Game Development Frameworks for SE Education

Bian Wu, Alf Inge Wang
Norwegian University of Science and Technology

***Abstract*--This paper presents a literature survey about the method of creating/modifying a game on a game development framework (GDF) as an assignment to learn software engineering (SE), and we share our recommendation for choosing an appropriate GDFs.**

I. INTRODUCTION

Games have been used in schools for many years to help students learn skills in math, language, science, engineering and other domains in an interesting and motivating way. Another innovative way is to provide exercises that require students to work individually or in groups to modify or develop a game as a part of a course using a game development framework (GDF) to learn skills within computer science or software engineering (SE) [1-3]. GDF denotes all toolkits used to develop games. This paper focuses on criteria for selecting appropriate GDFs that can be used in student exercises to learn SE skills. The motivation for teaching SE through game development is to utilize the students' enthusiasm for game creation. More specifically, we wanted to investigate how GDFs are used in SE education through our own experiences and a literature survey.

II. EXPERIENCES

We present our experiences as an example to explain how we apply XNA as a GDF in software architecture course in 2008 [1]. In this 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 to let students work in groups and apply the methods and theory from the course to design a software architecture for a game and implement it based on the XNA framework. The project consists of the following phases:

1) COTS (Commercial Off-The-Shelf) exercise: Learn the technology to be used through developing a simple game.

2) Design pattern: Learn how to use and apply design pattern by making changes in an existing game.

3) Requirements and architecture: List functional and quality requirements and design the software architecture for a game.

4) Architecture evaluation: Use the ATAM (Architecture Tradeoff Analysis Method) evaluation method to evaluate the software architecture of game project in regards to the quality requirements.

5) Implementation: Do a detailed design and implement the game based on the created architecture and on the changes from the Architecture evaluation.

6) Project evaluation: Evaluate the project as a whole using a PMA (Post-Mortem Analysis) method.

The course staff issued the tasks of making 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. Finally, we got positive feedback from students' survey [1-3].

III. RESEARCH CONTEXT SURVEY

The scope of this paper is limited to the selection of GDFs only used in SE education, as SE is the major teaching field where GDFs applied. The survey is based on literature from IEEE Xplore and ACM digital library.

When looking into the background of how GDFs are used in SE education, we focus on why apply a GDF in a SE course in the first place. It is common to describe the teaching design using a GDF from the angle of teachers previous experiences from the course, not explaining its learning theory context [4, 5]. However, we still can find literatures that explain this learning activity, especially in SE education field.

For example, the paper "Learning Through Game Modding" [2] presents its experiences of using a GDF to teach students SE. It considers the learning activity of modifying/creating a game in a GDF in SE education as a design activity that has educational benefits such as learning content, skills, and strategies [6]. Design activities are meaningful and engaging to students for exploring skills (analysis, synthesis, evaluation, revision, planning and monitoring) and concepts to understand how they can be applied in the real world. Further, learning by modifying/creating games can be considered as variant of several available construction activities.

Seymour Papert presents programming as one example of the constructionism learning theory [2]. Constructionism involves two activities [7]. The first is the mental construction of knowledge that occurs with world experiences, a view borrowed from Jean Piaget's constructivist theories of learning and development. The second is a more controversial belief that new knowledge can be constructed with particular effectiveness when people engage in constructing products

that are personally meaningful. The important issue is that the design and implementation of products are meaningful to those creating them, and that learning becomes active and self-directed through the construction of artifacts. In SE education, creating games on GDFs could be this artifact.

A similar positive response to above is [8]. It presents a case study to use double stimulation [9] to guide the exercise designs based on a GDF. It also considers that using a GDF in SE education could be a knowledge construction process. It describes how to use double stimulus to guide a teaching activity, including the learning activity from creating a game. In schools, learners face a challenge, a problem, or a task that has been designed for a particular pedagogical purpose or they face situations that are likely to appear in work and public life. In both cases the purpose of exploiting tools is for learners to respond to such challenges. Based on constructionism, it constructs the relationship between the educational tasks and the material artifacts. This relationship is at the heart of Vygotsky's notion of double stimulation [9], a method for studying cognitive processes and not just results. In a school setting, typically the first stimulus would be the problem or challenge to which learners are expected to respond. The second stimulus would be the available mediating tools, like GDFs.

Similarity, using GDFs in SE education is related to Problem-Based Learning (PBL) [10, 11]. PBL is a pedagogical model that emphasizes the role of a real-life problem and a collaborative discovery process in learning [12]. Within a typical PBL setting, students are first given a challenging but realistic problem of significant size, relevant to the learning objectives of a given course. They are then encouraged to solve the problem in a group throughout the semester as independently as possible with minimum help from the instructor of the course. Apart from the traditional lecture-oriented teaching approach, PBL puts more emphasis on the instructors' role as facilitators, to prepare meaningful and interesting problems, and to create and organize course materials in a manner that students have a just right dose of information in each class to incrementally develop a final solution based on a GDF to the primary problem of the semester.

