Computers & Education 55 (2010) 1098–1111

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Computer-game construction: A gender-neutral attractor to Computing Science

Mike Carbonaro [a], Duane Szafron [b,*], Maria Cutumisu [b], Jonathan Schaeffer [b]

[a] Faculty of Education, University of Alberta, Edmonton, AB, Canada T6G 2E8
[b] Department of Computing Science, University of Alberta, Edmonton, AB, Canada T6G 2E8

ARTICLE INFO

Article history:
Received 2 October 2009
Received in revised form
15 April 2010
Accepted 13 May 2010

Keywords:
Computing Science
Females in Science
Computer game construction

ABSTRACT

Enrollment in Computing Science university programs is at a dangerously low level. A major reason for this is the general lack of interest in Computing Science by females. In this paper, we discuss our experience with using a computer game construction environment as a vehicle to encourage female participation in Computing Science. Experiments with game construction in grade 10 English classes showed that females enjoyed this activity as much as males and were just as successful. In this paper, we argue that: a) computer game construction is a viable activity for teaching higher-order thinking skills that are essential for Science; b) computer game construction that involves scripting teaches valuable Computing Science abstraction skills; c) this activity is an enjoyable introduction to Computing Science; and d) outcome measures for this activity are not male-dominated in any of the three aspects (higher-order thinking, Computing Science abstraction skills, activity enjoyment). Therefore, we claim that this approach is a viable gender-neutral approach to teaching Computing Science in particular and Science in general that may increase female participation in the discipline.

1. Introduction

The March 23rd, 2006 edition of the Economist featured an article called "Computing and the Scientific Method" in which an interesting question was asked and answered:

> WHAT makes a scientific revolution? Thomas Kuhn famously described it as a "paradigm shift" – the change that takes place when one idea is overtaken by another, usually through the replacement over time of the generation of scientists who adhered to an old idea with another that cleaves to a new one. … Some 34 of the world's leading biologists, physicists, chemists, Earth scientists and computer scientists … have spent the past eight months trying to understand how future developments in Computing Science might influence science as a whole. They have concluded, in a report called "Towards 2020 Science", that computing no longer merely helps scientists with their work. Instead, its concepts, tools, and theorems have become integrated into the fabric of science itself. (Economist.com).

Computing Science is playing an increasingly important role across all scientific disciplines, to the point that advancements in Science are inherently linked to advancements in computational representations and functional implementation. One only needs to examine the human genome project to see the extensive role that computational techniques continue to play in helping us to understand our genetic code. Diverse subjects such as space travel, environmental modeling, cell phones, and online social networking all depend on the concepts, tools, and theorems linked to research in Computing Science.

Predictions by the U.S. Department of Labor Statistics (U.S. Department of Labor, 2007) indicate that information technology is the fastest growing economic sector, with a 68% increase in output growth rate projected for the 2002–2012 decade. Unfortunately, since the dot.com bubble burst in 2000–2001, Computing Science enrollment at universities and colleges has decreased by 50–70% (www.cra.org). In part, the sharp decline has been attributed to the fear that Computing Science is no longer a viable career path. This is ironic in the light of the increased dependence of Science on computing.

Stanford Professor Eric Roberts pointed out why the declining interest in Computing Science has serious implications beyond that of the computing industry and how it may severely hinder advances in Science (Roberts, 2007). For example, algorithms that model the movement

* Corresponding author. Tel.: +1 7804925468; fax: +1 7804921071.
E-mail address: duane@cs.ualberta.ca (D. Szafron).

0360-1315/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2010.05.007

of a robot's arm can also be used to assess whether a specific drug molecule can dock on a human protein and block the development of certain diseases. Roberts suggests the real barrier to generating interest in Computing Science is public education, especially at the K-8 level, where the teaching of Computing Science is rarely found (Cohoon & Aspray, 2006) and often deficient at the 9–12[1] level as well.

Although efforts are being made by many educational institutions to encourage students to study Computing Science at both the secondary and post-secondary level (www.csta.acm.org), few K-12 school teachers have formal training in the subject area and/or access to the necessary hardware/software (Goode, 2007). If students do take Computing Science in high school, it is often as an Advanced Placement option with restricted enrollment. Contrast this with other Science subjects like Mathematics, Biology, Chemistry, and Physics. For these subjects, schools have designed curricula and environments to support these disciplines and teachers have been formally trained in the appropriate domain content and pedagogy.

In addition to the lack of resources at the K-12 level for teaching Computing Science, there are other factors that impact students selecting this field of study. Carter (2006) surveyed 796 (363 male, 423 female) pre-calculus high-school students as to why individuals with an apparent aptitude for Computing Science did not select it as a major. The top reasons that both males and females cited for not selecting Computing Science were the lack of desire to sit in front of a computer screen all day and the fact that they had already selected a major. The number one reason for males selecting Computing Science was their experience with computer games, while for females it was their desire to use their Computing Science knowledge in another area of study. Overall only 36% of the males and 11% females surveyed indicated they were somewhat likely to select Computing Science as a major.

That few females are interested in Computing Science as a field of study should come as no surprise since traditionally there has always been a small percentage of undergraduate women in Computing Science (Camp, 1997; Singh, Allen, Scheckler, & Darlington, 2007). According to the 2006 Taulbee survey (www.cra.org), the percentage of Computing Science bachelor's degrees granted to women in the U.S. between 1994 and 2006 ranged between 14% and 18%, with 2005/06 having the lowest percentage at 14%. Such percentages may be expected, given differences in parental socializing behaviour. Parents often encourage males to engage in scientific thinking and activities (Crowley, Callanan, Tenenbaum, & Allen, 2001; Simpkins, Davis-Kean, & Eccles, 2005) more than they encourage females. In addition, girls' perceptions and experiences with Science are different from those of boys. For example, during the middle-school years, girls often begin to lose confidence in their ability to learn Science (Dreves & Jovanovic, 1998) and also suffer from gender stereotypes, such as 'physics is for boys' (Kessels, 2005).

One would expect this historical bias against females pursuing Science to result in low female interest in all major scientific disciplines. However, recent data indicates that this is no longer true. In 2004–2005, graduation levels across traditional scientific disciplines show the percentage of B.Sc. degrees granted to women in the U.S. were as follows: 62% in Biology, 51% in Chemistry, 46% in Mathematics, and 22% in Physics (National Center for Educational Statistics, nces.ed.gov). This data indicates that in some areas of Science, the female participation level has met or exceeded the male level. Only Physics B.Sc. graduation rates are close to the dismal showing found in Computing Science.

Why are Physics and Computing Science levels so low? High-school experience plays an important role in student selection of subject area interest at university (Goode, 2007). In the context of high-school Physics, there are a number of pedagogical approaches that create barriers to female participation and often negatively influence their performance (Hazari, Sadler, & Tai, 2008):

- Males are often allowed to dominate the classroom environment.
- Females must do unhelpful collaborative group work with males, especially in lab settings.
- The content is often specific to male interests.
- Traditional Physics instructional environments are typically isolated.

Research in Computing Science is less clear, but according to Goode, Estrella, and Margolis (2006): "We have witnessed female high-school students who want to be more deeply involved with technology, who know it is part of their future, who dream big, but who have no understanding of what deeper involvement entails, or what preparation is required." Barriers for female participation may come from an overall negative attitude towards using computers, lack of confidence with software and hardware tools, and a general misconception that work in Computing Science is conducted in isolation and is not relevant to society (Cohoon & Aspray, 2006).

There is strong evidence to suggest that gender is not a constraint in learning Mathematics and Science concepts. In a landmark publication, Hyde and Linn (2006) reported their meta-analysis on gender diversity and its role in selecting Mathematics and Science as career paths. Their research indicates that, regardless of gender, school-age children have similar cognitive abilities and psychological traits. They conclude that, "rather than focusing on gender differences, mathematics and science educators and researchers could more profitably examine ways to increase awareness of the similarities of performance and in ability to succeed." Shibley's and Linn's assessment provides important insight into the direction educators might take to encourage females' interest in disciplines like Physics and Computing Science.

