

XQUEST used in Software Architecture Education

Bian Wu

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Bian@idi.ntnu.no

Jan-Erik Strøm

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Janerist@stud.ntnu.no

Alf Inge Wang

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Alfw@idi.ntnu.no

Trond Blomholm Kvamme

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
trondblo@stud.ntnu.no

Abstract— This paper describes motivation and application of a Microsoft XNA extended library- XQUEST (XNA Quick & Easy Starter Template) in a software architecture course, and further presents the evaluation of how well the XQUEST was to use in a software architecture course. XQUEST was designed and implemented to save students' time in project development with flexible components. The evaluation was based on research methods and questionnaires from the students in the software architecture course. Finally, the questionnaires results were analyzed in three aspects: suitability, usefulness and usability. In many aspects, the results show that XQUEST enhances XNA in suitability as a teaching aid in software engineering learning, and that it can be a useful and helpful extension to understand XNA. The results also show that XQUEST is easy to use and save students time in development, thus giving students more time to focus on the practice of course theory.

Keywords- XNA; Software architecture; Software engineering education; Evaluation

I. INTRODUCTION

Research on games concept and game development used in higher education has been done before, e.g. [1, 2, 3], but we believe there is an untapped potential that needs to be explored due to development of new technologies. After some commercial SDKs have come out in recent years, such as XNA [6], iPhone SDK [19] or Android [20], we had considered how to use new technology and devices in the higher education to enrich the learning environment. This paper will focus on how to use a game development environment to teach software architecture or related courses. The motivation is to bring the same enthusiasm from playing games to learn to courses' contents through game development. The specific features of a game SDK can give new insights and provide support for the educational process used in the teaching directly, providing an open platform for students during teaching. The games and game development frameworks can be integrated mainly in three ways with a university course. First, they can be used to replace traditional exercises. This approach would motivate students to put extra effort into exercises and give teachers

and/or teaching assistants an opportunity to monitor how the students work with the exercises in real-time [22, 23]. Second, they can be integrated in lectures to improve the participation and motivation of students [24, 25]. The goal of proposed game concept is to prompt students and increase students' attendance in lectures. Third, the students can use them in projects to develop software to understand the courses' content related to software engineering or computer science [21, 26, 32].

This paper focuses on the latter, where game development is used in student projects to learn software engineering skills. Concretely, we focus on one specific SDK--XNA and contribution is to extend XNA game libraries to improve it more suitable for higher education in two ways: shorten students' development time, and improve the content and structure of XNA to fit certain course. Further, an evaluation and analysis of extension of XNA game libraries' application is presented.

The rest of the paper is organized as follows. Section 2 is an introduction of XNA and its application in software architecture course. Section 3 describes the XQUEST design and its structure. Section 4 describes an assessment of XQUEST application, Section 5 describes related work and Section 6 concludes the paper.

II. XNA USED IN HIGHER EDUCATION

This section is a detailed discussion of XNA structure and its application in a software engineering course at Norwegian University of Science and Technology (NTNU).

A. XNA Structure

XNA is a game development platform developed by Microsoft, which includes a programming framework and a set of tools to offer a complete game development package [4]. Based on the .NET platform, XNA offers game development for the PC, the Xbox 360, and more recently the Zune [5] media player. Further, XNA uses the C# programming language. XNA mainly targets students, hobbyists, and

independent game developers. XNA is free to use, but to deploy games on the Xbox 360, a subscription to the XNA Creators Club [6] is required. XNA was motivated by an earlier attempt at bringing the DirectX C++ multimedia API [7] over to the .NET platform, called Managed DirectX [8]. It was essentially a 1:1 mapping of the DirectX API onto .NET. XNA took the idea one step further and provides a complete game development solution, not just the programming API. First released version 1.0 is in December 2006, the latest version of XNA is 3.0, released in October 2008[6].

The overview architecture of XNA consisting of four layers is shown in Fig. 1.