IV. SURVEY OF GDFS USED IN SE EDUCATION

In order to identify the main feature of several GDFs, we classify them according to two categories: GDFs for novices, and GDFs for developers.

The focus of GDFs for novices is to provide visual interface for customizing game templates and to allow creating or designing games with little or no programming skills. Here are examples of GDFs used in assignments to learn SE from literature survey and its resource link: Alice [13-16]; Scratch [17-19]; CeeBot Series [20]; Warcraft3 Editors [2]; Never Winter Night Toolsets [21] ; Greenfoot [22]; Game maker [23, 24]; StarLogo TNG [25]; and Wu's castle [26]. The way these GDFs are used in SE education varies. E.g., Alice and Scratch are typically used for introducing programming or object-orientation concept to students where the students get introduced to programming concepts through visually manipulating objects in order to implement some simple game behaviors from scratch. Other GDFs are mainly editors or modifiers for existing games, such as the Warcraft3 editor or the Never Winter Night toolsets. The educational approach when using such GDFs are totally different, as the focus is on tailoring or modifying existing behavior in the game instead of building everything from scratch.

The focus of GDFs for developers is to offer toolkits that support development of high quality 2D/3D rendering, special effects, physics, animations, sound playback, and network communication in common programming languages such as C++, C# and Java. Most of the commercial game engines belong in this category. Here are examples of such GDFs used in SE education: BiMIP [27]; Unreal Engine [2, 5]; XNA [28, 29]; XQUEST[30]; XNACS1Lib framework [31]; Android/Sheep [8]; MUPPETS framework [32]; and SIMPLE framework [33]. When using GDFs such as XNA, XQUEST and Android/Sheep, the students will mainly develop everything from scratch and follow the whole software cycle. But for other GDFs, such as Unreal game engine, the basic game functionality is in place and the programming will focus on the game instance. This is a more restrictive approach in what you can learn and the application of the software development process. If the goal of the SE course is to go through the whole software cycle, game engines are not usually suitable GDFs.

V. RECOMMENDATIONS

From both of our experiences and literature survey, introducing a GDF in a SE course can have positive effects such as higher enrollment, improved student motivation and project group dynamics, and more effort put into projects/ assignments [34]. The higher enrollment is mainly due to most of students think it is more interesting to work on a game project than e.g. a banking system. The improved student motivation and group dynamics is mainly due to collaboration of the teamwork provides the possibility of creating their own imaginative games and game development require other than pure technical skills.

However, there are also some obvious disadvantages. The most evident one is that some students will focus too much on the game development thus loosing focus on what they shall learn in SE. This means that the design of the course and the project must be carried out in such a way that the students are forced to learn and use the SE methods and disciplines being taught in the course. One approach to enforce SE elements in exercises and projects is to require documentation during the whole project focusing on the SE learning goals and emphasize that the evaluation of the exercise and project will mainly focus on the quality of these SE deliverables and less on the game being produced. This is from our experiences on using XNA in the software architecture course. To ensure the SE focus, the students had to deliver part-deliveries focusing on different areas of software architecture, such as design and

architectural patterns, functional and quality requirements, a software architecture for the game described through several views, an architectural evaluation, and an implementation of the game where the students had to adhere to their quality requirements, their chosen patterns and their designed software architecture.

Further, it is really important to choose the appropriate GDF to be used in a SE course. There are many factors that come into play when conceiving an assignment based on a GDF:

Educational goal: The educational goal of the SE course will greatly affect the choice of GDF, e.g. if the focus of the course will be on requirements, software architecture, design, implementation, testing, maintenance, project management or the software process. As mentioned before, SE courses focusing on the whole development cycle should use GDFs that allow the students to develop a game from scratch such as XNA. However, if a SE course only focuses on testing or quality assurance, a game engine can be very effective for the education goals such as Unreal can work very well. Another important factor is whether course's focus on procedural programming vs. Object Oriented (OO) programming. For SE courses with more technical requirements, GDFs such as XNA, XQUEST or Android/Sheep are more appropriate. In other courses, the most important goal is not to learn programming, but rather to learn the SE principles such as requirements, design, and the project management. For such courses, GDFs with visual programming such as Alice, Scratch or the Warcraft3 editor can be used.

