THE USE OF GAME DEVELOPMENT IN COMPUTER SCIENCE AND SOFTWARE ENGINEERING EDUCATION

Alf Inge Wang, alfw@idi.ntnu.no and Bian Wu, bian@idi.ntnu.no
Dept. of Computer and Information Science,
Norwegian University of Science and Technology,
Trondheim, Norway

Abstract – This chapter presents the results of a literature survey on the use of game development in software engineering (SE) and computer science (CS) education.

Games and game development has been used in recent years to increase motivation, engagement and learning, and to promote doing career in software and computers. The authors of this chapter wanted to get a bird's eye perspective on how game development has been used in the recent years to teach other things than just game development.

The literature survey includes research articles from the period 2004 to 2012, and it investigates among other things in what context game development has being used, the subjects and topics taught through game development, and what tools and frameworks being used for game development to learn SE and CS. Further, this chapter includes a description of how game development was used in our software architecture course at the Norwegian University of Science and Technology, and the gained experiences from running this course. Finally, we come with our recommendations for choosing an appropriate game development framework for teaching SE and CS.

1. Introduction

Games are increasingly being using in education to improve academic achievement, motivation and classroom dynamics (Rosas, Nussbaum et al. 2003). Games are not only used to teach young school children, but it is also used to improve university education (Sharples 2000). There are many good examples that education within software engineering and computer science can benefit from using games and game development for teaching (Baker, Navarro et al. 2003, Natvig, Line et al. 2004, Navarro and Hoek 2004, El-Nasr and Smith 2006, Foss and Eikaas 2006, Distasio and Way 2007, Sindre, Nattvig et al. 2009). Over the years, there have been published several papers that describes how game development has been used to promote, motivate or teach software engineering and computer science. The goal of this book chapter is to present results from this research through a literature study ranging from the years 2004 to 2013. Further, we have included some of our own experiences from teaching software architecture through game development from 2008 to 2013. One very important factor when incorporating game development into a CS/SE course is the choice of an appropriate game development framework (GDF) and tools. To help others, we have included six factors that should be considered when choosing a GDF for a CS/SE course.

This chapter is attempting on giving answer to the following research questions:

- *RQ1: In what context/topics is game development used in CS/SE education?* The focus of this research question is to find the topics or domains where game development has been used as a teaching method documented in the research literature.
- *RQ2: What game development frameworks are used in CS/SE education?* The focus of this research question is to find the most common tools used, and possibly see if there are any shared experiences from using these tools.
- *RQ3: What are the experiences from using game development in CS/SE education?* The focus here is to see if there are any common experiences from using game development to teach CS/SE subjects.

The rest of the chapter is organized as follows: Section 2 describes experiences from using game development in a software architecture course, Section 3 presents a literature survey on how game development is been used in CS/SE education, Section 4 gives recommendations in how to select an appropriate game development framework, and finally Section 5 concludes this chapter.

2. Experiences from Using Game Development in a Software Architecture Course

To demonstrate how game development can be used in software engineering education, we included a section that describes how game development was introduced in a software architecture course, and the results from doing so.

2.1 The Software Architecture Course

The software architecture course at the Norwegian University of Science and Technology (NTNU) is for post-graduate CS and SE students where the workload is 25% of a semester. About 70-130 students attend the course every spring. Most of the students are Norwegian students, but about 20% comes from EU countries and other continents. The textbook used is Software Architecture in Practice (Clements and Kazman 2003) and additional articles are used to cover topics such as design patterns (Coplien 1998), software architecture documentation standards (Maier, Emery et al. 2004) and view models (Kruchten 1995). The learning outcomes from the course are:

> "The students should be able to *define* and *explain* central concepts in software architecture literature, and be able to *use* and *describe* design/architectural patterns, methods to design software architectures, methods/techniques to achieve software qualities, methods to document software architecture, and methods to evaluate software architecture."

The course is mainly taught in three ways:
1) Ordinary lectures (in English)
2) Invited guest lectures from the software industry
3) A software development project that focuses on software architecture

The software architecture course at NTNU is taught differently than at most other universities, as the students also have to implement their designed architecture in groups of 4-6 students. The motivation for doing so is to make the students understand the relationship between the architecture and implementation, and to be able to perform real evaluation of whether the architecture and the resulting implementation fulfill the quality requirements specified for the application. In the first phase of the project, the students get familiar with the COTS to be used through developing some simple applications. In the second face, the students are asked to implement a selection of design and architectural pattern in the chosen COTS. The third phase focuses on producing a complete requirement specification including quality requirements, as well as designing and documenting the architecture according to the specified quality requirements and architectural drivers. In phase four, the groups evaluate each others software architectures against the described quality requirements using the Architecture Tradeoff Analysis Method (ATAM) (Kazman, Klein et al. 1998). In the fifth phase, the students carry out a detailed design, they implement the application according to the software architecture, and finally test the application. The sixth and final phase involves a post-mortem analysis of the whole project to learn from successes and mistakes (Bjørnson, Wang et al. 2009). The student groups will deliver reports from all the phases, and code and executable for the phases 1, 2 and 5. The awarded grade is based on grading of the project (30%), and a final written examination (70%). The project will not be graded until the final delivery in phase 6, and the students are encouraged to improve and update deliveries during the whole project cycle.

2.2 How Game-Development was Introduced in the Software Architecture Course

Before 2008, the project in software architecture course at NTNU asked the students to develop an autonomous robot controller for the Khepera Robot Simulator in Java. The reason a robot controller was chosen for the project was that this domain had many well-documented architectural patterns with described software qualities (Elfes 1990, Lumia, Fiala et al. 1990, Simmons 1992, Toal, Flanagan et al. 2005). Many students found the robot domain difficult and not very interesting.

In 2009, the course staff decided to add video game as a domain to the software architecture project to boost motivation and interest of the course. To be able to do this, we had to do four changes to the course to be able to provide a game development project with emphasis on software architecture (Wang and Wu 2011).

First, we had to do some *course preparations* that involved choosing a game development framework (GDF) and making the necessary preparation to use it in the project. To do this we searched for existing GDFs and evaluated how well they were suited for teaching software architecture, what GDF resources (manuals, guides, examples, tutorials etc.) were available, and how to introduce the GDF to the students. Our choice landed on XNA from Microsoft due to its good documentation, tutorials and examples, high-level API, a programming language which was close to Java (C#), as well as the motivating factor of deploying games on PCs, XBOX and mobile devices. Although XNA provides a high-level API, we decided to develop the XQUEST framework to make it even easier to create games in XNA (Wu, Wang et al. 2009). The reason for doing this was to allow the students to focus more on software architecture than technical and game-specific matters.

Second, we had to *change the syllabus*. For the students to be able to create software architectures for the game domain, they needed some theoretical backing on design and architectural patterns for the game domain, how to design game architectures, and how to specify quality attributes for games. It turned out that this task was rather difficult. In contrast to the autonomous robot domain, there was not much mature literature on game architecture or patterns. The result was that we added some chapters from the book Game Architecture and

Design (Rollings and Morris 2004) to the syllabus, which was supported by a set of own composed slides with descriptions of relevant design and architectural patterns for games and how to specify quality attributes for games.

Third, *changes to the project* were made. The course staff decided to let the student teams themselves choose between the robot and the game project. The main structure of the project had to remain the same, but we had to support two domains and two different COTS. One major difference between the two domains was that for the robot controller project, the variation in what you could do with the application was limited, while for the game project we wanted to encourage creativity. Thus, the functional requirements for the robot version of the project were fixed while not fixed for the game version. Another thing we had to change was how to grade the project. The grading of the project has the main emphasis on quality and completeness of documentation and code, and the relationship between the code and the architecture. The main difference in grading the two variants was that to get the top grade (A) in the robot version, the robot had to solve its task efficiently and elegant, while for the game version it was required to be impressive in some way. It was also a requirement that both the robot controller and the game had to have a certain level of complexity in terms of number of classes and its structure of components.

