PERFORMANCE STUDY

In this course, we conducted a survey in the fall semester of year 2008, which covers 5 out of all 7 students. The average scored results for each category is as follows: (1) the score of the course objectives is 8.1 out of 10; (2) the score of the course materials is 8.9 out of 10; (3) the score of the homework is 8.2 out of 10; (4) the score of the quizzes and exams is 7.7 out of 10; and (5) the score of the instructor’s evaluation is 8.5 out of 10. Through this study, we realize that we still need additional revisions and updates in our efforts to continuously improve this course. This result will be adopted by our future game programming courses.

4. CONCLUSION

In this paper, we describe a strategy of integrating game programming into Computer Science curriculum. We explain the implementation plan including (1) game components and topics, (2) game exercises and objectives, and (3) game components and exercises. We also discuss the implementation, outcome assessment and performance study of the game programming course offered by the Department of Computer Science at Hampton University during the fall semester of year 2008. Since this course is observed for only one semester, it is still too early for us to claim any enhancement of retaining Computer Science major students or intriguing non-major students. However, our study shows that the some of our objectives have been achieved. Students were excited and stimulated by the course.

In the future, we plan to implement two separate sections of game programming. One is in CS1 level for all students without any prerequisite. The other is in CS3 level for those students who have at least one year Object-Oriented
Programming (OOP) experience and have passed the CS2 level data structure and algorithms course. The purpose of the CS1 level gaming programming course is to attract non-major students. However, the purpose for the CS2 level programming course is to retain major students. We also plan to revamp our existing course assessments to accommodate the capability of evaluating course materials, instructor performance, and achievement of retaining major students and attracting non-major students.

**ACKNOWLEDGEMENT**

We would like thank to Mr. Timothy A. Brown, Ms. Amanda D. Harvey, Ms. Candace N. Kizzie, Mr. Julian D. Ross, Mr. Gheric Speignger, Mr. Julian A. Strothers, and Ms. Ashley R. Thornton for their contributions. All of them are students of the department of Computer Science at Hampton University. Our thanks also go to the anonymous reviews. Their comments were very helpful and greatly improved the quality of this paper.

**REFERENCES**