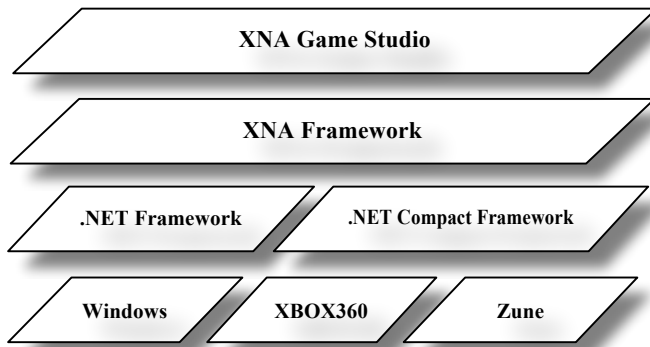


Figure 1. The Deployment View of XNA

B. XNA used in Software Architecture Course

The software architecture course is a post-graduate course offered to computer science and software engineering students at NTNU. The course is taught every spring based on the book *Software Architecture in Practice* [9], and its workload is 25% of one semester. In the software architecture course, 30% of the grade is based on an evaluation of a software architecture project all students have to do. The rest 70% is given from a written examination. The goal of the project is for the students to apply the methods and theory in the course to design software architecture and to implement a system based on XNA framework according to the architecture. The project consists of the following phases [32]:

- 1) *COTS (Commercial Off-The-Shelf) exercise*: Learn the technology to be used through developing a simple application.
- 2) *Design pattern*: Learn how to use and apply design pattern by making changes in an existing system.
- 3) *Requirements and architecture*: List functional and quality requirements and design the software architecture for the application (a game).
- 4) *Architecture evaluation*: Use the ATAM (Architecture Tradeoff Analysis Method) evaluation method to evaluate the software architecture of project in regards to the quality requirements.
- 5) *Implementation*: Do a detailed design and implement the application based on the created architecture and on the changes from the evaluation.
- 6) *Project evaluation*: Evaluate the project as a whole using a PMA (Post-Mortem Analysis) method [10].

The course staff issued the tasks to make a functioning game using XNA, based on students' own defined game concept. However, the game had to be designed according to a specified and designed software architecture. Further, the students had to develop an architecture where they had to focus on one particular quality attribute. We used following definitions for the quality attributes in the game projects: *Modifiability*, the game architecture and implementation should be easy to change in order to add or modify functionality; and *Testability*, the game architecture and implementation should be easy to test in order to detect possible faults and failures. These two quality attributes also were related to the course content.

III. MOTIVATION AND OVERVIEW OF XQUEST

XQUEST (XNA QUick & Easy Starter Template) [11] is a small and lightweight 2D game library/game template developed by the two master students Strøm and Kvamme at NTNU (co-authors of this paper) that contains convenient game components, helper classes, and other classes that can be used in the XNA game projects. The goal of the XQUEST project was to identify and abstract common game programming tasks and create a set of components that could be used by students of the course to make their programming life easier. We chose to focus mainly on 2D. There were a few reasons for this. First, the focus of the student projects is software architecture, not making a game with fancy 3D graphics. Second, students unfamiliar with game programming and 3D programming may find it daunting to have to learn the concepts needed for doing full-blown 3D in XNA, such as shade programming and 3D-modelling, in addition to software architectures. To keep the projects in 2D may reduce the effect of students only focusing on the game development instead of focusing on the software architecture issues. However, we still consider to implement basic 3D components in XQUEST and help documentation for the students interested in 3D gaming programming, but it is not a mandatory for students to use them.

A. Motivation for XQUEST

From the feedback of using XNA in the software architecture course [21], the majority of the students thought there was much time focus on game issues and little time on software architecture, even the XNA environment is very developer friendly with a high level graphic API. Following Table I is a collection data from 46 students' replies in software architecture course.

Besides negative feedback above, from our teaching experiences we have to extend more features to improve XNA more suitable in the software architecture course:

- Save time in C# learning and programming
- Provide appropriate guidance and cases on mini 3D game programming.
- Provide cases of good designed software architecture based on XNA and XQUEST.