Four, *changes were made to the schedule and the staff*. The change of course staff was basically that we had to hire one extra TA to provide technical support on the XNA framework. The main changes in the schedule were to add an extra two-hour COTS introduction lecture to give an introduction to both the robot controller and C# and XNA (in parallel), changing and extending a lecture on design and architectural patterns to include the game domain, and added one lecture on software architecture in games.

All in all, the chances were not major but it required quite a lot of effort from the course staff. Most effort was required to get familiarized with the XNA framework, adapting the project, and creating and finding theoretical foundation to be used in lectures on game architectures.

2.3 Gained Experiences

Since 2009 we have collected data on the game development project in the software architecture course at NTNU. Most of the experiences we have gained are positive, but we have also identified some challenges. One clear indication that the students welcomed a game project was that 73% of the students chose the game project vs. 27% the robot project in 2009. Since 2009, the percentage of students choosing the robot project has decreased. Further, the amount of students attending the course has increased from around 70 students in 2008 to over 150 in 2013. The course evaluations from 2009 to 2013 clearly show that game development is both popular among the students and motivates and engage the students to learn software architecture. The four most popular game genres developed by students in these projects are shooter (38%), strategy (25%), board games (13%), and platform games (12%). All the games developed so far have been 2D games. The course staff recommends the development of 2D games to avoid too much focus on technical challenges. Some students groups admit that they focus too much on the game itself and less on software architecture. Another comment from the students is that they spend way too much time on the game project so it hurts their effort in other courses. Figure 1 shows screenshots from the game BlueRose developed by one student group in XNA.

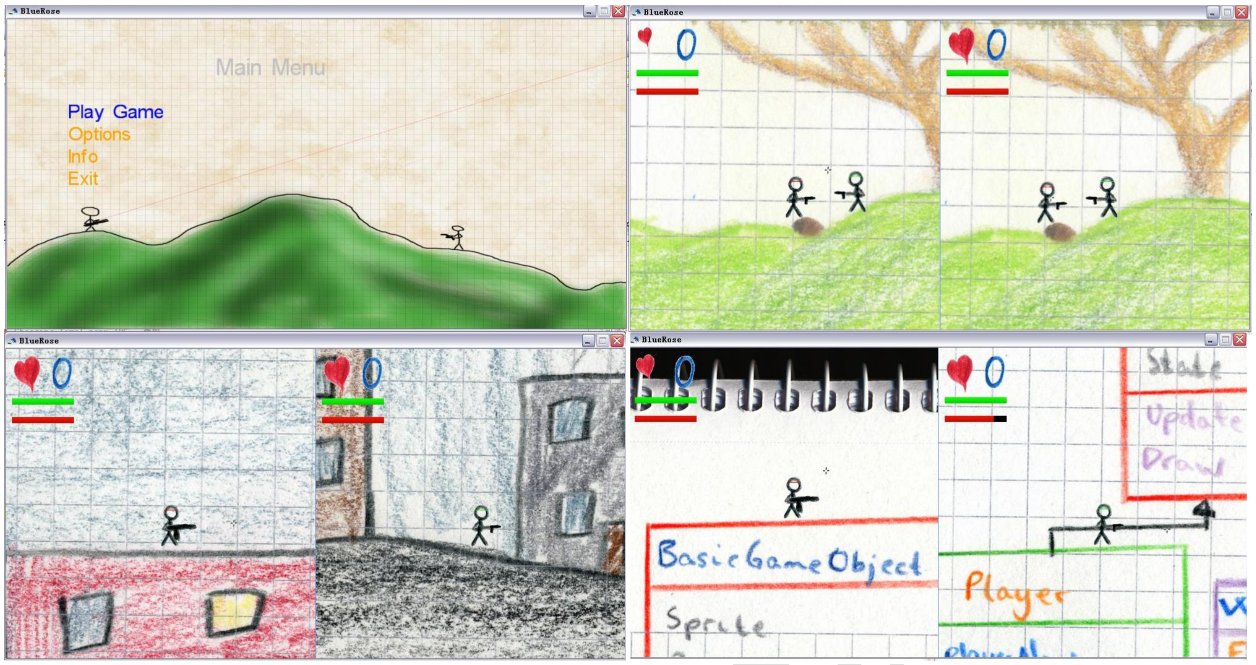

Figure 1 Screenshoots from the BlueRose XNA game

From a teacher perspective, we have found that it is necessary to have good insight both of the GDF (XNA) as well as the domain of game architecture to enable the students to learn well from a game development project. A noticeable difference in the students' attitude seen by the teacher is that the students are eagerly interesting in learning the XNA framework, while they are more reluctant to learn about the Khepra robot simulator.

In (Wang 2011), we performed an extensive evaluation of the software architecture course where we compared the effect of the robot controller vs. the game project. The main conclusion was that game development projects can be used successfully to teach software architecture. Further, the results showed that students who chose the game project produced software architectures with higher complexity and they put more effort into the project than students that chose the robot project. We found no statistically significant differences in the final grades awarded between the two variants of projects. However, the students that did the game project obtained on average a higher grade on their project than on their written examination, while the students that did the robot project on average scored higher on the written examination than on their project. Although not significant, the students that did the game project on average got a higher grade on the project compared to students that did the robot project.

3. Survey on the use of Game Development in CS and SE Education

This section presents the results from a literature survey on use of game development in CS and SE in articles from 2004 to 2012. The review followed the established method of systematic review (Khan, Ter Riet et al. 2001, Higgins and Green 2008), undertaking the following stages: 1) Protocol Development, 2) Data Source and Search Strategy, 3) Data extraction with Inclusion and Exclusion Criteria, and 4) Synthesis of Findings. The search for literature was carried out on the digital research article databases ACM, IEEE Xplore, Springer, and Science direct. The search string being used was "game AND (learning OR Teaching) AND (lecture or curriculum or lesson or course or exercise)". In addition, we searched the Google Scholar database with the search string “(game development) and (lecture or teaching or learning)” to pick up those we had missed.

The screening process of relevant articles was carried out in two steps. First, the abstracts were checked, and those articles that did not describe use of game development to learn or motivate for CS or SE were rejected. Articles that were typically removed focused on game theory or business games, or article that mentioned words like game and learning and development but focused on something else. The second step was to read through the whole text of the remaining articles and exclude those who were too vague or outside the scope of the survey. 66 articles relevant to game development and CS/SE education were found from the period 2004 to 2012. We noticed that there has been an increase in number of articles published on this topic since the year 2004 up to 2009. From 2009 to 2012 there have been published on average 12 articles per year on this topic. Further, out of the 66 articles, 65% were published by ACM, 26% by IEEE, and the reminding 9% equally divided between Springer, Elsevier and Hindawi. Another interesting fact is that 59% of the relevant articles are from America (mainly USA), 24% from Europe, 12% from Asia, and 5% international studies (involving studies across continents and countries).

3.1 Answer to RQ1: In what context/topics is game development used in CS/SE education?

To answer RQ1, we carried out an analysis of the 66 relevant articles. The first step of the analysis was to group the articles into the three main categories Software Engineering, Computer Science, and Applied Computer Science. Articles that ended up in the Applied CS category were articles where CS was applied (e.g. using computer tools like Alice or Scratch) but the main focus of the course was something else (e.g. like art). Of the 66 articles, 55% were articles about game development in CS subjects, 26% in SE, and 19% in applied CS. Another interesting finding was that 81% of the articles described the use of game development at colleges and universities, 9% in high schools, 9% in middle schools, and 1% in an elementary school (one article).