1.1. Computing Science: curriculum, learning, and games

Female students often make decisions about a possible career in Science during their middle-school years (American Association of University Women Educational Foundation, 1996). Unfortunately, Computing Science has traditionally been introduced in high school with a formal instructional approach. Most introductions to Computing Science focus on perplexing/irrelevant topics and use languages such as C or C++ that are often perceived to be cryptic. Introductory assignments often involve sorting and merging lists of numbers or text, testing algorithms, and creating files. For example, the Computer Science Teachers Association Model Curriculum for K-12 Computer Science (CSTA, 2006) suggests the following lab assignments for the high school Computing Science programming component: methods (functions) and parameters, recursion, objects and classes (arrays, vectors, stacks, queues), graphics, and event-driven interactive programming. They also suggest introducing hardware and systems: logic, gates and circuits, binary arithmetic, assembly language, operating systems, user interface, and compliers.

[1] Unfortunately to many teachers/parents, Computing Science means the study of word processors, spreadsheets, presentation tools, and web page construction.

This traditional approach to understanding Computing Science is less than appealing to many students, and particularly to female students (Cohoon & Aspray, 2006). Over the past several years there has been a concerted effort on the part of educators to: a) understand the barriers female students encounter to studying Computing Science (Sing et al., 2007), and b) develop innovative curriculum models that promote female participation in Computing Science (Beyer, Rynes, Perrault, Hay, & Haller, 2003; Graham & Latulipe, 2003; Kurkovsky, 2007; Vilner & Zur, 2006). This paper investigates topics related to the second issue.

We contend that introducing computer game *design* and *construction* into the high-school curriculum is now a viable way of promoting female interest in Computing Science. Using the computer gaming environment as a vehicle to encourage female participation in Computing Science may appear counter-intuitive, given that game players are almost always boys (Cummings & Vandewater, 2007). That being said, there is a critical difference between playing games and *constructing* computer games with respect to motivating learners and facilitating problem-solving (Kafai, Heeter, Denner, & Sun, 2008; Kelleher, Pausch, & Kiesler, 2007; McClay, Mackey, Carbonaro, Szafron, & Schaeffer, 2007; Szafron et al., 2005). Game programming as part of a university Computing Science curriculum is not new; most university computing programs offer such a course. However, computer game *construction* in K-12 is rare. In a recent example, an after-school program (with support from the National Science Foundation) called the Girls Creating Games program (GCG) was created to stimulate female interest in information technology (Denner, 2007; Denner, Werner, Bean, & Campe, 2005). Quantitative and qualitative studies involving female middle-school students (grades 6–8, average age 11.7 years) suggest that the GCG program improved skills and knowledge of computers while decreasing negative stereotypes associated with working in the field of information technology. Denner provides extensive resources for teachers (programservices.etr.org/gcgweb/). In particular, fairly simple games were programmed using Flash (www.macromedia.com). Note that the activity we are promoting consists of both the *design* and *construction* of games. For simplicity, we will use the term game *construction* to encompass both activities. We selected this word to emphasize that the result of these equally important activities is an actual artifact that can be shared and discussed. We often use the term author to describe an individual who participates in the design and construction process.

Constructivist learning theory argues that knowledge is built by the learner, very often in a social context (Vygotsky, 1978). A corollary of constructivism, *constructionism*, further argues that this knowledge-building process can be facilitated through the construction of tangible, shareable artifacts such as text, plays, stories, models, working machines, computer programs, and even toys (Papert, 1991). From a learning and problem-solving perspective, the act of constructing such an artifact is much different than interacting with that artifact in its final form (Penner, 2001). Recent research has shown that even though boys play computer games for significantly more hours than girls, there is no difference in their ability to construct computer games (software artifact) of comparable complexity (Carbonaro et al., 2008).

Unfortunately, the construction of a complex computer game, with 3D graphics and a multi-threaded 'story' architecture, is difficult because of the sophisticated level of programming knowledge that is required. Papert's (1980) early work on the development and use of the Logo programming language demonstrated how abstract and concrete relationships could be managed so that computer programming could be made accessible to children—without compromising the programming integrity of Computing Science. Simple programs could be constructed, observed, reflected upon, revised, and shared with others. The core ideas of his work, that programming could be made more accessible to children and non-programmers, has resonated throughout Computing Science, and is instantiated in new language environments such as the Lego NXT G graphical programming tool (Kelly, 2007) and the Alice object-oriented programming world (www.alice.org).

The Alice programming environment is an excellent example of how novice Computing Science students can learn basic programming constructs by using a simplified, yet functionally powerful programming environment. Alice allows students to write 3D animated movie type stories by dragging and dropping code elements to form scenes (Kelleher & Pausch, 2007). A modified version of Alice, called Storytelling Alice, has been successfully used with middle-school girls to motivate them to learn programming (Kelleher et al., 2007).

Other scientific disciplines have increased their enrollment by drawing from both genders. Without diversity of gender there is a real possibility that new ideas and potential opportunities in Science will be missed (Kenway & Gough, 1998). The research presented in this paper describes how student construction of computer games in grades K-12 can be used to increase interest in the Computing Science discipline for both genders, and to enhance general higher-order thinking skills that are applicable across all Science disciplines. Since males have a higher interest in playing computer games than females, one might expect that computer game construction activities may stimulate more interest and enthusiasm for Computing Science from males than females and further alienate females from this discipline. However, we show that despite a large disparity in interest and experience between males and females with respect to playing computer games, this relative lack of experience has no impact on their success and enjoyment in constructing computer games. An affinity for playing computer games certainly draws males to the discipline of Computing Science. We show that the lack of affinity for playing computer games is not an impediment to using game construction as a mechanism for providing an enjoyable scientific experience. We assert that this enjoyable scientific experience may in fact attract more females to the discipline.

We conducted a study of high-school students who constructed game adventures (interactive stories) for BioWare's Neverwinter Nights (NWN) game (BioWare, 2008), using a commercial game construction toolset (Aurora Toolset) and a scripting tool, ScriptEase (Cutumisu et al., 2007) that we created. The research conducted in this study builds upon previous work that demonstrated high-school students are capable of successfully creating and writing interactive stories (Carbonaro et al., 2008; McClay et al. 2007; Szafron et al., 2005).

In this paper, we argue that: a) computer game creation is a viable activity for teaching higher-order thinking skills that are essential for Science; b) computer game *construction* that involves scripting teaches valuable Computing Science abstraction skills; c) this activity is an enjoyable introduction to Computing Science; and d) outcome measures for this activity are not male-dominated in any of the three aspects (a. higher-order thinking, b. Computing Science abstraction skills, or c. activity enjoyment). To increase overall interest in Computing Science, we propose a general-neutral attractor that allows students to enjoy using Computing Science skills to construct tangible, shareable artifacts.

2. ScriptEase: computer game construction simplified

BioWare Corp.'s Aurora Toolset is a drag-and-drop CAD tool used to define and populate a game adventure that is run using the Neverwinter Nights infrastructure. It provides a rich palette of interior and exterior map tiles, objects, creatures, etc. for constructing the environments in which the game adventure will unfold. Fig. 1 shows an author placing a container (named *Magic Chest*) into a room.