TABLE I. COLLECTION DATA ABOUT DEVELOPING TIME ON XNA FROM STUDENTS

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Q: I spent too much time developing the game play and not enough time on the architecture	5%	30%	10%	40%	15%

- Provide some documentation to explain the trade-off between architecture design and COTS, especially XQUEST components.

B. Design Principles for XQUEST

Here were the general principles used to design XQUEST:

1) *Flexible structure*: Due to XQUEST is used for teaching software architecture, and students will design game projects based on suitable software architectures like Model-View-Controller, Pipe and filter, Layered, Task Control and etc., we tried to provide flexible components for the students that would not hinder the students to design their own software architecture.

2) *Easier to use*: From our teaching experiences, 90% students had programming skills in Java, but not in C#. XQUEST would help them to learn XNA in an easy and quick way with good comments on code and supported documentation. And XQUEST was based on XNA and it would be easier for students to use and save development time. Students could learn it quickly even they only had experience in Java programming.

3) *3D guidance*: We intended to lead students into the world of 3D, and gave them the basic ideas of 3D programming. But 3D programming needs more time on 3D models and some basic 3D transformations to 2D on screen. We used several demos in XQUEST to show some 3D technologies and gave the students some sensorial 3D concepts in mind and got a quick entrance into the 3D world. Thus, they did not need to know the Math basics of how to do 3D transformations into 2D.

4) *Providing tutorials to investigate the software architecture in game*: Very little reference had been published on the subject of software architecture in game development, although some attempts had been done [12, 13, 14]. However, these attempts failed to deliver a general high-level presentation of software architecture topics, and tended to focus more on the design and implementation of software modules that were common in games. We looked into the differences between traditional software development and game development, as well as identifying different architectural and design patterns that were useful for game development. Also, portraying the challenges of designing and implementing game architectures could be proved useful for determining the scope of such an endeavor.

5) *Reflect the quality attributes in a game architecture*: Quality attributes were the driving force behind every big decision in the development process, and had to be considered

at all times. We identified some quality attributes that were most relevant for game development, such as modifiability, testability, availability or usability. We would look at structures and patterns that underline certain quality attributes, and to use these elements in a game architecture.

C. XQUEST Structures and Components

The XQUEST library would be presented component by component. XQUEST functionality was split into eight components. Their relationships could be found in Fig. 2, and we put the XQUEST.GameObjectManagement component in the middle purposely to indicate its importance. It contains the game object system, which is at the heart of XQUEST.

Here is a brief description of each component.

1) *XQUEST.GameObjectManagement* contains the game object system, responsible for handling game objects such as players, enemies, power ups, etc. The object system allows for many different types of objects, 2D or 3D, and provides state tracking and a flexible collision detection system.

2) *XQUEST.CameraSystem* provides functionality for setting up both 2D and 3D cameras to view a scene or track game objects in multiple perspectives.

3) *XQUEST.GameStateManagement* handles state and state transitions in the game. It uses the concept of game screens. A game screen can be a menu screen, an inventory screen, a combat screen, etc.

4) *XQUEST.Audio* handles audio-related functionality like playback of sound effects and music, adjustment of volume levels, grouping into categories, etc.

5) *XQUEST.Misc* contains miscellaneous components of utility that do not fit into any other namespace.

6) *XQUEST.Input* handles querying and interpretation of keyboard, mouse, and Xbox 360 game pad input.

7) *XQUEST.Helpers* contains convenient helper classes for common tasks.

Besides of above components, we also provided some demos and directions to illustrate how to use these components, what kind of architecture used in one demo and what types of attributes (Modifiability, testability or others) does it focus on.

IV. EVALUATION OF XQUEST USED IN A SOFTWARE ARCHITECTURE COURSE

One goal of this paper was to investigate the XQUEST's application results in software architecture course. Concretely, we investigated the following research questions:

- RQ1: What is the usability of XQUEST?