The second step of the analysis was to identify the specific topics or domains where game development was used as a teaching aid. The following sub-sections describe the results from this analysis grouped according to CS, SE, and applied CS.

3.1.1 CS Topics/Subjects where Game Development is being used

Figure 2 shows the distribution of topics/subjects found in the articles where game development was used to teach CS. The figure clearly illustrates that for CS, game development is mainly used to teach programming (77%). Most of the articles within this category describe how game development was used as an introduction to CS and programming. There few articles that described game development used for advanced programming courses. The second largest topic was found to be artificial intelligence (11%). In these articles, students were typically asked to program artificial intelligence (AI) for non-playable characters (NPCs) in a game or create or change the AI in a strategy game. Other topics in CS where game development has been used are algorithm, parallel computing, data structures, and Boolean logic.

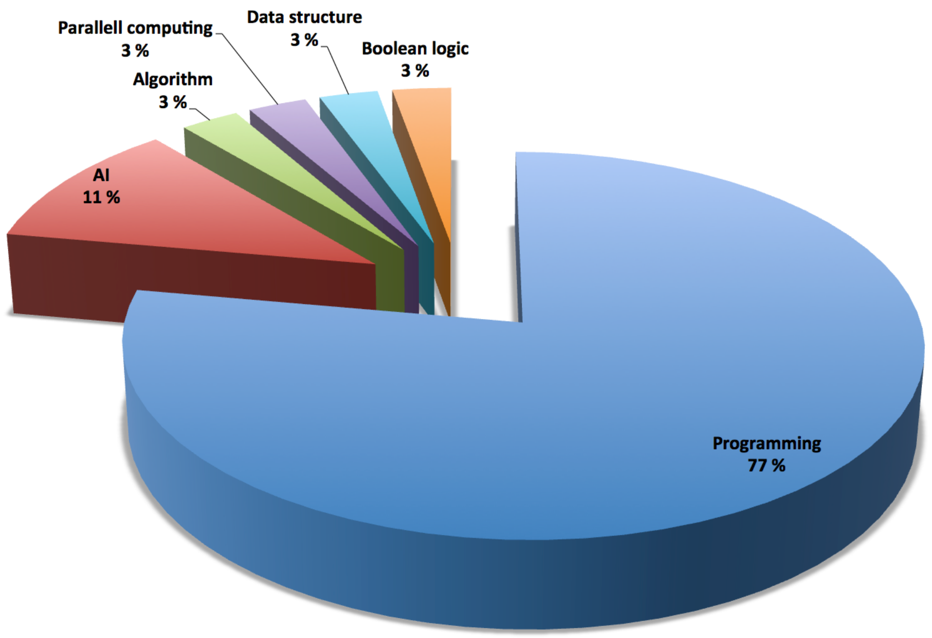

Figure 2 Distribution of topics taught using game development in CS

Table I gives an overview of the articles found where game development was used in CS education at college/university level. The table describes the main topic/subjects taught, and briefly how game development was used in the course.

Table I shows that most articles are about programming courses using game development to motivate students. However, there is a great variety in how game development is used in the different studies. In some studies, game development was only used in one or a few student assignments. Other articles describe approaches where the whole course was designed around a large game development project. The typical games developed in these projects and assignments are simple classical 2D games such as Pac Man BomberMan or board games.

Table I Articles where game development was used in CS education at university level

Article	How game development is used in course	Topic
(McGovern and Fager 2007)	Introduce AI adding code to game	AI
(van Delden 2010)	Create game in robot simulation	AI
(Timm, Bogon et al. 2008)	Learn AI adding code to board game	AI
(Barella, Valero et al. 2009)	Use multi-player game framework to learn AI	AI
(Weng, Tseng et al. 2010)	Learn logic through changing Pac Man code in Scratch	Logic
(Detmer, Li et al. 2010)	Develop learning games for high school teachers	Programming
(Hundhausen, Agrawal et al. 2010)	Develop the game battleship as an assignment	Programming
(Carter, Bouvier et al. 2011)	Improve motivation and learning through game development	Programming
(Rick, Morisse et al. 2012)	Use Greenfoot for simulations	Programming
(Alvarado, Dodds et al. 2012)	Introduce games to make CS more attractive for women	Programming
(Coleman and Lang 2012)	Meta-study of use of use of game and game development in courses	Programming
(Greenberg, Kumar et al. 2012)	Learning CS using context of art and creative coding	Programming
(Anewalt 2008)	Learn programming, math and logic in Alice	Programming
(Eagle and Barnes 2009)	Change loops and arrays in the game Wu's castle to learn programming	Programming

(Chaffin and Barnes 2010)	Teach students to develop serious games	Programming
(Williams and Beaubouef 2012)	Create simple games in assignments like tic-tac-toe to learn programming	Programming
(Sung, Hillyard et al. 2011)	Learn programming developing games in XNA	Programming
(Angotti, Hillyard et al. 2010)	Learn programming developing games in XNA	Programming
(Goldweber, Barr et al. 2013)	Motivate CS through game programming	Programming
(Fesakis and Serafeim 2009)	Learn programming through developing games in Scratch	Programming
(Kurkovsky 2009)	Survey on use of various mobile development platforms	Programming
(Garrido, Martinez-Baena et al. 2009)	Use games and visual programming to learn C++	Programming
(Jiau, Chen et al. 2009)	Learn to program using game-based simulations and metrics	Programming
(Chang and Chou 2008)	Learn programming by adding code to BomberMan game in C	Programming
(Tan, Ting et al. 2009)	Learning programming through contributing to game design components	Programming
(Stuurman, van Eekelen et al. 2012)	Develop snake game as introduction to programming	Programming
(Ambrósio and Costa 2010)	Implement simple naval battle game in assignment to learn about algorithms	Algorithm
(Lawrence 2004)	Teach data structures through advanced game intelligence programming	Data structure
(Bierre and Phelps 2004)	Learn programming through development of 3D objects and games	Programming

Table II presents articles where game development is used for promoting or teaching CS courses in middle and high schools. The articles reporting from middle and high school experiences mainly focus on how to motivate for CS using visual programming environment to create games. All the articles apart from one focused on programming. Interestingly, the non-programming article gave an example of how parallel computing can be introduced to high school students through games and game software. Parallel computing is a challenging and difficult topic, but through examples from games the students were able to learn and engage in the topic.

Table II Articles describing game development promoting or teaching in CS in middle and high schools

Article	How game development is used in course	Topic
(Wang, McCaffrey et al. 2006)	Teach 9th graders basic CS through game programming in StarLogo TNG	Programming
(Goldberg, Grunwald et al. 2012)	Promote CS in middle school through creating games like Frogger and Space Invaders	Programming
(Rodger, Dalis et al. 2012)	Promote CS in middle school through creating games in Alice	Programming
(Webb and Rosson 2011)	Promote CS for 8th grade girls through game programming in Alice	Programming
(Chesebrough and Turner 2010)	Demonstrate parallel computing in high school through games and game software	Parallel computing
(Al-Bow, Austin et al. 2009)	Using game creation for teaching programming both for students and teachers in high school	Programming
(Jenkins, Brannock et al. 2012)	Promote active learning and collaboration in high school through game programming	Programming

3.1.2 SE Topics/Subjects where Game Development is being used

Figure 3 presents an overview of the topics/subjects being taught in SE where game development was a part of the course. The most recurring topic is software architecture (41%). However, it is important to note that most of these articles come from the same source. Object-oriented design (23%) and software testing (12%) are two other popular SE topics where game development is used. Other SE topics we identified are teamwork, out-sourcing, reverse engineering, and software process (all of them 6%).