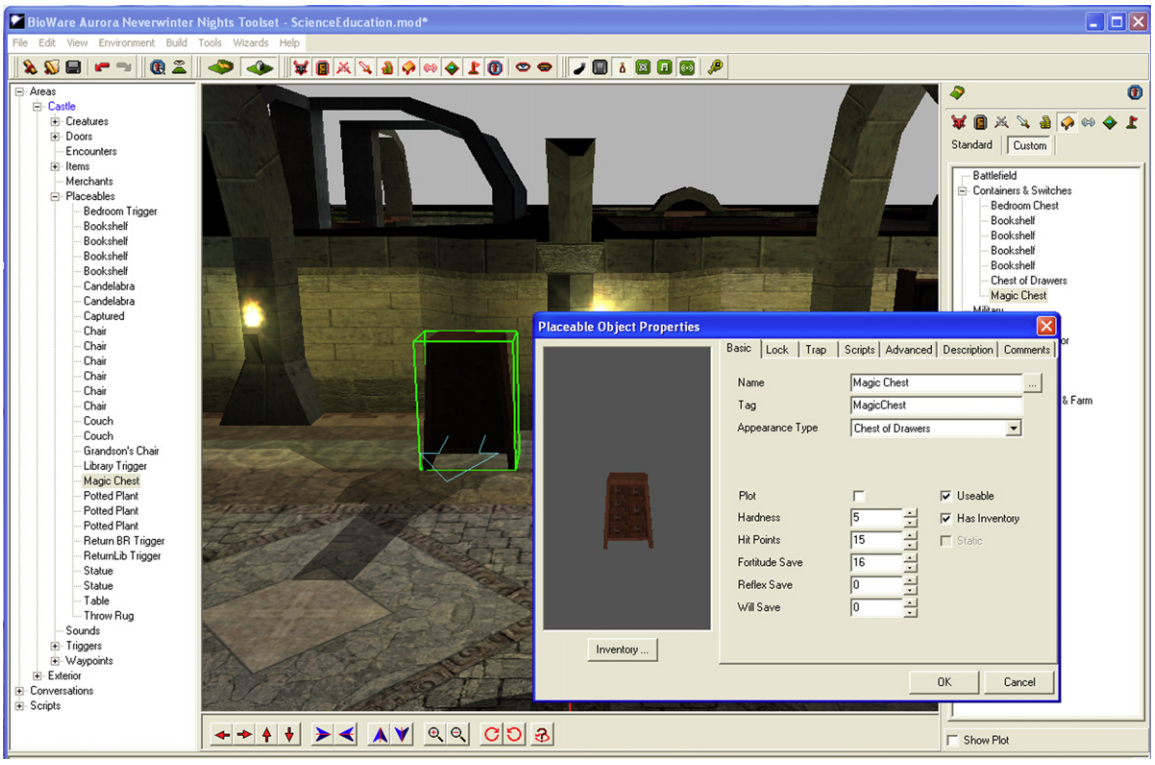

Fig. 1. Using the Aurora Toolset to manipulate objects in the game world.

ScriptEase (Cutumisu et al., 2007) is a tool that simplifies the task of defining the interactions between the player character and game objects. Normally, this is done using a scripting language, which is usually a simplified version of a programming language, such as C or C++. However, with ScriptEase, the game author specifies interactions through the use of patterns, a familiar concept or idiom in a game adventure, rather than writing explicit scripts. From an author's point of view, using ScriptEase to specify the game content is a simple three-step process:

1. Select an appropriate pattern and create an instance of it. ScriptEase provides a rich set of patterns for role-playing games. These patterns can be used for specifying the plot, dialogues, character/object interactions, and character behaviours. For example, when a magic container is used (e.g., opened or closed), the author might want to have a creature spawned nearby. This can be done using the encounter pattern[2] *Placeable use – spawn creature*, since a container is represented by a *Placeable* object in the game. Instantiation of a pattern generates a high-level natural language description. Fig. 2 shows this pattern in ScriptEase.
2. Adapt the pattern. Each pattern must be customized to match the intent of the author for the specific story being told. The author adapts the pattern by selecting appropriate game adventure information using menu selection and dialog boxes. For example, when using the *Placeable use – spawn creature* pattern, the author must specify *The Placeable*, the *Spawned Creature*, and a visual *Spawn Effect* that will be displayed when the creature is spawned. Fig. 2 shows *The Placeable* option being set to the *Magic Chest*. The simplicity of the adaptation process is a key to success. Setting options is the simplest form of pattern adaptation. There are other kinds of adaptation as well and the encounter pattern in Fig. 2 has been opened to show its contents so that the other kinds of adaptation can be described (next section).
3. Generate scripting code. The author presses a button and the scripting code is generated automatically from the adapted pattern. Most authors never see the scripting code.

Once the scripting code is generated, the author can push a button to test the game. When the Player Character (PC) opens the *Magic Chest*, a creature is spawned as shown in Fig. 3.

2.1. Adapting patterns in ScriptEase

The power of patterns comes from the ability of each pattern to represent a wide range of common game scenarios. For example, the *Placeable use – spawn creature* pattern can be applied to any placeable (chest, statue, lever, etc.) and is capable of spawning any kind of creature with any kind of spawn effect. This generality is captured by allowing the author to adapt the pattern by setting *The Placeable*, *The Creature*, and *Spawn Effect* options. However, if pattern adaptability (customization) was limited to only setting options, we would need many more patterns to capture common game scenarios. For example, if the author wanted the placeable to be destroyed after the creature

[2] An encounter pattern describes an interaction between the player character and a game object. A placeable refers to an inanimate game object such as a chest, a lever or a statue.

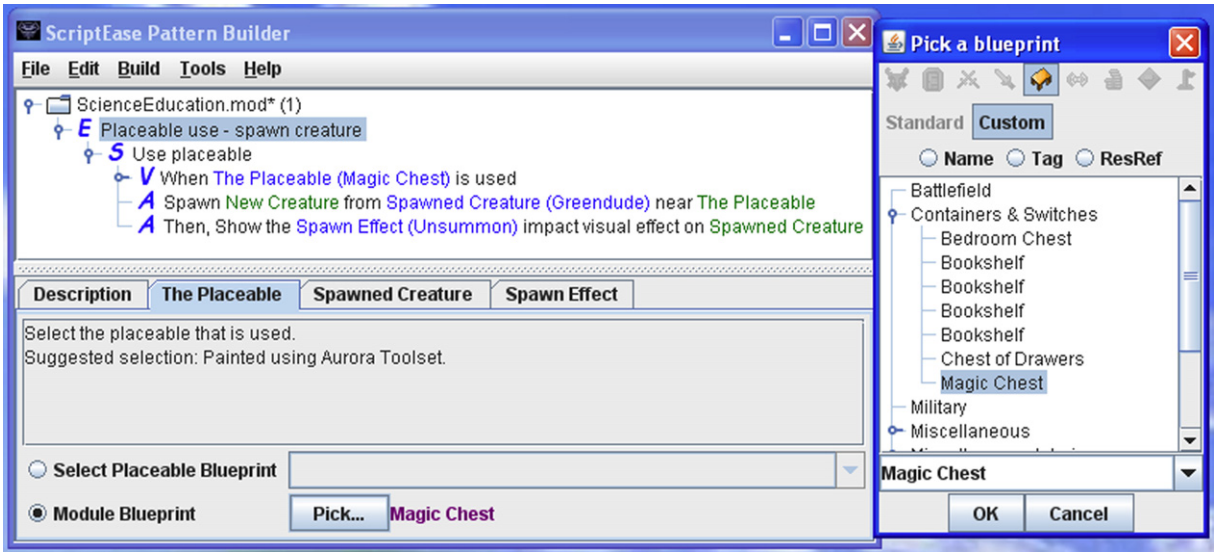

Fig. 2. Using ScriptEase to add a pattern and set an option.

was spawned, we would need a new pattern to represent this scenario. Instead, to keep the pattern catalogue at a reasonable size, we support a much broader range of adaptations. In this case, we allow the author to add an action to the existing pattern to destroy the placeable. In addition to setting pattern options, an author can adapt a pattern in any of the ways shown in Table 1. Since these adaptations correspond to programming, ScriptEase can be used to teach the basics of programming to individuals who do not know how to program. The difficulty categories listed in Table 1 (Cutumisu et al., 2007) were assigned based on the authors' experience in teaching programming concepts to undergraduate Computing Science students. The Computing Science skill column of Table 1 is new and is used in this research.

For example, if the author does not want a visual effect to appear when the creature in Fig. 2 is spawned, the action, *Then, Show the Spawn Effect (Unsummon)* …, could be selected and deleted. If the author wanted to add a new action to the encounter pattern, then the author could select the encounter, and use a menu to add a new action, such as *Destroy Object*. In this case, the author can then set the *Destroyer* option of the *Destroy Object* action to be the *New Creature*, so that the placeable is destroyed after the creature is spawned. Fig. 4 shows this action after it has been added to the encounter pattern in Fig. 2, with the *Destroyer* option being set to the *New Creature*. Note that the *Spawn Creature* action defines a new object, named *New Creature*. Authors have access to these new objects in the *Select Object* menu box at the bottom of the dialogue. Adding an action and setting an option is like adding a function, procedure or method call to a program and setting its arguments. The inset of Fig. 3 shows the in-game consequences of deleting the visual effect and adding the *Destroy Object* action. This is

Fig. 3. The NWN game adventure encounter created by the *Placeable use – spawn creature* pattern of Fig. 2, where the inset in the upper right shows the changed scene when the pattern of Fig. 4 is used instead – the visual effect is gone and the *Magic Chest* disappears after the creature has spawned.

Table 1
Adaptation operations for ScriptEase encounter patterns.

