10-2008

An Exploratory Overview of Teaching Computer Game Development

Mario Guimaraes
Kennesaw State University, mguimara@kennesaw.edu

Meg C. Murray
Kennesaw State University, mcmurray@kennesaw.edu

AN EXPLORATORY OVERVIEW OF TEACHING COMPUTER GAME DEVELOPMENT*

Mario Guimaraes and Meg Murray
Department of Computer Science and Information Systems
Kennesaw State University
1000 Chastain Road
Kennesaw, GA 30144
(770) 420-4424
mguimara@kennesaw.edu, mcmurray@kennesaw.edu

ABSTRACT

The computer game industry has exploded reaching sales of several billion dollars a year and, consequently, a majority of college students are familiar with the gaming environment. In fact, videogame development has been cited as one way to motivate students to explore the world of Computer Science. However, most videogames are extremely complex computer programs created by a team of developers including programmers and graphic artists and represent thousands of hours of work. Fortunately there are software tools available that provide a way for simple computer games to be created fairly easily using a building block approach. This paper discusses the successes and challenges of teaching a videogame design and development summer program using the software development tool, *Game Maker*, and from this experience examines how videogame development might be incorporated into a Computer Science curriculum. The first section provides an overview of the *Game Maker* program and outlines the material taught in the program. Observations of the most successful teaching methods and approaches utilized are also explored. We conclude with a discussion of where videogame design might best be suited in a Computer Science curriculum citing its attractiveness to non-Computer Science majors, its use as a way to introduce introductory programming concepts and as a way to help students learn to read code. While *Game Maker* is not sophisticated nor is it a substitute for teaching a standard

programming language, it can be easily integrated into introductory Computer Science courses.

INTRODUCTION

Engaging students in the study of any academic discipline has challenges. In the area of Computer Science, those challenges have been compounded by a perception of outsourced career opportunities and uninteresting curriculum. At the same time, the computer and video game industry continues to experience massive growth. In the US alone, sales in 2007 grew 43% to almost $18 billion [7]. The majority of college students today are exposed to videogames even if they are not avid players. Drawing on this familiarity has appeal for incorporating videogames into a Computer Science curriculum. It has generally been shown that familiarity with a topic heightens interest, provides a frame of reference from which learning can take place and helps to build a sense of purpose in learning and applying material. Computer/video gaming is one area that has wide appeal and also provides an application area that is engaging and challenging [1].

Computer/video games can be extremely complex computer programs. It may take a team of computer programmers and graphic artists thousands of hours to create a unique, enjoyable game. However, there are tools available that provide a much simpler approach to creating games. These tools allow less experienced computer enthusiasts to create their own worlds and story lines. *Game Maker* is one of these tools. It is an integrated development environment (IDE) that includes a drag and drop interface through which games can be constructed quickly without the need to write program code. The minimum steps necessary to create a program include creating an image (called a sprite), an object and a room. Objects are associated with sprites and with event-actions. Objects are made active by placing them in a room. In addition to sprites, objects and rooms, *Game Maker* resources include sounds, backgrounds, paths, fonts and timelines for enhancing program actions. Figure 1 shows the *Game Maker* IDE interface.

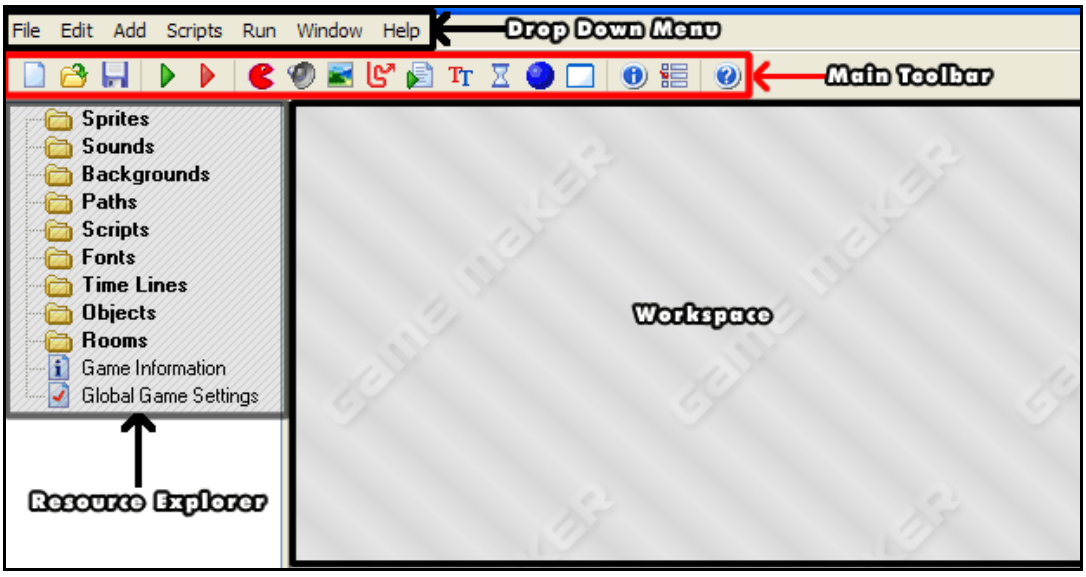

Figure 1. *Game Maker* IDE Environment

Figure 2 depicts the association of an object, in this example, 'player,' with a sprite and the association of the player object with a series of events and corresponding actions. In this example, when the event -- key-down --, is pressed, the player object will perform three actions, move (jump to a given position), change the image (change the sprite into player_down) and change direction (set variable direction to 270).