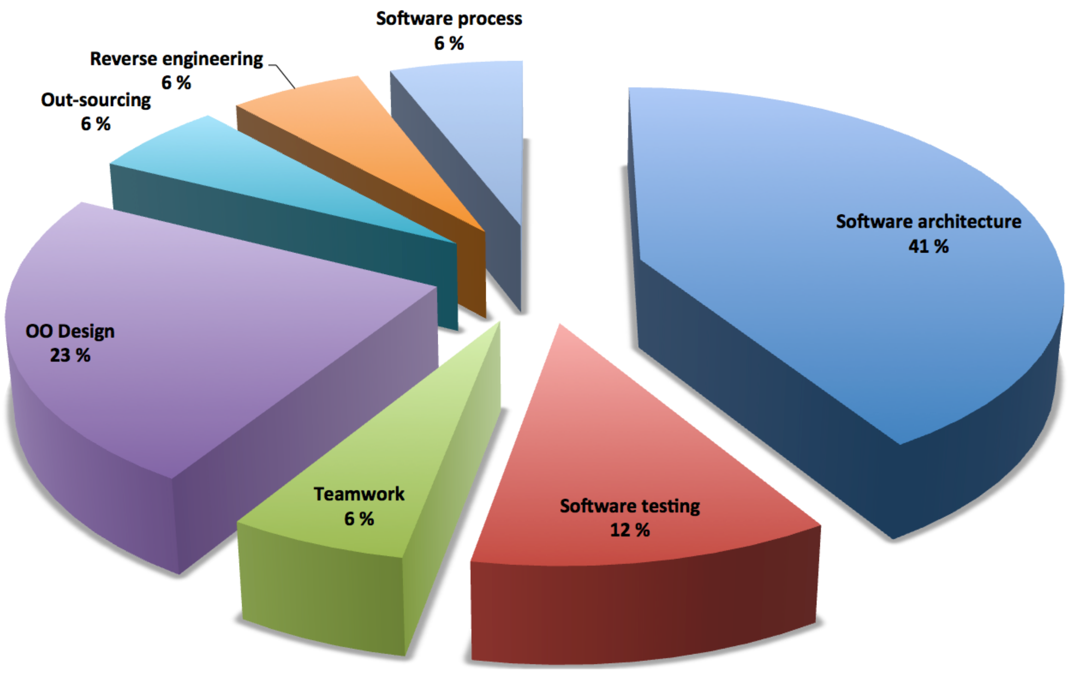

Figure 3 Distribution of topics taught using game development in SE

Table III presents an overview of the articles that describes how game development is used to teach SE. Similarly to the articles related to CS, some describe how game development was a minor part of the exercises (one or two assignments), while other courses were designed around a game development project that motivated the students to learn about a SE topic. We also discovered a variety in how the game development was carried out. Some articles described projects where students modified code from existing games, while other describe projects going through a complete development cycle and the games are developed from scratch.

Table III Articles describing game development in SE education

Article	How game development is used in course	Topic
(Chen and Cheng 2007)	Learn OO design and design through game programming	OO design
(Jun 2010)	Use game development to improve teamwork skills	Teamwork
(Wang and Wu 2009)	Learn design/architectural patterns and quality attributes through game development	Software architecture
(Adipranata 2010)	Learn OO design/programming using GameMaker and Warcraft	OO Design
(Ryoo, Fonseca et al. 2008)	Learn OO design/programming and RUP through game development and GUI programming	OO Design
(Mahoney and Gandhi 2012)	Learn reverse engineering by reverse engineering the executables of a Tetris game	Reverse engineering
(Nordio, Ghezzi et al. 2011)	Learn distributed and outsourced programming through the DOSE game platform	OO Design
(Wang 2009)	Learning software architecture through game development in XNA	Software architecture
(Wu, Wang et al. 2009)	Using high-level API in XNA to let students focus more on software architecture than game development	Software architecture

(Wu, Wang et al. 2010)	Using high level API in Android SDK to let students focus more on software architecture than game development	Software architecture
(Wang 2011)	Comparing the effect of using development of robot controller vs. game in software architecture course	Software architecture
(Wang and Wu 2011)	Learn software architecture through game development	Software architecture
(Wu and Wang 2012)	Comparing the effect of using development of apps. vs. games in software architecture course	Software architecture
(Honig and Prasad 2007)	Learning out-sourcing through development of various board games	Out-sourcing
(Smith, Tessler et al. 2012)	Develop the games Tetris and Boogle as assignments to learn software testing	Software testing
(Schild, Walter et al. 2010)	Learning scrum through game development in XNA	Software process

3.1.3 Applied CS Topics/Subjects where Game Development is being used

Figure 4 shows the distribution articles on topics within applied CS where game development was used. The two most recurring topics were game design (46%) and game development (31%). One could argue that game development should be a part of computer science, but in the articles described here game development was typically a course that focused on game design that also involved game programming, 3d-modelling and similar topics. The other two identified topics were art design (15%), and literacy (8%). The main difference between game design and art design is that art design has stronger emphasis on esthetics and artistic style and expression.

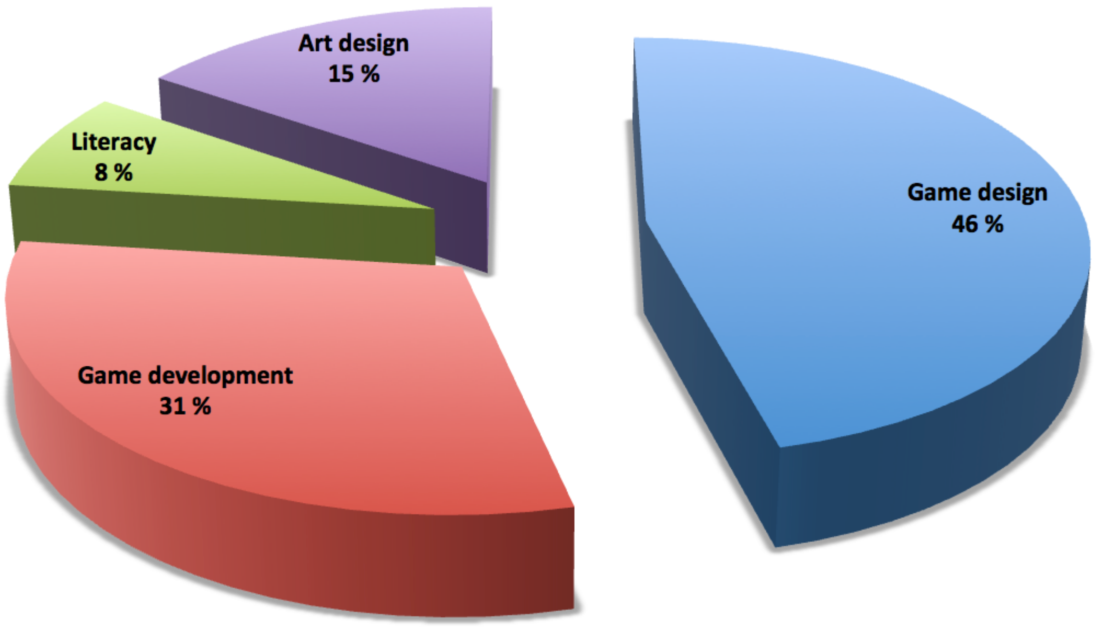

Figure 4 Distribution of topics taught using game development in Applied CS

Table IV presents an overview of the 12 articles we found where game development was used to teach applied CS. These articles displayed a much stronger emphasis on creating or changing games through graphical tools with less emphasis on programming. Typically, many of the articles focused on creating game levels using editors and tools to create terrains, game characters, game objects, and populate levels with game objects. Half of the articles describe use of game development at a pre college/university level.