Difficulty Category	Adaptations	Computing Science skill
0	Set options	Types and argument setting
1	Delete a situation	
2	Delete an action or a definition*	
3	Delete a condition*	
4	Replace an action or definition placeholder	Function, procedure or method calls
5	Add an action or a definition	Function, procedure or method calls
6	Replace a condition placeholder	Booleans and selection control structures
7	Add a condition	Booleans and selection control structures
8	Add a situation	Events

Note. Operations that are rarely used are marked with an asterisk (*).

a gentle introduction to function calls and argument setting, using menu selection and options. This high-level abstraction allows an author to learn the concept of function calls, while hiding the low-level code that is generated (shown in Fig. 5). This approach shifts the focus of learning about function calls from syntax to semantics.

Although the following terminology is not used when talking to game authors, the *Select Object* menu box is a complete list of variables in scope that are of compatible type for the option being set. In this case, the *Destroyer* option has type *Object* that matches any game object (*Creature*, *Placeable*, *Item,* etc.). This menu box includes any new local variables created by actions (*New Creature*), any implicit variables defined by the situation (*User* – the creature that used the placeable), and any variables of compatible type defined as options (see Fig. 2) in the encounter (*The Placeable*). The encounter option (*Spawned Creature*) is not included, since its type is the kind of creature to be spawned (*Greendude*), not the actual creature spawned. The encounter option (*Spawn Effect*) is not included since it has type *Visual Effect*, which is not compatible with a game object (*Object*). The authors obtain a gentle introduction to the concept of "types", an important Computing Science concept, without the jargon normally associated with them in a more sophisticated course. A *type* is used in Computing Science to categorize the data that is manipulated in an algorithm or program, with different operations defined on each type. For example, a *Creature* is a different type than a *Placeable*, since a *Creature* can walk, talk, and use a *Placeable*, while a *Placeable* cannot do any of actions.

The *Add a condition* adaptation is more complex. For example, if the author wanted the encounter in Fig. 4 to occur only if the PC was not carrying a particular magic book (*The Origins of Magic*), then the author could use a menu to add a definition (*Whether a creature/placeable has an item*) and a condition (*If negative*) to the encounter and set the options on the definition and condition. Adding a condition teaches authors about Booleans and selection control structures, although these terms are not used. Fig. 6 shows the author setting the *Owner* option of the new definition, where *User* is pre-defined as the creature that used the placeable (in this case the *PC*). Fig. 7 shows some of the script code generated by ScriptEase for the condition in Fig. 6.

Tenth grade English students with no previous programming experience have successfully used ScriptEase to manipulate patterns of this complexity for constructing their game adventure stories (Carbonaro et al., 2008).

3. Methodology

3.1. The ScriptEase study participants and components

Two tenth grade high-school classes used ScriptEase and the Aurora Toolset to author interactive NWN game adventures. There were 50 participants in the study, 24 females and 26 males. Each class was from a different high school located in suburban middle class

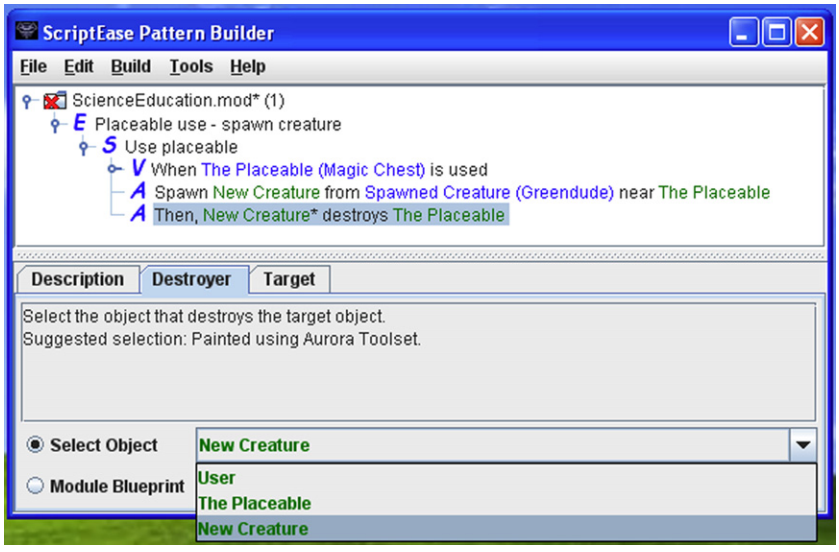

Fig. 4. Adding an action to a pattern and setting its options.

```
// Define User as the user of Magic Chest
User_SE0 = GetLastUsedBy();

// Define The Placeable as the object the event is fired on
ThePlaceable_SE1 = OBJECT_SELF;

// Main code body - checks conditions and executes actions

// Spawn New Creature from Greendude near The Placeable
NewCreature_SE2 = SE_Ac_SpawnCreatureNearObject("goblina001", ThePlaceable_SE1);

// New Creature*  destroys The Placeable
AssignCommand(NewCreature_SE2, ActionDoCommand(DestroyObject(ThePlaceable_SE1)));
```

Fig. 5. The code generated by the *Destroy Object* action shown in Fig. 4, along with code for the option objects.

neighborhoods in Edmonton, Canada. The classes were multi-ethnic as reflected by the fact that 14 of the 50 participants who answered the language fluency question were fluent in at least one language other than English (Albanian, Arabic, Cantonese, Farsi, French, Hindi, Mandarin, Polish, Punjabi, Russian, or Spanish). The subject area for both selected classes was English Literature/Composition, and most of the students had no previous formal experience with computer programming and had made no explicit decision to study Computing Science (Carbonaro et al., 2008).

Each class attended a two-day workshop at the University of Alberta where they spent a total of 6 h working through two tutorials (www.cs.ualberta.ca/~script/supplementary01.html), followed by 2 h in which they began to construct an interactive game adventure. After this, students spent a total of four more hours in their school, finishing their interactive adventure. During both the workshop and the school sessions, the teacher and graduate students supervised the students and answered questions about the tools. The first tutorial introduced students to playing NWN and the second tutorial taught them about the Aurora Toolset and ScriptEase. Students were then provided with an adventure module containing two areas, the exterior and interior of a castle. No objects (placeables, items, creatures, etc.) and no scripts were provided. Students then constructed their game adventure by populating these two areas with game objects (characters and props) using the Aurora Toolset and generating scripts for these objects using ScriptEase.

3.2. The instruments for the ScriptEase study

Several instruments were used to measure student attributes. In this paper, we use results from exit surveys (www.cs.ualberta.ca/~script/supplementary01.html), a rubric that assesses the higher-order thinking skills that a player of a game adventure would require, and a rubric that assesses the Computing Science skills used to author a game adventure. The exit survey included data about enjoyment of the game adventure construction activity.

3.3. Higher-order thinking

As a first attempt to measure the utility of game adventure construction as a means of learning skills applicable across scientific domains, we used the "Video Game Higher Order Thinking Evaluation Rubric" created by Rice (2007). This rubric produces a score in the range from