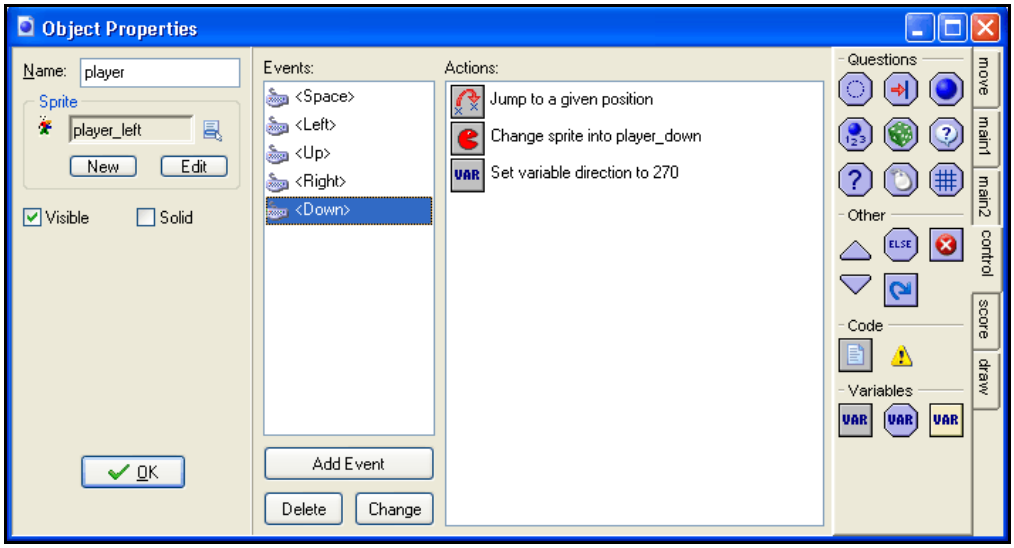

Figure 2. Example Association of Objects with Sprites and Actions

EXPERIENCE TEACHING VIDEOGAME DESIGN

Our initial experience with *Game Maker* was during the summer of 2007 when we hosted six videogame camps for high school and junior high school students. The camps were offered through the continuing education unit of Kennesaw State University. The camps ran for four days, Monday through Thursday from 9 am to 4 pm. Course enrollment was eighteen students. While the camp was open to students in grades 6-12, no junior or senior high school students enrolled.

An analysis of each camp was conducted at its conclusion and findings were used to make changes to subsequent camps. For instance, the intensity of the instruction was increased and student presentations of their game creations were added on the last day of the camp. Through the analyses, several strategies that contributed to student engagement and learning were identified. These included:

- It is beneficial for students to play an example game, modify that game and then create their own game implementing the features exemplified in the example game.
- It is important to properly set expectations for what students can achieve in such a short time.
- It is important to provide time for students to work independently allowing them to experiment on their own and develop their own individual learning processes.
- Requiring student presentations of their work provides motivation for students to achieve at higher levels.

Even within the limited time available in the videogame camps, it was possible to cover a wide range of topics with students achieving a high level of success. An outline of the flow of the camps and the order of topics covered is presented in Table 1.

ADVANTAGES OBSERVED USING *GAME MAKER*

There is much research that states programming, and especially OO concepts, are not easy to teach [9]. Further, a disconnect often exists between student perceptions of computer programming and the reality behind what it takes to build programming skill. One approach to address this gap is to teach students how to learn classes first before they learn to create them [5]. *Game Maker* provides an environment conducive to introducing students to programming through an object first approach and with an environment familiar to students. *Game Maker* is extremely intuitive and easy to use and allows students to construct programs without writing code but yet learning to use objects and apply logic. Further, *Game Maker* provides a well defined, concise and powerful set of tools and primitives. This restricted environment allows beginning programming students to retain focus, whereas other more highly-featured environments provide a multitude of options that can be overwhelming to a novice.

	Topics	**Activities**
Day 1		
	Exploring example games [Shooter, Maze] examining concepts including keyboard control, mouse control, moving between rooms, including multiple levels of play	Playing and modifying the first game
	Identifying and using basic game components including sprites, objects, and room and available action-events	Manipulating screen coordinates
	Understanding screen coordinates	
Day 2		
	Fundamentals of videogame design specification	Development and sharing of design specifications
	Understanding the concept of Inheritance	
Day 3		
	Identifying and using advanced components including gravity, friction, paths and timeline	Implementation of game design specifications through game creation and development
	Exploring the basic principals of animation	
Day 4		
	Changing objects	Student game presentations
	Understanding control structures	

Table 1: Videogame Camp Topics and Activities

Another advantage to *Game Maker* is that it includes a set of interesting sample programs. This provides the opportunity for students to practice reading and modifying code before they engage in any code creation. The well-known software developer, Grady Booch, states that one of the major problems in Computer Science curricula is that students are often first directed to build programs before they are taught how to read code. As an analogy, he notes that literature students are expected to read many famous poems before they are requested to write one. Computer science students, on the other hand, are rarely exposed to good programs [3].