Table IV Overview of articles where game development was used to teach applied CS

Article	How game development is used in course	Topic
(van Langeveld and Kessler 2009)	Promoting CS creating digital characters in 3D	Art design
(Rankin, Gooch et al. 2008)	Learn CS through game design and development in GameMaker	Game design
(Estey, Long et al. 2010)	Use game design in Flash and GameMaker to learn communication and team-working skills	Game design
(Seaborn, Seif El-Nasr et al. 2012)	Teach CS using game design and development in GameMaker in High School	Game design
(Huang, Ho et al. 2008)	Game development to teach integration of programming and art design	Game design
(Wynters 2007)	Motivate/Creation CS through modification in Unreal, Photoshop and 3DS Max	Art design
(Robertson and Howells 2008)	Learn game design through Neverwinter Nights 2 in Middle School	Game design
(El-Nasr and Smith 2006)	Learn CS, math and programming through changing code in Warcraft 3 and Unreal at High School and University	Game design
(Ritzhaupt 2009)	Free or cheap game development tools like Torque, Milkshape 3D, Audacity, and GIMP used to teach game development	Game development
(Sullivan and Smith 2011)	Teach high school students CS and programming skills through game development in high school	Game development
(Werner, Denner et al. 2009)	Learn programming through game development in Alice in Middle School	Game development
(Owston, Wideman et al. 2007)	Game development to motivate and engage students in literacy activities in Elementary School	Literacy

3.2 Answer to RQ2: *What game development frameworks are used in CS/SE education?*

To answer RQ2 we analyzed the 66 relevant articles with respect to the tools, frameworks, and programming languages used to develop games. Figure 5 shows the distribution of game development frameworks (GDFs) used for game development in the articles in the survey. The *others* category (15%) includes development frameworks or tools like CeeBot series, Maya, Neverwinter Nights 2 editor, Open GL, Open Scene Graph, Photoshop, Processing, Robot simulation framework (unspecified), Scala, StarLogo TNG, Torque, VPython 3D, and Windows binary. Figure 5 shows that many uses their own frameworks for developing games in their courses. Many of these frameworks were development in Java. One reason for using own frameworks is to adapt the development to fit the educational goals of the course. Two other popular GDFs are XNA and Java (both 11%). Many chose XNA because the flexible and high-level API and good documentation, while many chose Java since it was familiar to the student or that the context was a Java programming course. Other popular GDFs are Alice, Android SDK, C++, GameMaker, Flash and Scratch. It is interesting to note that the use of the currently very popular indie-game engine Unity 3D was not found in any article. This picture is likely to change over the years. E.g., Microsoft has decided to stop all support for XNA, which will force many over on new platforms.

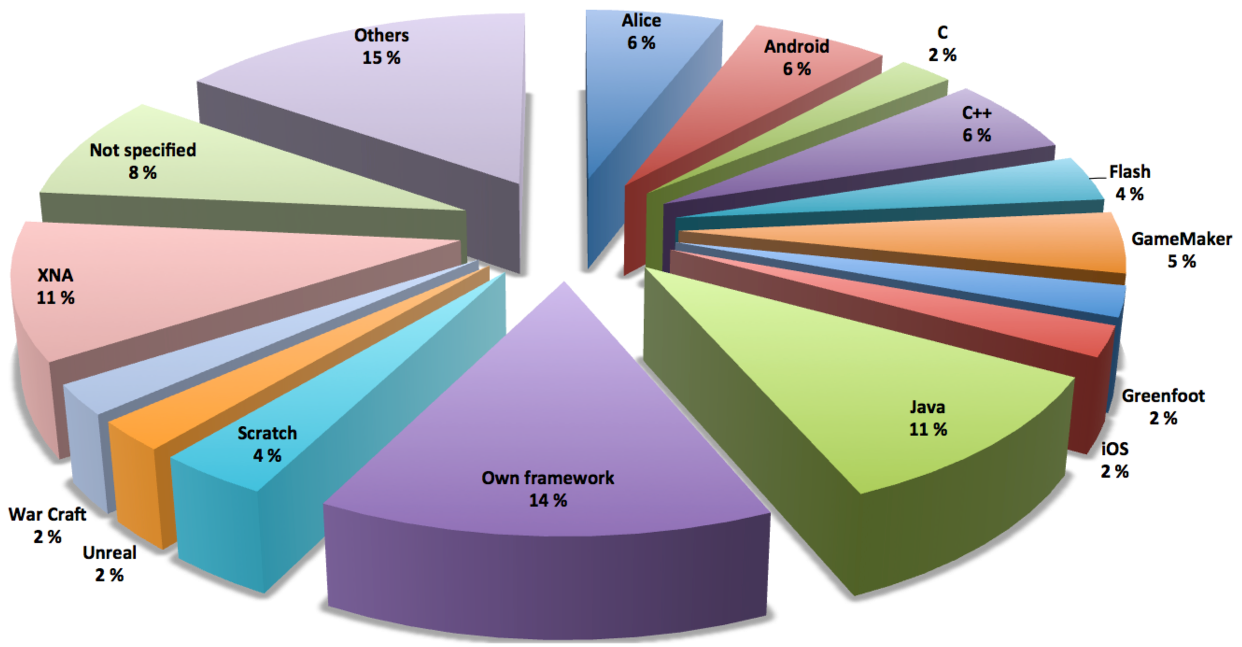

Figure 5 Game development frameworks used in CS, SE and applied CS

Figure 6 shows the distribution of programming languages used to teach SE, CS and applied CS through game development. In the articles in the survey, 27% uses some kind a visual programming/editing to develop games. The visual programming category includes Alice, GameMaker, Maya, Neverwinter Nights 2 (editor), Open Scene Graph, Photoshop, Scratch, StarLogo TNG, Torque Editor, Unreal editor, Warcraft editor, and Wu's Castle (own framework). The most common programming language is Java with 24%, where many articles describe the usage of additional framework in Java to make the game development faster, easier and a better match with the educational goals. C# is also being used frequently due to the XNA GDF.