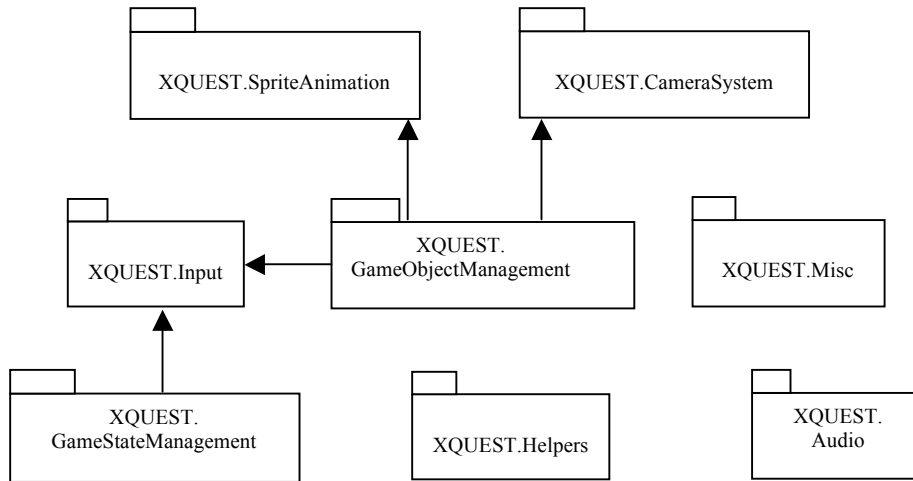


Figure 2. Structure View of XQUEST

- RQ2: What is the suitability and usefulness of XQUEST?
- RQ3: What is the usefulness of the specific components in XQUEST?
- RQ4: What other positive or negative issues are related to XQUEST?

A. Preparation for Evaluation

In this part, we present our research methods and students' background during the evaluation process.

1) The research methods:

a) *System Usability Scale*: The System Usability Scale [17], hereafter SUS, is a usability questionnaire consisting of ten generic Likert items. Responses to the questionnaire result in a score, called the SUS score, a single number between 0 and 100 indicating the overall usability of the system being studied. The SUS is used for subjectively measuring usability of a system on a high level. The outcome of a SUS questionnaire is a score within the range of 0 to 100, where higher values indicate a higher measured usability of the system. Each item in the SUS is responded to by assigning a scale value from 1 to 5, where 1 indicates strong disagreement and 5 indicates strong agreement. To calculate the SUS score, we first sum together the score contributions for each question. Each question's score contribution is a number in the range 0 to 4. There are totally 10 questions, for odd-numbered questions (1, 3, 5, 7, 9), the score contribution is given by the scale position minus 1. For even-numbered questions (2, 4, 6, 8, 10), the score contribution is 5 minus the scale position. The sum of the score contributions are then multiplied by 2.5 and divided by the number of replies to the survey to obtain the final SUS score. We had incorporated the SUS items in our XQUEST questionnaire.

b) *Empirical investigation*: The survey was based on the method of conducting an empirical investigation [15]. We had

applied recognized methods combined with our own subjective assessments to implement the survey. Measureable items in the questionnaires had been formed as Likert [16] items. These were not questions, but statements that the respondents responded to by specifying their agreement to the statements. We had used 5-level Likert items, where the levels of agreement were: Strongly Disagree; Disagree; Neutral; Agree and Strongly Agree. The items from the questionnaires were assessed subjectively. And these subjective analyses were based on our teaching experiences.

2) The participants and Environments.

The participants of our survey were postgraduate students of the software architecture class spring 2008 at NTNU. They used an online e-learning platform during the course. The questionnaires were published three days after the delivery deadline of students' projects on the e-learning platform by using its survey functionality. Each question was prefixed with a context name indicating which section it belonged to. The participants and environment was authentic in the sense that the students of the course were the intended users of XNA and XQUEST.

B. Results from System Usability Scale (SUS) Questions

Here we present the results from the SUS part of the XQUEST questionnaire.