Finally, an important observation made using the *Game Maker* program is that designing videogames was attractive and motivating to the students. Using videogames as a motivating factor in programming is one of the most often cited reasons for their inclusion in the curriculum [1, 2, 5, 8, 9]. One of the major criticisms for using videogames is that gaming is not attractive to women. While enrollment in the videogame camps was 90% male, the females who did attend were as successful as their male counterparts. As [5] points out, "Contrary to common misperceptions, women make up 45% of all game players." Even though women may not be as highly associated with videogaming as men, they are familiar with the videogame environment and this familiarity provides a common knowledge base from which the teaching of programming concepts can be made [1]. Videogame use continues to increase and this "huge interest" in games makes the incorporation of gaming into programming courses compelling [5].

GAME MAKER IN THE COMPUTER SCIENCE CURRICULUM

The incorporation of videogame development into the Computer Science curriculum is growing [1, 6]. The International Game Developers Association has even developed a framework curriculum for game development education [4]. There are several reasons why gaming is attractive. Games, by their very nature, are "complex pieces of software that combine advanced concepts from a wide variety of areas within CS." [2] Games can be simple, such as the typical arcade game, or they can represent complex simulations. This continuum of knowledge and skill provides a rich environment for instruction. According to [5], they use a games first approach because it provides a motivating environment that allows them to maintain a high level of technical depth.

There are different approaches used in videogame curriculum. They span from a full-fledged degree program in videogame development with the intent of graduating students ready to enter the professional videogame market to using gaming as a theme for programming assignments or as a way to attract non-majors. *Game Maker* is most well suited for attracting students and providing an exploratory overview of computer programming and complex Computer Science concepts. While *Game Maker* allows students to create enjoyable games, it is not suited for creating professional games that incorporate sophisticated features such as 3-D graphics and multi-player capability. It does, however, include an easy to use interface and comes with a scripting language; albeit not a standard one. From a teaching standpoint, *Game Maker* provides the opportunity for students to engage and experiment with basic CS concepts in a motivating environment that provides immediate feedback. From a course perspective, *Game Maker* incorporates well into a CS0 level class or as an introductory exercise in a CS1 course.

CONCLUSION

Research has shown that understanding complex and abstract concepts such as object-orientation and algorithmic thought cannot be done passively [9]. Students must actively engage with these concepts through design, development, implementation, and practice. Further, concepts are not easily learned when isolated from their application. When combined, students see the point in their learning [1, 9]. *Game Maker* provides a means to achieve these objectives at an introductory level through which foundational learning can be initiated. As [9] points out, "Designing and building a game, can therefore, be a very useful way towards a more thorough understanding of the algorithms, data structures and other topics in Computer Science."

REFERENCES

[1] Bayliss, J. D. and Strout, S., Games as a "flavor" of CS1, *SIGCSE Bulletin*, 38, (1), 500-504, 2006.

[2] Becker, K. and Parker, J. R.. Serious games + Computer Science = serious CS. *Journal of Computing in Small Colleges*, 23, (2), 40-46, 2007.

[3] Booch, G., Readn', writ'n, 'rithmetic..and code'n. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, USA, March 07 - 11, 2007). SIGCSE '07. ACM, New York, NY, 197-197, 2007.

[4] International Game Developers Association Education Committee, IGDA Curriculum Framework Version 2.3 beta, 2003, http://www.igda.org/academia/curriculum_framework.php, retrieved May 1, 2008.

[5] Leutenegger, S. and Edgington, J., A games first approach to teaching introductory programming. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, USA, March 07 - 11, 2007), 115-118, 2007.

[6] Masuch, M. and Nacke, L., Power and peril of teaching game programming, 2004, http://www.cs.uni-magdeburg.de/~nacke/Masuch-Nacke-PowerPeril.pdf, retrieved May 1, 2008.

[7] NPD Group, 2007 U.S. video game and PC game sales exceed $18.8 billion marking third consecutive year of record-breaking sales, 2008, http://reborn2603.wordpress.com/2008/05/08/video-game-industry/, retrieved May 1, 2008.

[8] Overmars, M., Teaching Computer Science through game design, *Computer*, 37, (4), 81-83, 2004

[9] Rai, S., Wai Wong, K., and Cole, P., Game construction as a learning tool, *ACM International Conference Proceeding Series*, 23, 231-236, 2006.

[10] Yo Yo Games, *Game Maker*, 2008, retrieved May 1, 2008 from http://www.yoyogames.com/make.