Playing a Game: The Ecology of Designing, Building and Testing Games as Educational Activities

Alexander Repenning and Clayton Lewis
University of Colorado at Boulder
Boulder, CO, USA
[ralex, clayton]@cs.colorado.edu

Abstract: The design and implementation of educational games can be highly motivational to undergraduate students. In many cases it allows them to build the kind of computational artifacts that they envisioned building when they entered a computer science program. Additionally, the design and implementation of games is demanding, as it requires to master a variety of skills and to combine them in a context that typically includes collaborative and interdisciplinary work stiles. Initially, computer science programs did not welcome the notion of game design, as they perceived games as a non-serious application of computer science principles. With the game industry growing at an enormous rate and the complexity of the games clearly reaching a level of complexity approaching, and in many cases exceeding, the level of most "serious" computer science applications the evidence has reached critical mass indicating that games have become computer science showcases. At the same time there is also increasing evidence that games can have high educational potential. Rich simulations, for instance, promise to engage learners in activities in ways not previously possible with traditional media such as books and even electronic media such as movies. Our goal was to combine these two directions by offering courses on game design for education. The main point of this paper is to share our experience over three iterations of this course.

Introduction

The University of Colorado computer science department has been offering an educational game design course in 2005 for the third time. Before the Trails project the idea of game design was explored in the context of more traditional courses such as the Object-Oriented Design (OOD) course and topic courses. The OOD course is popular with undergraduate computer science students. However, the OOD course is considered challenging to teach since student's expectations vary widely. Some prefer a highly theoretical treatment while others prefer to gain project-oriented hands on experience. In spring of 2002 the OOD course employed the idea of games as means to bridge the gap between theoretical and concrete treatments of design and implementation issues. Using software engineering approaches including UML diagrams and product description of classic arcade games students had to design and implement simple versions of games including Space Invaders, Sokoban, Pacman and the Sims. This experience made clear that the popularity of games could be usefully combined with educational goals.

The positive experience with using games in a computer science course let to the game inspired version of the Trails course at the University of Colorado. In the spirit of the Trails framework regarding project-oriented, collaborative and interdisciplinary approaches a course called games4education recruited not only students from the computer science department but also from education and fine arts. Currently in its third incarnation this course is trying to balance ideas of game design with education. Students are exposed to topics ranging from a design, implementation and testing to the theory of learning. Initially, students work on several smaller projects introducing them to the mechanics of games and play testing. In the second part of the course they need to present a project proposal. The proposal has to address issues of learning as well as engagement. Students would have to make field trips to schools to play test their prototypes and final products with K-12 students. Final assessment of the projects included a number of criteria including educational value and potential for engagement.

A significant challenge to undergraduate computer science education is the general lack of authentic, project-oriented courses involving teams of students to produce non-trivial artifacts. Early curriculum focuses on highly isolated skills typically reducing the notion of a project to throw away programs resulting from individually implemented textbook algorithms. Team-based projects of open-ended problems are only found at senior levels. Quite often these senior projects result in negative "educational" experiences conveying how not to organize a project. This frustration stems from the previous lack of teamwork experience and exposure to projects of significant complexity without a well-known solution.

Games are leading edge computational artifacts fueled by an explosively expanding industry. They not only integrate virtually all computer science topics but also are highly motivational. A large percentage of computer science students mention their interest in games as one of the main reasons to have entered the computer science field. The "Gamelets for Education" course is concerned with the acquisition of computer science related design and implementation skills by having undergraduates build educational games for students at various educational levels. There is a need to understand and to support a complex ecology involving instructors, university students, K-12 students and external design knowledge in order to be able to successfully build these games in a relatively short amount of time. In this paper we describe some of the challenges we faced.

Game Design for Education Challenges

The main challenges we experienced with game design in education include the need to balance engagement versus learning, to provide diverse background information, to prepare university students to interact with K-12 teacher and students and finally to select appropriate tools.

Educational Game design needs to carefully balance engagement and learning

We have found that it is extremely difficult to achieve a good balance between engagement and learning. Figure 1 (below) illustrates the balance between engagement and learning as a continuum. It is not completely clear where exactly well working educational games need to be positioned in this continuum. A game can be tremendously engaging without even the slightest hint of learning. On the other hand a game may offer excellent learning opportunities but simply not be much fun to play at all. Certainly the extreme end points need to be avoided but a good balance does not necessarily imply that a game should be exactly in the middle of the continuum.

Figure 1. The Engagement / Learning Continuum. Educational Games should balance engagement and learning by combining game design with backward design.