The result of SUS score is shown in Table II. Our system achieved a score of 60.53 out of a possible 100 and it is above the average usability, which indicates that the system is not difficult to use. The main challenge for some students was to spend much time on becoming familiar with 2D/3D structure in XQUEST. This process can be improved by giving an introduction lecture about 2D/3D concept after first simple exercise when the students had already setup some experiences and context of XNA programming environment. We also need to improve 2D/3D components into two separate components in next XQUEST version.

C. Results from Suitability and Usefulness Questions

The results from the questions about suitability and usefulness of XQUEST are shown in Table III. This questionnaire was only for students using both XNA and XQUEST in project. We received a total of 19 responses from the students using XQUEST out of the 46 students that worked on an XNA game project.

From the investigation, it showed that students could use XQUEST as a template or referencing it as a library, 40% modified the code of XQUEST and 30% kept it unchanged. This reflects of one successful design philosophy of XQUEST was to create it as abstract and reusable as possible, students could chose any ways that help the project design.

Q2 shows a positive result that XQUEST prepared most commonly used components for students and it saves the time in game development.

And we found 60% students disagreed that they spent too much time looking into the source code of XQUEST. This positive result indicates our good documentation work and self-explanatory public interfaces and we also thanks to Visual Studio greatly benefits source code commenting in that these comments show up on the IntelliSense [18] tooltips that pop up while the programmer is typing in code. For example, when creating a new instance of a class, the IntelliSense will display any comments available for the different parameters that the constructor accepts.

It is also infeasible that students should both focus on architectural matters and on technical matters, but from Q4 result, to a certain extent, XQUEST still could help one third students more on architectural matters than on technical matter.

D. Results from Usefulness of Every Component Questions

From Table IV result, we could find out how the students used XQUEST in their projects, every components value and where we should focus on the improvement.

As we expected, the most popular component was the Animated Sprite Framework. Since all groups worked on 2D games, this is understandable since sprite rendering is the easiest way to output graphics in a 2D environment. Another popular component was the Game Object Management component. When going through the XNA deliveries, we were surprised to find out that most groups did not create their own implementation of the `IGameObject` interface, but rather used the standard `BasicGameObject`. Using `BasicGameObject` has some limits, because it is tightly interwoven with the Sprite Animation Framework for representing the game object using sprites. This implies that groups that used this approach, also needed to use the Sprite Animation Framework. Looking at the high percentage of students who responded to have used both of these components, there is no doubt that the use of `BasicGameObject` is the main reason for this. So this is the point that reminds the students to pay attention during teaching.

In third place comes the `InputManager` component. This is probably the most useful component in XQUEST, since every game needs to handle input in some forms. It contains several methods for supporting all the XNA input devices such as

keyboard, mouse, and up to four Xbox 360 game pads. By looking at the deliveries, we found that most games were single-player games played with a keyboard, or hot seat multiplayer games where all players shared the same keyboard. Some games used the mouse as the primary input device, but very few implemented game pad support, since they did not have access to Xbox 360 game pads during the project unless they brought one themselves. The input needs may therefore have not been so great that it required a component like the `InputManager`. We will therefore simplify the functions in `InputManager` component to minimize the amount of code the students needs to read to save total time of code reading, such as delete input support for XBOX360 according to the practical application from students.

The least used components were the `AudioManager` and `TextOut` components. Having music and sound effects in your game may not take the greatest priority in a school project where the evaluation criteria leans more towards software architecture and fulfilling an assigned quality attribute. For this reason, many groups decided not to implement audio features in their games to save development time, and hence no need for the `AudioManager` component. Still, almost half of the games used this component. The `TextOut` component was a component we thought would be more popular. It is really simple to use, and has features that makes it very convenient for text display. It may be the fact that text display is so simple that the students did not see the need for using it. The standard way of displaying text with `SpriteBatch` may fulfill all the desired text rendering needs. In this way, we could also cut some functions in `TextOut` to save coding read workload.

E. Open Questions Analysis

Table V is the collection of main feedback from students to the open question. 20% of the students agreed that some components were missing in XQUEST. Pixel-perfect collision detection is a very performance-intensive operation that we described as not suitable for a multi-purpose game object system such as the one in XQUEST. However, we decided to include support for it in next XQUEST version. It is disabled by default, but can be enabled on a per-object basis, meaning the user is in total control of how the collision detection should be executed for every object in the scene.