SE constraints: All GDFs have constraints related to SE in how they have been designed or how they are released. One example is open source GDFs that make it possible to do white-box testing on the GDF, while for other GDFs the source code is not available for the students. Open source GDFs are also important in courses where it is necessary to understand the details of the components used in students' game creation. Further, some GDFs might constrain how you can design your games, what design and architectural patterns you can use, how event handling must be managed, the freedom of expanding the GDFs functionality and more. These constraints must be integrated in the SE teaching to introduce the students to the real world where software never is built from scratch. Another important issue is the openness of the GDF to other tools. This issue could be very important e.g. the integration of test tools.

Programming experience: The programming experience of the students will highly affect the choice of GDF between the ones for novices and the ones for developers. Another factor is what programming languages the students know, such as Java, C#, C, C++ etc. E.g. to use XNA/XQUEST or Android/Sheep, the students must know OO programming well and be familiar to design patterns and OO principles in addition to C# and Java. And some GDFs offer their own programming languages to simplify the game programming (scripting). From our own experience, the hardest part for the students is not the programming language itself but rather the libraries and APIs they have to learn.

Staff expertise: It is essential that the course staff have technical experience in a GDF used in a SE course to provide help to students to avoid having them focusing on only the technical matter and not the SE challenges. From our own experiences on running a software architecture course, it is necessary to have dedicated staff to provide technical GDF support. Although it is important that the teacher teaching the SE course knows the basics of the GDFs, it is not necessary for this teacher to have a complete technical insight of the GDF. However, it is critical to have course staff available that can help the student with technical problems during the exercises or project.

Usability of the GDF: To avoid too much focus on technical matters and problems, it must be possible to learn the GDF quickly without too much of a hassle. In practice this means that the GDF must be well-designed, have a logical structure, provide high-level APIs, provide correct, updated and available documentation, provide helpful and many examples, and have many available tutorials. It is also a huge advantage if an active developer community supports the GDF. XNA is a good example of a GDF, which is well designed with high-level APIs, well documented and supported, and an active community. It is recommended to establish a GDF community within a course e.g. using a web forum, as well as encouraging the students to use external web resources.

Technical environment: Technical considerations must be taken into account when selecting a GDF. Typical technical considerations include operating system and hardware compatibility, license policies, tool support, support for third-party tools, and how difficult the software is to install on the students' PCs. The technical requirements might also be an economical issue, as the choice of GDF might force hardware upgrades or paying for expensive licenses. A typical problem is e.g., that XNA runs only on Windows, and many students now have PCs running Linux or Mac OS X. As our experiences on using XNA in a software architecture course, many of the students did not have a Windows PC at first and these students were told to use the available computer labs. Soon, however, we discovered that the existing computer labs running thin-clients were insufficient for running XNA. The problem was partly solved by the students themselves as many of the Mac OS X and Linux users installed Windows on their PCs (dual boot). In addition, our department gave access to a computer lab with stand-alone PCs powerful enough to run XNA.

The list of considerations above should be included in the process of finding the appropriate GDF for a SE course. If an appropriate GDF is chosen and the project or exercises "force" students to provide SE deliveries through the semester, the result is likely to be improved project results as the students are better motivated and put more effort into the work.

VI. Conclusions and Further Work

Through our experiences and literature survey on the theoretical context and various GDFs used in SE education, it

has shown that this method has potential motivation to help students to learn SE courses. In order to select an appropriate GDF, we also identify the impact factors that play important roles on design process for the course when using GDFs in SE education. We believe that our study can provide the guidance for the teachers or researchers in the area of SE education, even for the GDFs' designers in the aspect of the enhancement of GDFs' educational features.

However, time, cost and expertise are significant barriers to experimenting with GDFs in educational settings, and there are limitations to what skills can be acquired using GDFs [2]. Based on our initial survey, this area deserves more research on the applications of GDF for SE education and how to design and improve the teaching process to maximize the effectiveness of using GDF in education.

REFERENCE

[1] A. I. Wang and B. Wu, "An Application of a Game Development Framework in Higher Education," *International Journal of Computer Games Technology,* vol. 2009, 2009.

[2] M. S. El-Nasr, "Learning through game modding," *Computers in entertainment,* vol. 4, 2006.

[3] B. Wu*, et al.*, "An Evaluation of Using a Game Development Framework in Higher Education," *Proceedings / Conference on Software Engineering Education and Training,* 2009.

[4] S. v. Delden, "Industrial robotic game playing: an AI course," *J. Comput. Small Coll.,* vol. 25, pp. 134-142, 2010.

[5] E. L. Wynters, "3D video games: no programming required," *J. Comput. Small Coll.,* vol. 22, pp. 105-111, 2007.

[6] S. Puntambekar and J. L. Kolodner, "Toward implementing distributed scaffolding: Helping students learn science from design," *Journal of Research in Science Teaching,* vol. 42, pp. 185-217, 2005.

[7] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas.* New York, 1980.