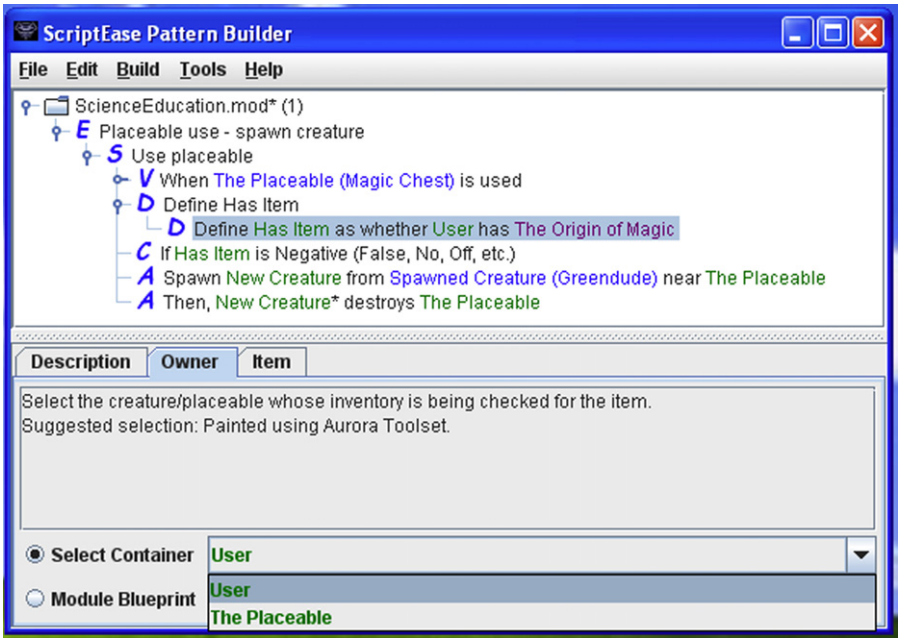

Fig. 6. Adding a definition and condition to a pattern and setting their options.

```
// Define Has Item as whether User has The Origin of Magic
HasItem_SE3 = GetIsObjectValid(GetItemPossessedBy(User_SE0, GetTag(TheOriginofMagic_SE2)));

// Define User as the user of Magic Chest
User_SE0 = GetLastUsedBy();

// Define The Placeable as the object the event is fired on
ThePlaceable_SE1 = OBJECT_SELF;

// Define Has Item as whether User has The Origin of Magic
HasItem_SE3 = GetIsObjectValid(GetItemPossessedBy(User_SE0, GetTag(TheOriginofMagic_SE2)));


// Main code body - checks conditions and executes actions

// If Has Item is Negative (False, No, Off, etc.)
if( ! HasItem_SE3 ) {

  // Spawn New Creature from Greendude near The Placeable
  NewCreature_SE4 = SE_Ac_SpawnCreatureNearObject("goblina001", ThePlaceable_SE1);

  // New Creature*  destroys The Placeable
  AssignCommand(NewCreature_SE4, ActionDoCommand(DestroyObject(ThePlaceable_SE1)));

 }
```

Fig. 7. The code generated by a ScriptEase condition.

0 to 20. The score is then mapped onto a Video Game Cognitive Variability Scale (VGCVS). The VGCVS is defined in Table 2. Although this rubric is designed to evaluate the higher-order thinking skills used when playing video games, it is a good first step to measure the higher-order thinking skills used to construct game adventures as well, since the author uses similar skills to create situations as the player uses to solve them. However, as outlined by Papert (1991) and illustrated by many others (Bers, Ponte, Juelich, Viera, & Schenker, 2002; Penner, 2001; Resnick, Berg, & Eisenberg, 2000; Vygotsky, 1978), the constructionist activity of producing an artifact (the interactive game adventure) provides an even more powerful learning experience than studying a previously constructed artifact. There are twenty binary questions in Rice's rubric, which we illustrate by listing six of them: 1) has a storyline, 2) has a complex storyline with characters users care about, 3) offers simple puzzles, 4) has complex puzzles requiring effort to solve, 5) allows different ways to complete the game, and 6) requires gathering of information in order to complete.

3.4. Computing Science abstraction skills

We created a new instrument to assess the Computing Science skills that the students used to construct their game adventure. The instrument counts the percentage of students who used each kind of pattern adaptation listed in column two of Table 1, as a measure of the computing skills (column three) that they displayed in constructing their game adventure.

If one teaches programming "top down" in an interpreted environment like Smalltalk, Lisp, Logo or Alice, then picking a function requires typing a function call so that it can be evaluated (executed) immediately. In this case, the first programming skill learned is how to pick an appropriate library function from a large set to solve a specific problem and how to set the parameters of this function to values appropriate for the specific problem. For example, in Smalltalk a student might learn how to use the "+" message to add two numbers by entering: "3 + 4" in a workspace and observing the answer, 7. In ScriptEase, this skill is introduced and learned when students select an appropriate pattern from a menu that will create the appropriate situation in their game module. After selecting the appropriate pattern, a student must set options (parameters) that the pattern uses in the game. These options are usually game objects or values. For example, if a student selects the *Placeable use – spawn creature* pattern, the student must select two options: a specific placeable whose use triggers the situation and a specific creature blueprint to use in spawning. Selecting a pattern and setting its options is rated as difficulty level 0 in Table 1.

Adapting a pattern consists of various operations that are also used in computer programming. We previously categorized these adaptations by difficulty level (Table 1), based on our experience in teaching computer programming. For example, adding a condition is more difficult than adding an action, since adding an action corresponds to adding a simple library function call, but adding a condition requires the author to understand Boolean values and the equivalent of selection control structures. Replacing a component is easier than adding a component since the location does not need to be determined when replacing.

Table 2
VGCVS, (Rice, 2007, p. 94).

20	*Perfect score*. Game displays highest elements of cognitive viability.
18–19	*Upper-range*. Game holds several positive characteristics lending itself to higher-order thinking.
14–17	*Mid-range*. Game is probably acceptable for some higher-order thinking opportunities.
9–13	*Lower-range*. Fewer opportunities for higher-order thinking will take place in the game.
0–8	*Little or no cognitive viability*. Typical score range for arcade-style only games.

3.5. Activity enjoyment (fun)

To draw more students to Computing Science, we need to increase the number of introductory activities that are enjoyable. To increase the percentage of female students interested in Computing Science, we must ensure that the activities that we use to introduce Computing Science are enjoyable to female students. Otherwise, we risk further alienation of the female students. The following five questions from the exit survey were used to gather data about student enjoyment of this activity.

1. How much fun did you have writing your traditional short story at school before the field trip to the University of Alberta? (circle):
   none not much some quite a bit a huge amount
2. How much fun did you have learning to write an interactive story during your field trip to the University of Alberta? (circle):
   none not much some quite a bit a huge amount
3. How much fun did you have writing your own interactive story at school after the field trip to the University of Alberta? (circle):
   none not much some quite a bit a huge amount
4. Would you like to write more traditional short stories in school (circle):
   no yes
5. Would you like to write more interactive stories in school (circle):
   no yes

Questions one and four are control questions that attempt to determine a base line of fun for students in these classes. These questions were designed to measure their fun level in performing a traditional activity in the school. Since the classes were tenth grade English classes, we picked a typical activity in this class – writing a traditional short story in the school environment. Questions two and three were designed to determine if the game adventure authoring activity was enjoyable, not only in the field trip mode (question two), but also as an activity that could be done in the school (question three) over a longer period of time.

4. Results

Using ScriptEase is a new gentle approach to introducing Computing Science. We describe the higher-order thinking skill evaluation, the Computing Science abstraction skill measurement, and our measurements of student enjoyment for this approach. In each case, we also discuss the differences in the scores based on gender.

4.1. Higher-order thinking

Two raters used Rice's (2007) rubric to evaluate the student's interactive game adventures with respect to their ability to promote higher-order thinking. The fifty high-school students each authored a video game adventure. The results are shown in Fig. 8, where the score for each evaluator is shown as a separate bar in a pair of adjacent bars. The final score for each student is the average of the scores assigned by the two evaluators. The mean final score across all students is 15.44 (SD = 1.13). This puts the score in the mid-range (14–17) as defined by Rice (2007) in Table 2. Therefore, we conclude that this game adventure authoring activity stimulates higher-order thinking skills. We evaluated the inter-rater reliability of these ratings using Cohen's (1960) kappa coefficient and obtained a value of .994, which indicates almost perfect agreement (Landis & Koch, 1977). Fig. 8 shows the overlap of the two evaluators, with perfect agreement on fourteen items.

We divided the higher-order thinking data, based on gender as shown in Fig. 9. The mean scores for interactive games for the males and females were 15.12 (SD = 1.12) and 15.92 (SD = 1.13) respectively. Given that the games produced by the females had a higher mean score, we performed a two-sample *t*-test to determine if the female scores were significantly higher than the male scores at a 95% confidence level. There is a significant difference ($p = .015$).

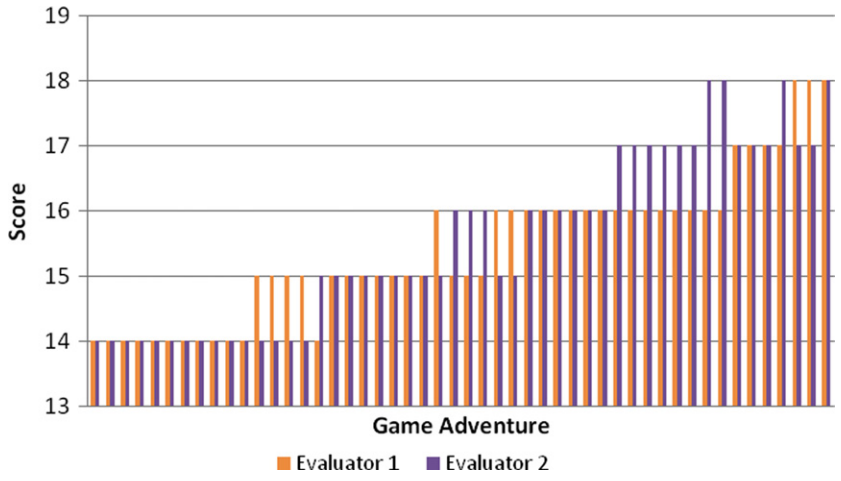

Fig. 8. Higher-order thinking scores by evaluator.