`BasicGameObject` is definition not supposed to be flexible. The flexibility of the game object management system in XQUEST lies in the `IGameObject` interface, of which `BasicGameObject` is an implementation. As expressed above, we were surprised that so few groups did not take advantage of this flexibility by providing their own implementation of the `IGameObject` interface. By doing so, they could have tailored it for their game. Instead, they chose to use the standard implementation in `BasicGameObject`, which of course also constrained them to using the `Sprite` class for the graphical representation.

TABLE II. RESULTS FROM THE SUS

Question	Sum score contribution of 19 students
1 I think that I would like to use this system frequently.	35
2 I found the system unnecessarily complex.	52
3 I thought the system was easy to use.	50
4 I think that I would need the support of a technical person to be able to use this system.	55
5 I found the various functions in this system were well integrated.	47
6 I thought there was too much inconsistency in this system.	48
7 I would imagine that most people would learn to use this system very quickly.	44
8 I found the system very cumbersome to use.	45
9 I felt very confident using the system.	40
10 I needed to learn a lot of things before I could get going with this system.	44
Sum:	460
SUS Score:	$460 * 2.5 / 19 = 60.53$

TABLE III. THE 5 GENERAL QUESTIONS LABELED Q1-Q5

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
Q1: I found that I could use XQUEST as is without modifications	15%	25%	30%	25%	5%
Q2: I think XQUEST saved me a lot of time and effort by providing components and functionality that I otherwise would have had to create myself	10%	10%	15%	40%	25%
Q3: I spent too much time looking into the XQUEST source code	10%	50%	25%	15%	0
Q4: I think XQUEST helped me focus more on architectural matters and less on technical matters	5%	35%	30%	30%	0

TABLE IV. QUESTIONS ABOUT EVERY COMPONENT IN XQUEST

#	I used the following components of XQUEST		
Component	Sprite/ AnimatedSprite	GameObject Manager	InputManager
Percentage	90%	85%	75%
Component	TextureStore	AudioManager	TextOut
Percentage	65%	40%	30%

TABLE V. OPEN QUESTION COLLECTION

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Q: I think there were components missing that most students could benefit from	0	25%	55%	10%	10%
Q: If you felt there were components missing, which ones would you like to see in a future version of XQUEST?					
A1. Sprite layers and pixel collision detection.					
A2. Pixel-based collision detection, system for being called with certain intervals, better modifiability.					
A3. More flexible BasicGameObject, allowing non-sprite objects.					

V. RELATED WORK

This paper described experiences how to improve and enhance the XNA teaching functions in a software architecture course. As far as we know, XNA is always *directly* used in education, there are only few papers describe its application in education and no paper goes further to describe the idea to extend the XNA's structure to enhance its features as a teaching aid for certain course. However, there are some related approaches used in education described in this section.

Joe Linhoff describes a game development course that uses the XNA platform to allow a heterogeneous group of students

to gain experience in all aspects of console game creation [31]. It uses the features of XNA directly for the teaching, such as Pipeline or console that could be XBOX360 to activate students' programming interesting.

Youngblood describes how XNA game segments can be used to engage students in advanced computer science education [27]. Game segments are developed backs providing the full code for a segment of a game with a clear element left for implementation by a student. The paper describes how XNA was used in a artificial intelligence course where the students was asked to implement a chat bot, motion planning, adversarial search, neural networks and flocking. Finally the

paper describes seven design principles for using game segments in CS education based on lessons learned.

Oliver Denninger and Jochen Schimmel present their experiences utilizing game programming for project courses based on XNA [30]. Game programming usually involves many repetitive and time consuming tasks such as accessing hardware resources and managing game content, but since XNA framework relieves programmers from many of the tedious tasks and allows them to develop a feature complete game and to gain experience with the process of software development, students were so fascinated by the subject that they prefer to spent more time on the courses.