[8] B. Wu*, et al.*, "Extending Google Android's Application as an Educational Tool," presented at the The 3rd IEEE Information Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL 2010), Kaohsiung, Taiwan, April 12-16, 2010. , 2010.

[9] L. S., *Mind in society: The development of higher psychological processes*, 1978.

[10] A. Garrido*, et al.*, "Using graphics: motivating students in a C++ programming introductory course," in *EAEEIE Annual Conference,*, 2009, pp. 1-6.

[11] J. Ryoo, "Teaching object-oriented software engineering through problem-based learning in the context of game design," in *21st Conference on Software Engineering Education and Training*, 2008, p. 137.

[12] H. S. Barrows, "A taxonomy of problem-based learning methods," *Medical Education,* vol. 20, pp. 481-486, 1986.

[13] R. H. Seidman, "Alice first: 3D interactive game programming," *SIGCSE Bull.,* vol. 41, pp. 345-345, 2009.

[14] E. W. Amerikaner, "Introduction to computer science using Alice 2.0: tutorial presentation," *J. Comput. Small Coll.,* vol. 25, pp. 141-141, 2010.

[15] K. Anewalt, "Making CS0 fun: an active learning approach using toys, games and Alice," *J. Comput. Small Coll.,* vol. 23, pp. 98-105, 2008.

[16] L. Werner*, et al.*, "Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study," presented at the Proceedings of the 4th International Conference on Foundations of Digital Games, Orlando, Florida, 2009.

[17] G. Fesakis and K. Serafeim, "Influence of the familiarization with "scratch" on future teachers' opinions and attitudes about programming and ICT in education," presented at the Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, Paris, France, 2009.

[18] P. A. G. Sivilotti and S. A. Laugel, "Scratching the surface of advanced topics in software engineering: a workshop module for middle school students," *SIGCSE Bull.,* vol. 40, pp. 291-295, 2008.

[19] W. Jui-Feng*, et al.*, "Teaching Boolean Logic through Game Rule Tuning," *IEEE Transactions on Learning Technologies,* vol. 3, pp. 319-328, 2010.

[20] T. Phit-Huan*, et al.*, "Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception," in *International Conference on Computer Technology and Development, 2009(ICCTD '09)*, 2009, pp. 42-46.

[21] J. Robertson and C. Howells, "Computer game design: Opportunities for successful learning," *Computers & Education,* vol. 50, pp. 559-578, 2008.

[22] M. Al-Bow*, et al.*, "Using game creation for teaching computer programming to high school students and teachers," *SIGCSE Bull.,* vol. 41, pp. 104-108, 2009.

[23] Y. Rankin*, et al.*, "The impact of game design on students' interest in CS," presented at the Proceedings of the 3rd international conference on Game development in computer science education, Miami, Florida, 2008.

[24] Yulia and R. Adipranata, "Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment," in *2nd International Conference on Education Technology and Computer (ICETC),*, 2010, pp. V2-355-V2-359.

[25] K. Wang*, et al.*, "3D game design with programming blocks in StarLogo TNG," presented at the Proceedings of the 7th international conference on Learning sciences, Bloomington, Indiana, 2006.

[26] M. Eagle and T. Barnes, "Experimental evaluation of an educational game for improved learning in introductory computing," presented at the Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, USA, 2009.

[27] A. Garrido*, et al.*, "Using graphics: motivating students in a C++ programming introductory course," in *EAEEIE Annual Conference, 2009*, 2009, pp. 1-6.

[28] B. Wu*, et al.*, "An Evaluation of Using a Game Development Framework in Higher Education," *22nd Conference on Software Engineering Education and Training, 2009,* pp. pp.41-44, 2009.

[29] K. Sung*, et al.*, "Game-Themed Programming Assignment Modules: A Pathway for Gradual Integration of Gaming Context Into Existing Introductory Programming Courses," *IEEE Transactions on Education,* 2010.

[30] B. Wu*, et al.*, "XQUEST used in software architecture education," in *International IEEE Consumer Electronics Society's Games Innovations Conference,(ICE-GIC 2009)*, 2009, pp. 70-77.

[31] R. Angotti*, et al.*, "Game-themed instructional modules: a video case study," presented at the Proceedings of the Fifth International Conference on the Foundations of Digital Games, Monterey, California, 2010.

[32] K. J. Bierre and A. M. Phelps, "The use of MUPPETS in an introductory java programming course," presented at the Proceedings of the 5th conference on Information technology education, Salt Lake City, UT, USA, 2004.

[33] H. C. Jiau*, et al.*, "Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics," *IEEE Transactions on Education,* vol. 52, pp. 555-562, 2009.

[34] A. I. Wang, "Extensive Evaluation of Using a Game Project in a Software Architecture Course," *Transactions on Computing Education (ACM),* vol. Volume 11,, February 2011. 2011.