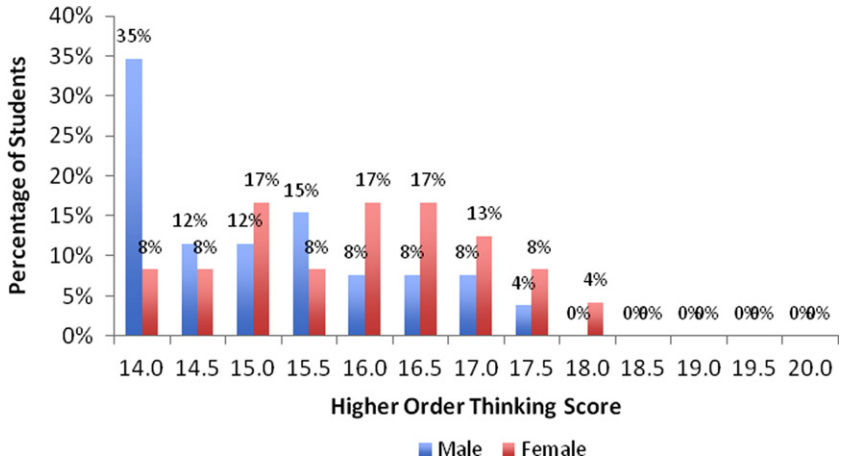

Fig. 9. Higher-order thinking scores by gender.

4.2. Computing Science abstraction skills

We analyzed the modules of each of the students and recorded all of the patterns that they used and all of the adaptations performed by each student on each of their patterns. Since these measures were objective counts of ScriptEase lines performed by machine, no inter-rater reliability was necessary. The stated goal of the activity was to construct a game adventure and the students spent a total of 6 h on this activity (after 6 h of training). In this time they were encouraged to create game objects and place them in the game, write dialogue for the character objects, and use ScriptEase to make the objects interact. There were no requirements or guidelines about how much time they should spend on each of these three sub-activities. Since there were no explicit instructions to students that they had to use ScriptEase to demonstrate a specific skill level, we cannot conclude that any student who did not use ScriptEase was incapable of using it. Therefore, when we report that a certain percentage of students used a skill, we can conclude that they successfully learned that skill. However, we cannot conclude anything about the relationship between the other students and the skill. For example, we cannot conclude that the other students tried the skill and failed to use it when building their game. The amount of time spent using ScriptEase on their game adventure varied from student to student and ranged from 0 h to about 2 h out of the 6 h. The majority of the time was spent using the Aurora Toolset to populate their game adventure with game objects and to write dialog for the characters.

Of the students in this study, 98% (49/50) selected at least one pattern for their module. The other student did not use any patterns. This student's module was still playable since it had characters and dialogue. However, since no patterns were used, no scripts were generated, so the interaction with the PC was very limited. We cannot conclude that the student was incapable of using a pattern – we can only conclude that the student chose not to use any patterns. We can conclude that 98% of the students learned a valuable Computing Science skill: how to select an appropriate computational artifact from a library. Only one of the students who successfully selected a single pattern did not set the options properly, so 96% (48/50) were successful in learning how to select an artifact from a library and customize the artifact by setting options (parameters) appropriate to the problem being solved.

The percentages of students who successfully used each adaptation type are shown in Fig. 10. The labels on the *x*-axis are in order of our nominal difficulty level listed in Table 1, except that the two rare abstractions (Delete an action or definition; Delete a condition) are put at the far right. The "set options" adaptation is excluded, since we already know that 96% of students were able to perform this adaptation. Only two students that successfully used patterns did not perform any adaptations other than setting options. This means that 94% (47/50) students successfully used patterns in which other adaptations were used. The green bars in the graph indicate that 80% of the students learned to delete situations, 84% learned to replace actions and/or definitions and 50% learned to add actions and/or definitions. The skill required to perform these actions is analogous to the skill required to write sequential computing programs where the statements are

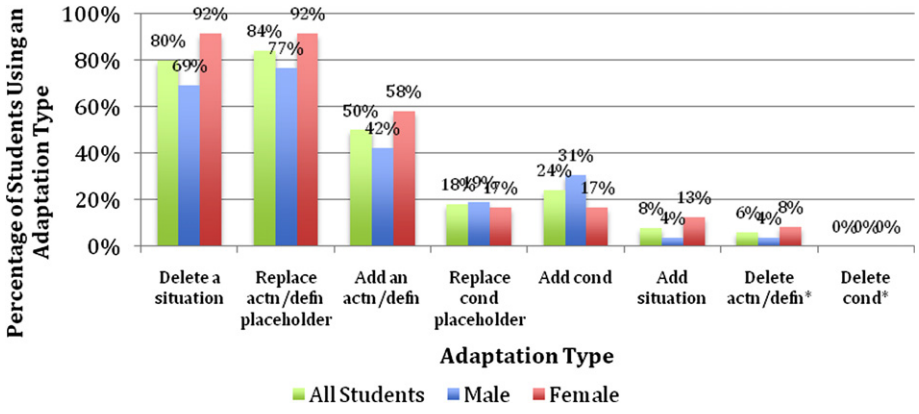

Fig. 10. Percentage of students that used each type of abstraction.

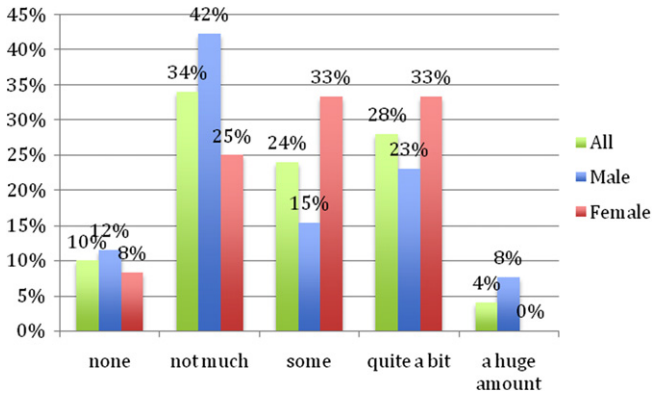

Fig. 11. Percentage of students who enjoyed traditional story writing at school (the control activity).

function calls (or message expressions in the object-oriented paradigm). The groups of students who performed these operations were not monotonic, in that not every student who replaced an action also deleted a situation. That is why the percentage of students who used at least one adaptation (94%) is higher than the percentage of students who used any single kind of adaptation.

The next two types of adaptations shown in the graph (Replace a condition placeholder – 18%) and (Add a condition – 24%) correspond to learning Computing Science skills related to Booleans and selection control structures. Given the short amount of time students had to complete their stories and the focus on complete stories rather than versatility of interactions in the stories, we were pleasantly surprised that so many students were able to find time to perform these adaptations.

The most difficult adaptation is to add a situation. A situation consists of an event, a condition, and a set of actions that are performed when the event occurs and the conditions are met. This is the most conceptually difficult adaptation and is most analogous to writing a function or method in a program. Nevertheless, four students (8%) learned to use this kind of adaptation, even though there was very little time to learn. Fig. 10 provides some evidence that our difficulty ranking in Table 1 is correct.

Given the limited time that students spent on the ScriptEase (programming) part of this activity (0–2 h), we feel the percentages shown in Fig. 10 indicate that students attained a reasonable level of Computing Science skills. We also factored this data by gender, as shown in Fig. 10 and used *t*-tests to compare the percentages for each kind of adaptation by gender. The only difference that is statistically significant ($p = .04$) is that the percentage of female students that deleted a situation was higher than the percentage of male students. However, since replacing an action/definition placeholder is slightly more difficult and there is no statistically significant difference between the percentage of male (77%) and female (92%) students who performed this type of operation, the differences in percentages for deleting a situation can safely be ignored.

4.3. Activity enjoyment (fun)

Figs. 11–13 show the percentage answers for exit survey questions one, two, and three respectively. We subsequently assigned a numerical score to each answer, from 0 for "none" to 4 for "a huge amount". Table 3 shows the mean scores for each of the first three questions for both males and females.