El-Nasr and Smith describes how the use of modifying or modding existing games can be used to learn computer science, mathematics, physics and ascetic principles [26]. The paper describes how they used modding of the WarCraft III engine to teach high school students a class on game design and programming. Further, the describe experiences from teaching university students a more advanced class on game design and programming using the Unreal Tournament 2003 engine. Finally, they present observations from student projects that involve modding of game engines. Although the paper claims to teach students other things than pure game design and programming, the game engine is used in the context of game development courses.

The Labyrinth [28] was implemented in Java and it is a flexible and easy-to-use computer game framework. The framework enables instructors to expose students to very specific aspects of computer science courses. The framework is a finished game in the Pac-Man genre, highly modular, and it lets the students change different aspects of the game. However, it cannot be used to develop different genres of game and there is little room for changing the software architecture of the framework.

The JIG (Java Instructional Gaming) project [29] is a collaborative effort between Scott Wallace (Washington State University Vancouver) and Andrew Nierman (University of Puget Sound) in conjunction with a small group of dedicated students. It has three aims: 1) to build a Java instructional game engine suitable for a wide variety of students at all levels in the curriculum; 2) to create a set of educational resources to support the use of the game engine at small, resource-limited, schools; and 3) to develop a community of educators that use and help improve these resources. The JIG project was proposed in 2006 and the JIG engine 1.0 is available now.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the principles to design XQUEST to improve XNA teaching functions for students in the exercise of the software architecture course. Furthermore, we evaluated the XQUEST application and analyzed several aspects of XQUEST's suitability, usefulness and usability based on questionnaires. In many aspects, the results show that XQUEST enhances XNA in suitability as a teaching aid in software engineering learning, and that it can be a useful and helpful extension to understand XNA. The results also show that it is easy to use and save students time in development, and

let students have more time to focus on the practice of course theory.

Due to the first time using XQUEST in the software architecture course, from our experiences and evaluations, more works need to be done in improving the components in XQUEST to make them more useful, and updating the documentation due to updated XNA versions. We will go further to extend game library and enriching help resources of XQUEST in 3D development.

ACKNOWLEDGMENT

We would like to thank Jan-Erik Strøm and Trond Blomholm Kvamme for implementing XQUEST and for their inputs to this paper. This work has been sponsored by the Leiv Eriksson mobility program offered by the Research Council of Norway.

REFERENCES

- [1] Alex Baker, Emily Oh Navarro, and André van der Hoek. Problems and Programmers: an Educational Software Engineering Card Game. In ICSE '03: Proceedings of the 25th International Conference on Software Engineering, pages 614–619, Washington, DC, USA, 2003. IEEE Computer Society.
- [2] Emily Oh Navarro and André van der Hoek. SimSE: an Educational Simulation Game for Teaching the Software Engineering Process. In ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, pages 233–233, New York, NY, USA, 2004. ACM Press.
- [3] Lasse Natvig, Steinar Line, and Asbjørn Djupdal. Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals. In FIE 2004: Proceedings of the 2004 Frontiers in Education Conference, 2004.
- [4] N. Landry, "Microsoft XNA: Ready for Prime Time?," in CoDe Magazine. vol. Sept/Oct, 2007.
- [5] Microsoft, "XNA.net", <http://www.xna.net/>. Retrieved April 24, 2008.
- [6] Microsoft, "XNA Creators Club Online", <http://creators.xna.com/>. Retrieved May 21, 2008.
- [7] F. Luna, Introduction to 3d Game Programming with Direct X 9.0c. Plano: Wordware Publishing, Inc, 2006.
- [8] T. Miller, "Managed DirectX 9 Graphics and Game Programming," Sams Publishing, 2004.
- [9] P. Clements L. Bass and R. Kazman. Software Architecture in Practice Second Edition, 2003. Addison-Wesley.
- [10] A. I. Wang, T. Stålhane. Using Post Mortem Analysis to Evaluate Software Architecture Student Projects , Conference on Software Engineering and Training 2005, 8 pages.
- [11] Trond Blomholm Kvamme and Jan-Erik Strøm, "Evaluation and Extension of an XNA Game Library used in Software Architecture Projects", Master thesis in NTNU, June 2008.
- [12] A. Rollings and D. Morris, Game Architecture and Design, 2 ed.: New Riders Publishing, 2003.
- [13] D. Eberly, 3d Game Engine Architecture. Amsterdam: Morgan Kaufman Publishers, 2005.
- [14] R. Rucker, Software Engineering and Computer Games. Boston: Addison-Wesley, 2003.
- [15] V. R. Basili, "The Experimental Paradigm in Software Engineering," in Dagstuhl Workshop. vol. Experimental Software Engineering Issues: Critical Assessment and Future Directives, H. D. Rombach, V. R. Basili, and R. W. Selby, Eds. Dagstuhl Castle, Germany: Springer-Verlag, 1992, pp. 3-12.
- [16] Microsoft, "Shader Series Primer: Fundamentals of the Programmable Pipeline in XNA Game Studio Express," 2007. <http://creators.xna.com/downloads/?id=128>