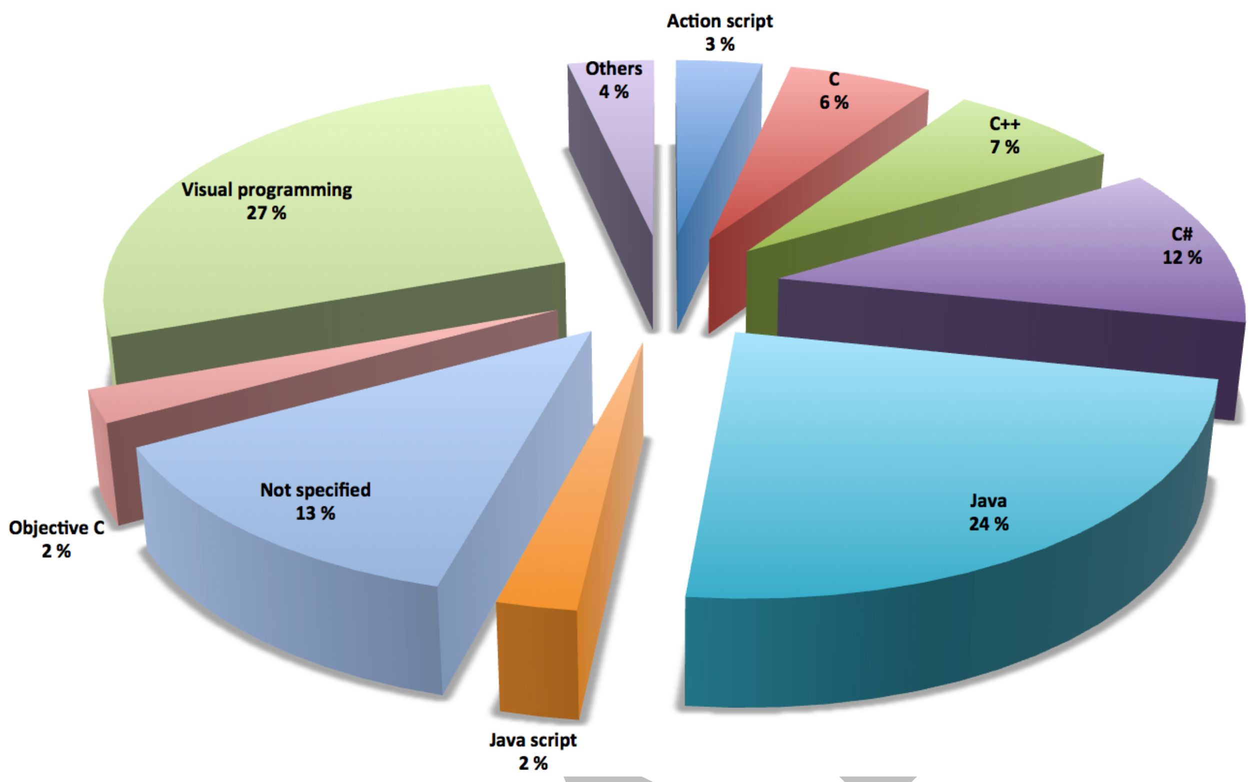

Figure 6 Programming languages used to teach SE, CS and applied CS through game development

The articles in the survey also describe how the GDFs are used and experiences from using them. Visual programming in Alice, Scratch, and StarLogo TNG works well for introducing programming, learning programming concepts & OO, promoting CS. Many conclude that visual programming is a fun and engaging way of learning logic and sequences. Experiences from using Scratch shows that visual programming can boost students confidence in programming that allows them to explore and improve their skills and knowledge significantly. Articles also show that mobile development platforms such as Android SDK or iOS SDK engage and motivate students through being able to play their own games on own devices and share them with friends.

The use of graphic and game libraries/APIs is important to reduce the required effort and complexity to develop software that is fun to interact with. In most articles in the survey, the students are asked to create simple 2D games from ground up. Another approach is to use *game modding*, which means to change or modify existing code to change the behavior and looks of a game. Some articles have proved that game modding in Unreal, Warcraft, and Wu's Castle work well to introduce programming and programming concepts without the need to implement the whole game. A motivating factor for doing so is that the game being modified can be a highly rated commercial game with impressive graphics and game play. Several articles have emphasized the importance of using high-level APIs to accelerate the development enabling a stronger focus on learning other things than game development like programming, software architecture, and AI. Finally, in studies from K-12, visual programming environments like Scratch and Alice are most commonly used. There are also several studies where visual programming environments are used to introduce university students to programming with good results. The main benefits for the latter, is a better understanding of programming concepts such as sequences and how different parts of a program interact.

3.3 Answer to RQ3: What are the experiences from using game development in CS/SE education?

This section summarizes experiences found in the articles in the survey regarding using game development for teaching CS and SE.

The majority of articles report positive effects of using game development in teaching. Some articles do not share any experiences other than game development was a part of their course. These are typically courses where game development was used in one or two assignments and did not play a major role.

Most studies in the survey reports that game development *improve student motivation and engagement*. Only one study reports about neutral or negative results (Rankin, Gooch et al. 2008), but it is unclear whether the negative results were caused by improper usage of game development or the limited number of subjects in the study. Most articles report that game development makes programming fun as the students' results can be observed visually on the screen. Further that students are more engaged and put more effort into projects when they do game development vs. other type of software development. However, only few studies report increased learning in terms of better grades. An observed problem with introducing game development in CS and SE courses is the tendency that some students focus too much on the game and game development instead of focusing on the topic being taught.

The survey discovered several examples of how game development can be used to learn other topics than only programming such as parallel computing, out-sourcing, software architecture, teamwork, AI, software testing, reverse engineering, logic, art and literacy. Also it showed that adding or modifying code in existing games is an effective and motivating approach for teaching AI, as the results of the AI can be observed visually in the game. This approach works both with games specially designed for teaching AI or the use of commercial games that comes with modification tools such as Warcraft or Unreal Tournament. A novel approach to teach out-sourcing described in one article is to have groups students from two different universities develop the same game, where parts of the game is out-sourced to the students from the other university (Honig and Prasad 2007). Similarly, another article described how reverse engineering was taught by making the students change the executable files of a Tetris game (Mahoney and Gandhi 2012).

Finally, many articles in the survey reported that game development worked very well in recruiting students to study CS or SE, as well as improving enrollment of existing CS and SE courses. Especially many articles that described the use of game development at middle and high schools reported that the students got a more positive attitude to CS and SE, and they were more likely to chase a carrier in this field.

4 Recommendations for the Use of Game Development Frameworks

In this section we will give some recommendations for how to choose an appropriate game development framework (GDF) that will give the best result when teaching CS or SE. The recommendations are divided into six main factors you should consider when introducing game development in a CS or SE course. The recommendations are based on the survey presented in this book chapter as well as (Wu and Wang 2011).

Educational goal: The educational goal of the course will greatly affect the choice of GDF. For instance if the course focuses on a complete development cycle, it should be possible to develop a game from scratch according to specified requirements, have tools available to specify the design and tools for testing. However, if the educational goal only focuses on some parts of the development cycle, such as software testing, AI, teamwork, software maintenance, out-sourcing or quality assurance, it is possible to use a game editor for an existing game like Unreal Tournament or Warcraft.

Subject constrains: Most CS or SE subjects put constraints on what you should be able to do in the GDF. E.g. if you teach a software architecture, it is very important to have the freedom to design and implement various design and architecture patterns in the framework chosen. Another example can be if your subject is AI, it is important that the framework or game being used allow specifying and implementing AI similar to what being taught in the textbook. Many times, additional articles must be used to link the subject to how it is used in games.

Programming experience: The programming experience of the students will highly affect the choice GDF. Typically, young students without any programming experience should start with visual programming environments such as Scratch or Alice. For students with programming experience, one factor that should be considered when choosing a GDF is what programming languages do the students know. What is most important here is whether the students only know procedural programming languages or also know object-oriented programming languages. Experiences from articles in the survey showed that it usually does not require too much work for students that e.g. know Java to learn C# or vice versa. The programming language used by the GDF should not be too far away from the students' programming background.

Staff expertise: One thing you have to consider when introducing game development into a CS or SE course is the course staff's knowledge and technical experience with a GDF. This is important both to plan the project to fit with the educational goals, as well as providing help to students getting stuck with technical problems. It is not necessary for the teacher to know everything about GDF in detail, but it is very useful to have at least one in the course staff that knows it fairly well. The minimum requirement is to be able to point the students to online resources where they can find help.

Usability of the GDF: We do not want the students to get stuck in technical problems during the project, as then they will not learn as much about the CS/SE topic. Thus, it is important that the GDF is easy to use and let the students be productive with relatively small amount of code. This means in practice that the course staff should before introducing game development in a course, create some simple games in the GDF to get a feeling of how difficult it is to use. The GDF should provide high-level APIs, it should be well structured, and it should have a logical structure. In addition, it is important that the GDF is well supported through documentation, tutorials, example code, and a living development community. It is also a good thing if all resources are free for the students to use.