A repeated measure using the Greenhouse-Geisser (Field, 2005) correction for unequal correlations between the three levels of the independent variable (i.e., 'How Much Fun', the first three questions) was used to analyze the mean scores reported in Table 3. Results from the analysis show a significant difference between the levels, $MS_R = 19.192$, $F = 21.758$, $p > .001$, indicating there are reliable differences between the levels (Question 1: *fun-traditional*, Question 2: *fun-interactive-university*, Question 3: *fun-interactive-school*). Within subject contrasts show *fun-traditional* is significantly different from *fun-interactive-school* ($MS_R = 56.866$, $F = 31.5$, $p > .000$) and that *fun-interactive-university* is significantly different from *fun-interactive-school* ($MS_R = 34.533$, $F = 45.306$, $p > .000$). Therefore, students appeared to enjoy interactive game adventure writing over traditional writing and they also enjoyed writing this interactive game adventure at the university

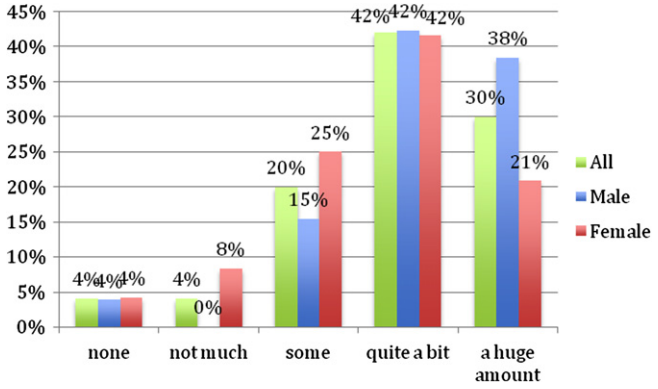

Fig. 12. Percentage of students who enjoyed game adventure authoring during the field trip.

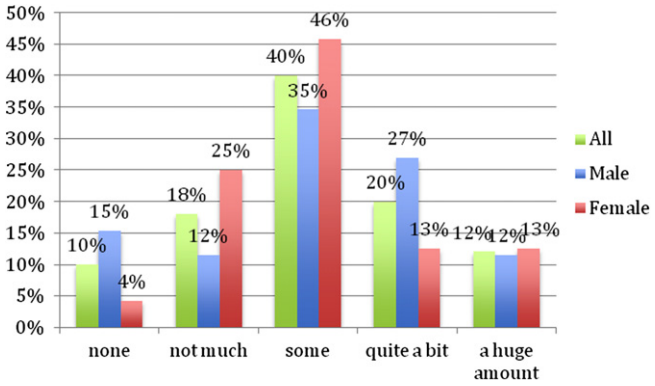

Fig. 13. Percentage of students who enjoyed game adventure authoring at school.

over writing it at school. Furthermore, there was no significant difference across gender, $MS_R = .123$, $F = .180$, $p > .673$, for any level. This indicates no difference in the enjoyment of these activities between males and females.

The results indicate that most of the extra enjoyment was derived from the novelty of being on a field trip at the university rather then on the activity itself. Nevertheless, for question three, with 72% of the students indicating that they enjoyed writing a game adventure in school at the level of 'some' (40%) or higher (32%), this activity can be regarded as an enjoyable way to learn higher-order thinking skills in general, and Computing Science skills in particular.

Questions four and five provide additional information on whether students enjoyed writing either traditional or interactive stories in school. The results are shown in Fig. 14. Although only 34.0% of all students (30.8% of males and 37.5% of females) would like to write more traditional stories in school (the control activity), 63.3% of all students (69.2% of males and 56.5% of females) would like to write more interactive stories in schools.

Given the dichotomous nature of the data for questions four and five, we used Chi-Square (χ^2) to compare *more-traditional-in-school* (Question 4) and *more-interactive-in-school* (Question 5) across gender, resulting in $\chi^2\,(1, N = 50) = .252$, $p = .210$ and $\chi^2\,(1, N = 49) = .845$, $p = .185$ respectively. Thus there were no significant differences on each of these questions with respect to males and females.

We also used the McNemar test (Field, 2005) for comparing two paired dichotomous observations, *more-traditional-in-school* versus *more-interactive-in-school*, and obtained $\chi^2\,(1, N = 49) = 5.63$, $p = .018$. There is a significant difference in favor of the students wanting to write *more-interactive-in-school*. This result is partly due to the higher rating on Question 5 given by males (69%) than females (56%) and partly due to the lower rating on Question 4 writing given by males (31%) than females (38%). A subsequent analysis of *more-traditional-in-school* versus *more-interactive-in-school* by gender reveals no significant difference for females, $\chi^2\,(1, 23) = .006$, $p = .388$, but a significant difference, $\chi^2\,(1, N = 26) = 2.006$, $p = .031$, for the males. Therefore, it appears that males favor writing interactive stories at school over traditional stories, whereas girls report no difference in this context.

5. Discussion

The results support our contention that computer game *construction* (as opposed to game playing) is a viable activity for teaching higher-order thinking skills that are essential for Science. Each game adventure constructed by a student represents a narrative artifact in a computational form. As the findings in Fig. 8 show, every interactive game adventure constructed by a student was classified in the midrange or above on Rice's (2007) assessment scale. The idea that students can build computational models of abstract ideas that are sharable and can be reflected upon by others is an important pedagogical activity of scientific understanding, thinking, and conceptual development (Penner, 2001). What makes this observation even more profound is: a) females ($M = 15.92$) exercised their higher-order thinking skills at a significantly higher level than males ($M = 15.12$); and b) Rice's scale was designed to assess video games that were professionally produced, whereas our games (in a story context) were produced by grade 10 English students.

Our second point is that computer game construction that involves scripting teaches valuable Computing Science abstraction skills. The results also support this argument. Students were able to use our scripting tool (ScriptEase) to construct complex games (stories) that embody a deep level of computer programming knowledge without ever writing a single line of 'traditional' code. The ability for a student to set function parameters and embed conditional logic demonstrates a mastery of abstract programming knowledge of the type typically taught in introductory Computing Science. Other Computing Science abstraction skills include selecting an appropriate entity from a library (98% of participants) and setting parameter options (96%), writing sequential statements that use function calls (50%), selection control structures and Booleans (24%), and writing custom functions (8%). We feel these percentages would be even higher if the scripting part of this activity was performed for a longer time than 2 h. Given that we selected a non-Science course (grade 10 English) to teach Computer

Table 3
Mean score of student's ratings on the questions concerning "how much fun" they had writing stories in three different settings.

		How much fun (mean scores)		
		Writing traditional story at school	Writing interactive story at university	Writing interactive story at school
Gender	Male	1.73	3.12	2.08
	Female	1.93	2.67	2.04

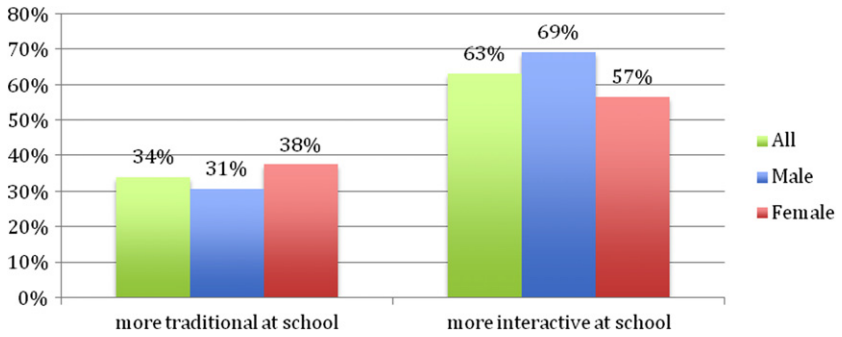

Fig. 14. Percentage of students who would like to write more traditional and interactive stories at school and at home.

Science concepts, we were pleased that students enjoyed the interactive game adventure activity. Teaching Computing Science concepts through narrative construction clearly demonstrates the power of the ScriptEase toolset.

The results support the activity of interactive game adventure authoring as enjoyable introduction to Computing Science with 92% of the students enjoying it at least "some" in a field-trip setting, 72% enjoying it at least "some" in a high-school setting, and 63% who would like to continue this activity for a longer period of time in school. The findings showed that interactive game adventure authoring was more fun that traditional story writing in both the university and school setting. It is not surprising that the novelty of the university setting appeared to be the most fun. A field trip to the university for high school students is generally considered to be an exciting event. There were no gender differences found across the independent variable (fun). This is an important finding when it comes to introducing Computing Science concepts.