The fundamental design questions are not limited to the learning versus engagement continuum but also the need to address the type as well as the quality of the connection between learning and engagement. As we have seen with the highly controversial concept of *edutainment* the mere juxtaposition of learning and engagement is not likely to produce compelling content (Brown, 1995). Imagine, for instance, a Pacman-type game in which a user cannot simply pickup powerpils to defeat attacking ghost but instead needs to solve double digit addition math

problems. In this example the learning goals are about elementary school math whereas the context of the Pacman game is employed as engaging activity. There is no intrinsic connection between the two. A lack of a *meaningful* connection between learning and engagement can easily result in designs that are neither well suited for learning nor fun to use. The blurred line between learning and engagement of Figure 1 is used to symbolize the need for a progression.

A second philosophical consideration regarding the balance of learning and engagement is the design process used to reach meaningful connections. In other words the continuum is not only a space containing individual points but also a space that can be used to capture design processes as transitions from one point to another. A design process may start at one point in the continuum but gradually migrate towards a different point. The processes that we have witnessed in the Trails project fall into two different design philosophy categories:

1) ***Educational Design*** (Learning → Engagement): Educational design's main objective is learning. This design process starts with learning but gradually adds elements of engagement. A popular design approach used in education is *backward design* (Wiggins and McTighe, 2000). The backward design process starts with the identification of learning goals, then designs assessments and, finally, plans activities in support of the learning goals. To reach some degree of engagement these activities may be implemented as games.
2) ***Game Design*** (Engagement → Learning) Game design is highly focused on motivational aspects such as engagement and fun (Koster, 2004). Most games have clever scaffolding mechanisms built in (Gee, 2004) allowing their users to gradually solve more complex problems. However, these mechanisms are typically used to learn about using the game and not about some educational topic. Most game design approaches are highly user centered and iterative. Potential game users are exposed to design prototypes early and often in order for developers to gain insights into the degree of engagement actually achieved by a game. The reality is that even the most experienced game designers have to learn to mistrust their own intuitions and replace introspection with user testing.

At this point we should admit that we do not completely comprehend the notion of meaningful connections between engagement and learning. Reviewing a number of commercial games we found that sometimes neither game design nor education design theories would be good predictors for the engagement and educational effectiveness of a game. One such game is called the Typing of the Dead (Park, 2001). The Typing of the Dead is about learning to touch type. An unending stream of blood soaked zombies is constantly attacking the player. Each zombie holds a sheet with a small piece of text. The player has to use the keyboard to match that text as fast as possible in order to make the zombie go away. As the player progresses in terms of typing skills, the text will become more complex and the zombies can also increase their attack speed. Most of the Trails members' personal evaluation of this game was extremely polarized. Either people concluded that this was an example of the worst possible educational game or of the best possible educational game. On the surface there is no obvious meaningful connection between engagement and learning in this game. The engagement is based on the drama of attacking zombies. The learning is about touch-typing. At a cognitive and ontological level there is no apparent connection between zombies and touch-typing. However, at a procedural learning level we find that perhaps the nature of learning how to type, which is highly connected to subconscious motor skills, is indeed closely and meaningfully connected to a game context such as the Typing of the Dead. The ability of the game to adaptively balance typing challenges and skills through a game context is related to the notion of flow (Csikszentmihalyi, 1991).

Providing diverse background information

Instructors will have to cope with the fact that students are likely to be more knowledgeable about certain aspects of game design including the knowledge about the most current versions of commercial games. The variety and complexity of game creation tools is too large for most instructors to handle. The requirement to build educational games makes it necessary to provide sufficient educational background to be able to create educational content that is relevant to K-12 curriculum.

We underestimated the need for good background materials in earlier versions of the course. Also, we overestimated our students' ability to pick up the needed concepts from readings in the literature. In our latest iteration we wrote a set of notes that (we hoped) would provide concrete guidance to our students on learning theory, its application to analysis and design of games, and the "theory of fun", and its application. Especially in the latter area we felt we were breaking new ground, in that (we feel) there are no suitable presentations of what makes games fun that are concretely applicable to educational game design.

Even with these notes, we feel the problem is at best only partly solved. We based our notes on design for learning on Anderson's ACT-R framework (Anderson), which distinguishes procedural and declarative knowledge. Making this distinction in the context of the learning objectives for games proved problematic. One could argue that this is an unavoidable problem, since people can and do shift knowledge across these forms, but the resulting confusion and uncertainty is an obstacle to making learning analysis adequately concrete in the context of design.

A further problem is that we didn't provide enough examples of worked-out analyses to supplement the presentation in the notes. Adding examples would help set the right expectations for our students, some of whom were inclined to skimp on analysis in favor of "just designing" or even "just coding".

Preparing CS students to interact with K12 students and teachers

Collaboration between university and K-12 students needs to be heavily scaffolded since few university students have any experience in user centered design approaches involving actual contact with users.