Technical environment: Technical considerations should also be taken into account when choosing a GDF. One very important consideration is the software and hardware requirements of the GDF. E.g., if XNA is chosen, Microsoft Windows is required to run both the development platform and the resulting executables. This could be a challenge if many students have PCs running Mac OSX or Linux, or the computer labs runs other operating systems than

Microsoft Windows. Hardware requirements are also important, as some GDFs can have some tough requirements in terms of CPU, GPU, and memory. A problem could be that an existing computer lab does not have powerful enough PCs to run the GDF or the resulting games. Finally, it is important to consider licenses and cost of the GDF.

5 Conclusion

This chapter focused on describing how game development has been and can be used to teach and promote SE and CS. We shared our own experiences from introducing game development in a software architecture course. Further we presented a literature survey with articles from 2004 to 2012 on game development in CS and SE. Finally, based on our own experiences and the survey, we gave some recommendations on how to choose a game development framework to be used in a SE/CS course.

The goal of the chapter was to give answers to three research questions. In the *first research question* was about finding the contexts and topics game development is used in CS/SE education (RQ1). We found that game development is being used to teach a variety of subjects, but the most common subject is programming and more specifically introduction to programming. Other major CS/SE topics are AI, software architecture, OO design, and software testing. We also found that the majority of articles (81%) are studies from colleges or universities, while the remaining (19%) are from elementary, middle and high schools. There is also a great difference in how and how much game development is used in the various courses. Some courses simply use game development in a few short assignments, while others wrap the course around a large game development project.

The *second research question* was about what GDFs are being used for developing games in CS/SE courses (RQ2). The results of the literature study revealed a usage of a wide variety of tools and frameworks where the most frequent ones were own developed frameworks, Java, XNA, Android SDK, Alice, C++, GameMaker, Flash, and Scratch. In terms of programing languages the three largest categories were visual programming frameworks (e.g. Alice, Scratch etc.), C# and Java. We foresee that the GDFs and programming languages used for this purpose will change significantly in the time to come. E.g., Microsoft has stopped further development and support of XNA. At many universities there has been a shift from Java programming courses to Python and HTML5/Java script.

The *third research question* focused on experiences from using game development based learning. The main conclusion here is that the experiences are overall positive in terms of student motivation and engagement, and it works well for promoting CS/SE and recruiting students. However, there are too few studies still that show significant improvement in terms of academic achievements (grades). More studies must be performed to study the educational effects of game development to teach other topics.

Acknowledgement

We would like to thank Richard Taylor and Walt Scacchi at the Institute for Software Research (ISR) at University of California, Irvine (UCI) for providing a stimulating research environment and for hosting a visiting researcher.

References

Adipranata, R. (2010). Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment. Education Technology and Computer (ICETC), 2010 2nd International Conference on, IEEE.

Al-Bow, M., et al. (2009). "Using game creation for teaching computer programming to high school students and teachers." ACM SIGCSE Bulletin **41**(3): 104-108.

Alvarado, C., et al. (2012). "Increasing women's participation in computing at Harvey Mudd College." ACM Inroads **3**(4): 55-64.

Ambrósio, A. P. L. and F. M. Costa (2010). Evaluating the impact of pbl and tablet pcs in an algorithms and computer programming course. Proceedings of the 41st ACM technical symposium on Computer science education, ACM.

Anewalt, K. (2008). "Making CS0 fun: an active learning approach using toys, games and Alice." Journal of Computing Sciences in Colleges **23**(3): 98-105.

Angotti, R., et al. (2010). Game-themed instructional modules: a video case study. Proceedings of the Fifth International Conference on the Foundations of Digital Games, ACM.

Baker, A., et al. (2003). Problems and Programmers: an educational software engineering card game. Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, IEEE Computer Society.

Barella, A., et al. (2009). "JGOMAS: New Approach to AI Teaching." IEEE Transaction on Education **52**(2): 228-235.

Bierre, K. J. and A. M. Phelps (2004). The use of MUPPETS in an introductory java programming course. Proceedings of the 5th conference on Information technology education, ACM.

Bjørnson, F. O., et al. (2009). "Improving the effectiveness of root cause analysis in post mortem analysis: A controlled experiment." Information Software Technology **51**(1): 150-161.

Carter, J., et al. (2011). Motivating all our students? Proceedings of the 16th annual conference reports on Innovation and technology in computer science education-working group reports, ACM.

Chaffin, A. and T. Barnes (2010). Lessons from a course on serious games research and prototyping. Proceedings of the Fifth International Conference on the Foundations of Digital Games, ACM.

Chang, W.-C. and Y.-M. Chou (2008). Introductory c programming language learning with game-based digital learning. Advances in Web Based Learning-ICWL 2008, Springer**:** 221-231.

Chen, W.-K. and Y. C. Cheng (2007). "Teaching Object-Oriented Programming Laboratory With Computer Game Programming." IEEE Transaction on Education **50**(3): 197-203.

Chesebrough, R. A. and I. Turner (2010). Parallel computing: at the interface of high school and industry. Proceedings of the 41st ACM technical symposium on Computer science education, ACM.

Clements, P. and R. Kazman (2003). Software Architecture in Practices, Addison-Wesley Longman Publishing Co., Inc.

Coleman, B. and M. Lang (2012). Collaboration across the curriculum: a disciplined approach todeveloping team skills. Proceedings of the 43rd ACM technical symposium on Computer Science Education. Raleigh, North Carolina, USA, ACM**:** 277-282.

Coplien, J., O. (1998). Software design patterns: common questions and answers. The patterns handbooks: techniques, strategies, and applications, Cambridge University Press**:** 311-319.

Detmer, R., et al. (2010). Incorporating real-world projects in teaching computer science courses. Proceedings of the 48th Annual Southeast Regional Conference, ACM.

Distasio, J. and T. Way (2007). Inclusive computer science education using a ready-made computer game framework. Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education. Dundee, Scotland, ACM.

Eagle, M. and T. Barnes (2009). Experimental evaluation of an educational game for improved learning in introductory computing. ACM SIGCSE Bulletin, ACM.

El-Nasr, M. S. and B. K. Smith (2006). "Learning through game modding." Computer Entertainment **4**(1): 7.

Elfes, A. (1990). Sonar-based real-world mapping and navigation. Autonomous robot vehicles, Springer-Verlag New York, Inc.**:** 233-249.

Estey, A., et al. (2010). Investigating studio-based learning in a course on game design. Proceedings of the Fifth International Conference on the Foundations of Digital Games, ACM.

Fesakis, G. and K. Serafeim (2009). Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education. ACM SIGCSE Bulletin, ACM.

Foss, B. A. and T. I. Eikaas (2006). "Game play in Engineering Education - Concept and Experimental Results." The International Journal of Engineering Education **22**(5).

Garrido, A., et al. (2009). Using graphics: motivating students in a C++ programming introductory course. EAEEIE Annual Conference, 2009, IEEE.

Goldberg, D. S., et al. (2012). Engaging computer science in traditional education: the ECSITE project. Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education, ACM.

Goldweber, M., et al. (2013). "A framework for enhancing the social good in computing education: A values approach." ACM Inroads **4**(1): 58-79.

Greenberg, I., et al. (2012). Creative coding and visual portfolios for CS1. Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM.

Higgins, J. P. and S. Green (2008). Front Matter. New York, NY, USA, John Wiley & Sons.

Honig, W. L. and T. Prasad (2007). A classroom outsourcing experience for software engineering learning. ACM SIGCSE Bulletin, ACM.

Huang, C.-H., et al. (2008). Computer game programming course for art design students by using flash software. Cyberworlds, 2008 International Conference on, IEEE.

Hundhausen, C., et al. (2010). Does studio-based instruction work in CS 1?: an empirical comparison with a traditional approach. Proceedings of the 41st ACM technical symposium on Computer science education, ACM.

Jenkins, J., et al. (2012). Perspectives on active learning and collaboration: JavaWIDE in the classroom. Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM.