- [17] J. Brooke, "SUS - A quick and dirty usability scale," in Usability Evaluation in Industry London: Taylor and Francis, pp. 189-194.
- [18] Wikipedia; "IntelliSense", <http://en.wikipedia.org/w/index.php?title=IntelliSense&oldid=208720089>. Retrieved May 7, 2008.
- [19] Apple. "iPhone Dev Center", <http://developer.apple.com/iphone/>, Retrieved February 2, 2009.
- [20] Google. "Android - An Open Handset Alliance Project", <http://code.google.com/intl/en/android/documentation.html>. Retrieved February 2, 2008.
- [21] Bian Wu, Alf Inge Wang, Jan-Erik Strøm, Trond Blomholm Kvamme, "An Evaluation of Using a Game Development Framework in Higher Education," CSEET, pp.41-44, 2009 22nd Conference on Software Engineering Education and Training, 2009
- [22] G. Sindre, L. Nattvig, M. Jahre, "Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals", to appear in IEEE Transaction on Education.
- [23] B.A. Foss and T.I. Eikaas, "Game play in Engineering Education - Concept and Experimental Results", The International Journal of Engineering Education 22(5), 2006.
- [24] A. I. Wang, O. K. Mørch-Storstein, T. Øfsdahl, "Lecture quiz - a mobile game concept for lectures", The 11th IASTED International Conference on Software Engineering and Application (SEA 2007), November 19-21, 2007.
- [25] A. I. Wang, T. Ø. and O. K. Mørch-Storstein: "An Evaluation of a Mobile Game Concept for Lectures", 21st IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2008), Charleston, S. Carolina, USA, April 14-17, 2008,.
- [26] M. S. El-Nasr and B. K. Smith, "Learning through game modding", ACM Computer Entertainment 4(1), Jan. 2006.
- [27] Youngblood, G. M. 2007 Using XNA-GSE Game Segments to Engage Students in Advanced Computer Science Education. In The 2nd Annual Microsoft Academic Days Conference on Game Development, February 22-25.
- [28] Distasio, J. and Way, T. 2007 Inclusive computer science education using a ready-made computer game framework. In ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, 116-120.
- [29] Washington State University Vancouver and University of Puget Sound. 2008 The Java Instructional Gaming Project. Web: <http://ai.vancouver.wsu.edu/jig/>, Retrieved June 2008.
- [30] Oliver Denninger, Jochen Schimmel, Game Programming and XNA in Software Engineering Education, Proceedings of Computer Games and Allied Technology (CGAT08), 2008.
- [31] Joe, L. and S. Amber (2008). Teaching game programming using XNA. Proceedings of the 13th annual conference on Innovation and technology in computer science education. Madrid, Spain, ACM.
- [32] Wang, Alf Inge; Wu, Bian. An Application of a Game Development Framework in Higher Education. International Journal of Computer Games Technology 2009 ;Volume 2009.