Ideally we could have a course setup in a way that it would allow in-service teachers to take the course as well. There are a number of problems with getting teachers to enroll into such course. One problem is simply the time of the course. Most teachers will require a course to be scheduled after four pm to avoid interference with their teaching schedules. An even bigger problem is the lack, in most states, for the need of information technology (IT) certification. Teachers who are planning to use technology peripherally or as main teaching subjects are not required to get any kind of IT certification. With their busy schedules this means that in order to fulfill their pre- and in-teaching education requirement they are not likely to have the time, motivation or resources to attend tuition-based university courses. At the political level, on a long-term scale, this will need to be changed by adding an IT component to teaching certification. In the short term, however, this means that teachers need to be financially supported to take these kinds of courses.

Tool Complexity

Over the years we have observed different outcomes based on the kind of tools students have used. One point that is essential is to include some expectation management. This management will put the use of a certain technology into perspectives including the explanation of trade offs between game design, game complexity and learning design.

1) ***3D from Scratch***. Especially to computer science students there is a natural affinity to use high end programming tools. 3D games with complex rendering are often considered the holy

grail of engagement. Since many CS students also feel that learning essentially emerges from engagement they are compelled to spend most of their efforts on the engagement side. Unfortunately, creating a 3D application is by no means a simple endeavor. A course limited to one semester simply does not provide the time necessary to prepare students in the use of relatively low-level application programming interfaces (APIs) such as OpenGL or Direct3D, and to provide them the necessary background in game design and learning design. As a result, with the exception of perhaps the best groups of students including individuals with previous experience, students are likely to face a frustrating time. Student will have high expectation, which simply cannot be matched because of their lack of experience and the mere lack of time.

2) ***3D with Game Engines***. Game Engines are software packages that will substantially leverage the design and implementation of 2D/3D games. In some sense they can be considered a middleware layer between the low-level 3D APIs and the game application. Modern game engines are highly optimized towards the kinds of games they are used for. Most of the popular game engines are produced by the company that builds and sells the games based on that engine. On the one hand this provides a great opportunity for creating truly sophisticated games reaching a level of quality only found in commercial games. On the other hand most of these game engines are tricked out to the point where they are highly specialized towards a certain kind of game. For instance, attempts to use a game engine in a non-first/third person mode may result in serious conceptual as well as technical problems. Additionally, these engines are not well suited for prototyping for a number of reasons. The learning curve can be extremely steep. Most game engines are very large, and often poorly documented pieces of software. In addition to programming challenges game engines also pose artistic challenges because having seen games created with that game engine students feel obliged to create high-end 3D models. Students feel compelled to have cool models even for early versions of their games. Unfortunately, the creation of these models can take a lot of time because they need to be created with complex 3D modeling tools, which also have a steep learning curve. As a result a lot of time that should have been used to explore a solid game concept is misused with premature high-end content creation.

3) ***Gamelets***: Simple Web-Based Game Building Tools. In lieu of the likely complexity emerging from 3D games we have explored the notion of so called Gamelets as simple versions of games. A Gamelet has a complexity comparable to classic arcade games such as Pacman. Our goal was to cap complexity so that students, by using the AgentSheets simulation-authoring tool or the Flash Web animation-programming tool could build early prototypes of games in a matter of minutes and hours instead of days. The tools we used allowed the students to shift much of their time from implementation to design. Students could quickly produce prototypes that were sufficiently concrete so that users could meaningfully react to them. This kind of interaction was important to facilitate an iterative design/implementation approach.

In hindsight we liked the Gamelets approach the best as it provided the best means to iteratively explore a constructive balance between engagement and learning.

Conclusions

Educational game design has some very attractive features as a focus for projects in computer science: technical scope (in design, implementation, and evaluation), a high level of student interest, and (for some students) the appeal of service learning. But our experience has been that

is quite hard to exploit these advantages fully, for reasons we have tried to articulate here. We feel that our course development is very much an iterative process: each offering of the course gets better. We hope our experiences may help others home in more quickly on the target, which is well worth hitting.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0205625. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Wiggins, G., & McTighe, J. (2000). Understanding by Design. Prentice Hall.

Csikszentmihalyi, M. (1991). Flow: The Psychology of Optimal Experience. Perennial.

Brown, E. (1995). That's Edutainment: A Parent's Guide to Educational Software/Book and Cd-Rom. Mcgraw-Hill Osborne Media.

Gee, J. P. (2004). What Video Games Have to Teach Us About Learning and Literacy. Palgrave Macmillan.

Anderson, J. R., D. Bothell, et al. An Integrated Theory of the Mind. Psychological Review, 111(4), 1036-60.

Park, A. (2001). review of The Typing of the Dead. CNET reviews.

Koster, R. (2004). Theory of Fun for Game Design. Paraglyph.