Jiau, H. C., et al. (2009). "Enhancing self-motivation in learning programming using game-based simulation and metrics." Education, IEEE Transactions on **52**(4): 555-562.

Jun, H. (2010). Improving undergraduates' teamwork skills by adapting project-based learning methodology. Computer Science and Education (ICCSE), 2010 5th International Conference on, IEEE.

Kazman, R., et al. (1998). "The Architecture Tradeoff Analysis Method." Fourth IEEE International Conference on Engineering Complex Computer Systems.

Khan, K. S., et al. (2001). Undertaking systematic reviews of research on effectiveness: CRD's guidance for carrying out or commissioning reviews, NHS Centre for Reviews and Dissemination.

Kruchten, P. (1995). "The 4+1 View Model of Architecture." IEEE Softw. **12**(6): 42-50.

Kurkovsky, S. (2009). Can mobile game development foster student interest in computer science? Games Innovations Conference, 2009. ICE-GIC 2009. International IEEE Consumer Electronics Society's, IEEE.

Lawrence, R. (2004). "Teaching data structures using competitive games." IEEE Transaction on Education **47**(4): 459-466.

Lumia, R., et al. (1990). "The NASREM Robot Control System and Testbed." International Journal of Robotics and Automation **5**: 20-26.

Mahoney, W. and R. A. Gandhi (2012). "Reverse engineering: is it art?" ACM Inroads **3**(1): 56-61.

Maier, M. W., et al. (2004). "ANSI/IEEE 1471 and systems engineering." Systems Engineering **7**(3): 257-270.

McGovern, A. and J. Fager (2007). Creating significant learning experiences in introductory artificial intelligence. Proceedings of the 38th SIGCSE technical symposium on Computer science education. Covington, Kentucky, USA, ACM.

Natvig, L., et al. (2004). "Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals." Proceedings of the 2004 Frontiers in Education Conference.

Navarro, E. O. and A. v. d. Hoek (2004). SimSE: an educational simulation game for teaching the Software engineering process. Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education. Leeds, United Kingdom, ACM.

Nordio, M., et al. (2011). Teaching software engineering using globally distributed projects: the DOSE course. Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development, ACM.

Owston, R., et al. (2007). Computer Game Development as a Literacy Activity, Institute for Research on Learning Technologies, York University.

Rankin, Y., et al. (2008). The impact of game design on students' interest in CS. Proceedings of the 3rd international conference on Game development in computer science education. Miami, Florida, ACM.

Rick, D., et al. (2012). Bringing contexts into the classroom: a design-based approach. Proceedings of the 7th Workshop in Primary and Secondary Computing Education, ACM.

Ritzhaupt, A. D. (2009). "Creating a game development course with limited resources: An evaluation study." ACM Transactions on Computing Education (TOCE) **9**(1): 3.

Robertson, J. and C. Howells (2008). "Computer game design: Opportunities for successful learning." Computers & Education **50**(2): 559-578.

Rodger, S., et al. (2012). Integrating computing into middle school disciplines through projects. Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM.

Rollings, A. and D. Morris (2004). Game Architecture and Design - A New Edition, New Riders Publishing.

Rosas, R., et al. (2003). "Beyond Nintendo: design and assessment of educational video games for first and second grade students." Computer Education **40**(1): 71-94.

Ryoo, J., et al. (2008). Teaching Object-Oriented Software Engineering through Problem-Based Learning in the Context of Game Design. Proceedings of the 2008 21st Conference on Software Engineering Education and Training - Volume 00, IEEE Computer Society.

Schild, J., et al. (2010). ABC-Sprints: adapting Scrum to academic game development courses. Proceedings of the Fifth International Conference on the Foundations of Digital Games, ACM.

Seaborn, K., et al. (2012). Programming, PWNed: using digital game development to enhance learners' competency and self-efficacy in a high school computing science course. Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM.

Sharples, M. (2000). "The design of personal mobile technologies for lifelong learning." Comput. Educ. **34**(3-4): 177-193.

Simmons, R. (1992). "Concurrent Planning and Execution for Autonomous Robots " IEEE Control Systems **1**: 46-50.

Sindre, G., et al. (2009). "Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals." IEEE Transaction on Education **52**(1): 10-18.

Smith, J., et al. (2012). Using peer review to teach software testing. Proceedings of the ninth annual international conference on International computing education research, ACM.

Stuurman, S., et al. (2012). A new method for sustainable development of Open Educational Resources. Proceedings of Second Computer Science Education Research Conference, ACM.

Sullivan, A. and G. Smith (2011). Lessons in teaching game design. Proceedings of the 6th International Conference on Foundations of Digital Games, ACM.

Sung, K., et al. (2011). "Game-themed programming assignment modules: A pathway for gradual integration of gaming context into existing introductory programming courses." Education, IEEE Transactions on **54**(3): 416-427.

Tan, P.-H., et al. (2009). Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception. Computer Technology and Development, 2009. ICCTD'09. International Conference on, IEEE.

Timm, I. J., et al. (2008). Teaching distributed artificial intelligence with RoboRally. Multiagent System Technologies, Springer**:** 171-182.

Toal, D., et al. (2005). "Subsumption architecture for the control of robots." ACM SIGCSE Bulletin **37**(3): 138-142.

van Delden, S. (2010). "Industrial robotic game playing: an AI course." Journal of Computing Sciences in Colleges **25**(3): 134-142.

van Langeveld, M. C. and R. Kessler (2009). Two in the middle: digital character production and machinima courses. ACM SIGCSE Bulletin, ACM.

Wang, A. I. (2009). "An Extensive Evaluation of Using a Game Project in a Software Architecture Course." Submitted to Transaction on Computing Education (ACM).

Wang, A. I. (2011). "Extensive Evaluation of Using a Game Project in a Software Architecture Course." Trans. Comput. Educ. **11**(1): 1-28.

Wang, A. I. and B. Wu (2009). "An Application of Game Development Framework in Higher Education." International Journal of Computer Games Technology, Special Issue on Game Technology for Training and Education **2009**.

Wang, A. I. and B. Wu (2011). "Using Game Development to Teach Software Architecture." International Journal of Computer Games Technology **2011**.

Wang, K., et al. (2006). 3D game design with programming blocks in StarLogo TNG. Proceedings of the 7th international conference on Learning sciences, International Society of the Learning Sciences.

Webb, H. C. and M. B. Rosson (2011). Exploring careers while learning Alice 3D: a summer camp for middle school girls. Proceedings of the 42nd ACM technical symposium on Computer science education, ACM.

Weng, J.-F., et al. (2010). "Teaching boolean logic through game Rule tuning." Learning Technologies, IEEE Transactions on **3**(4): 319-328.

Werner, L., et al. (2009). Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study. Proceedings of the 4th International Conference on Foundations of Digital Games, ACM.

Williams, L. and T. Beaubouef (2012). "Comparing learning approaches: sample case studies." Journal of Computing Sciences in Colleges **27**(5): 85-91.

Wu, B. and A. I. Wang (2012). "Comparison of learning software architecture by developing social applications versus games on the Android platform." International Journal of Computer Games Technology **2012**: 5.

Wu, B., et al. (2010). Extending Google Android's Application as an Educational Tool. Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2010 Third IEEE International Conference on, IEEE.

Wu, B., et al. (2009). XQUEST used in software architecture education. Games Innovations Conference, 2009. ICE-GIC 2009. International IEEE Consumer Electronics Society's, IEEE.

Wu, B. and I. Wang (2011). Game development frameworks for SE education. Games Innovation Conference (IGIC), 2011 IEEE International, IEEE.

Wynters, E. L. (2007). "3D video games: no programming required." Journal of Computing Sciences in Colleges **22**(3): 105-111.