A major impetus for our work is to encourage female students to enjoy learning Computing Science in a context where both males and females are having fun. Prensky (2002) argued for the concept of 'fun' in computer gameplay as a motivational tool for learning. In our findings, 'fun' at writing interactive stories is gender neutral thereby placing males and females on equal footings as computer game builders, as opposed to computer game players, which favor males. There are two important questions related to enjoyment. First, is game adventure authoring 'fun enough' that it might help to increase interest in Science (in general) and Computing Science (in particular) above the current percentages of 36% of the males and 11% females that were somewhat likely to select Computing Science as a major? Second, could game adventure authoring be "fun enough" for females that it might help to raise the percentage of graduating students from Computing Science programs that are females above 14%? We believe that the enjoyment rates in Figs. 12 and 13, and Fig. 14 provide some evidence that the answer is yes to both questions. A positive result was finding no gender difference with respect to writing traditional versus writing interactive stories. Interactive stories are synonymous with computer games and the idea that girls enjoyed writing in either traditional or interactive stories equally well is encouraging.

Although we found no within-group gender differences with respect to students wanting to write more traditional or more interactive stories in school (questions 4 & 5), we did find that students overall would prefer writing interactive stories at school as opposed to traditional stories (between-group difference for questions 4 & 5). This is a positive indication that, given the right tools and instructional context, interactive game adventure authoring would be a well-received pedagogical intervention in schools. Subsequent analyses showed that males did differ from females on this issue and thus were more inclined to like to write more interactive stories at school than traditional stories at school.

Given the interactive game adventure authoring tools, methods, and outcome measures used for this study, it was encouraging to find that females scored significantly better than males on higher-order thinking skills and that Computing Science abstraction skills were gender neutral. It was even more encouraging to see that students were actively incorporating Computing Science concepts into the construction of their stories without even realizing it. In this sense, both game adventure authoring and programming appear as a seamless interwoven construction process, one that is analogous to Papert's (1980) early work on the relationship between Mathematical concepts and Logo. We view interactive game adventure authoring as an important scaffolding activity that supports the learning of Computing Science concepts by allowing students to be game constructors in a gender neutral way. Interestingly males, more than females, liked writing interactive stories at school over writing traditional stories at school. This result did not surprise us, given that males at that age tend not to enjoy writing as much as females (MacArthur, Graham, & Fitzgerald, 2008). We interpret males wanting to write interactive stories as a positive result in the overall context of writing.

6. Conclusion

This study represents the importance of building a strong collaboration between Science and Education to help overcome a serious issue, the underrepresentation of females in Computing Science. A recent study by Anderson, Lankshear, Timms, and Courtney (2008) concluded that senior high-school girls perceive advanced computing subjects as boring and irrelevant, and they often express a strong aversion to computers. The perception by females that Computing Science is boring is in direct contrast to boys who often perceive the subject as an exciting area, largely do to their unprecedented attraction to the playing of computer games (Carter, 2006). That being said, research suggests that when the environment is supportive and female-friendly, retention in Computing Science is possible (Cohoon, 2006). To increase the representation of females in the field of Computing Science, it is critical to create an educational environment that they perceive as more enjoyable. Our study shows that given the right toolset and pedagogical context, female students can develop higher-order thinking skills, master abstract computational concepts, construct meaningful computational artifacts (interactive stories), in a gender neutral activity, and have fun doing it. Our hope is that this learning environment encourages female students to pursue Computing Science as a field of study and work.

References

American Association of University Women Educational Foundation. (1996). Girls in the middle: working to succeed in school. www.aauw.org/research/girlsinMiddle.cfm Retrieved Dec 17, 2008.

Anderson, N., Lankshear, C., Timms, C., & Courtney, L. (2008). 'Because it's boring, irrelevant and i don't like computers': why high school girls avoid professionally-oriented ICT subjects. *Computers & Education, 50*, 1304–1318.

Bers, M. U., Ponte, I., Juelich, C., Viera, A., & Schenker, J. (2002). Teachers as designers: integrating robotics in early childhood education. *Information Technology in Childhood Education (ITCE) Annual* (1), 123–145.

Beyer, S., Rynes, K., Perrault, J., Hay, K., & Haller, S. (2003). Gender differences in computer science students. *SIGCSE Bulletin, 35*(1), 49–53.

BioWare Company. (2008). http://www.bioware.com.

Camp, T. (1997). The incredible shrinking pipeline. *Communications of the ACM, 40*(10), 103–110.

Carbonaro, M., Cutumisu, M., Duff, H., Gillis, S., Onuczko, C., Siegel, J., et al. (2008). Interactive story authoring: a viable form of creative expression for the classroom. *Computers & Education, 51*(2), 687–707.

Carter, L. (2006). Why students with the apparent aptitude don't choose to major in computer science. *SIGCSE Bulletin, 38*(1), 27–31.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37–46.

Cohoon, J. M. (2006). Just get over it or get on with it: retaining women in undergraduate computing. In J. M. Cohoon, & W. Aspray (Eds.), *Women and information technology: Research on underrepresentation* (pp. 205–237). Cambridge MA: MIT Press.

Cohoon, J. M., & Aspray, W. (2006). A critical review of the research on women's participation in postsecondary computing education. In J.M.Cohoon., & W.Aspray. (Eds.), *Women and information technology: Research on underrepresentation* (pp. 137–180). Cambridge MA: MIT Press.

Crowley, K., Callanan, M. A., Tenenbaum, H. R., & Allen, E. (2001). Parents explain more often to boys than to girls during shared scientific thinking. *Psychological Science, 12*(3), 258–261.

CSTA (2006). ACM K-12 CS Model Curriculum, 2nd Edition http://www.csta.acm.org/Curriculum/sub/CurrFiles/K-12ModelCurr2ndEd.pdfRetrieved May 25, 2010.

Cummings, H. M., & Vandewater, E. A. (2007). Relation of adolescent video game play to time spent in other activities. *Archives of Pediatric and Adolescence Medicine, 161*(7), 684–689.

Cutumisu, M., Onuczko, C., McNaughton, M., Roy, T., Schaeffer, J., Schumacher, A., et al. (2007). ScriptEase: a generative/adaptive programming paradigm for game scripting. *Science of Computer Programming, 67*(1), 32–55.

Denner, J. (2007). The girls creating games program: an innovative approach to integrating technology into middle school. *Meridian: a Middle School Computer Technologies Journal, 1*(10). www.ncsu.edu/meridian/win2007/girlgaming/index.htm.

Denner, J., Werner, L., Bean, S., & Campe, S. (2005). The girls creating games program: strategies for engaging middle school girls in information technology. *Frontiers: A Journal of Women's Studies, 26*(1), 90–98.

Dreves, C., & Jovanovic, J. (1998). Male dominance in the classroom: does it explain the gender differences in young adolescents' science ability perceptions? *Applied Developmental Science, 2*(2), 90–98.

Economist.com. (2006, March 23). Computing and the scientific method. http://www.economist.com/displaystory.cfm?story_id=5655067 Retrieved June 3, 2009.

Field, A. (2005). *Discovering statistics using SPSS* (2nd ed.). New York, NY: Sage Publishing.

Goode, Estrella, & Margolis. (2006). Lost in translation: gender and high school computer science. In J. M. Cohoon, & W. Aspray (Eds.), *Women and information technology: Research on underrepresentation* (pp. 89–114). Cambridge MA: MIT Press.

Goode, J. (2007). If you build teachers will students come? The role of teachers in broadening computer science learning for urban youth. *Journal of Educational Computing Research, 36*(1), 65–88.

Graham, S., & Latulipe, C. (2003). CS girls rock: sparking interest in computer science and debunking the stereotypes. *SIGCSE Bulletin, 40*(4), 107–110.

Hazari, Z., Sadler, P., & Tai, R. (2008). Gender differences in the high school and affective experiences of introductory college physics students. *The Physics Teacher, 46*, 428–432.

Hyde, J. S., & Linn, M. C. (2006). Gender similarities in mathematics and science. *Science, 314*, 599–600.

Kafai, Y. B., Heeter, C., Denner, J., & Sun, J. Y. (2008). *Beyond barbie and mortal kombat*. Cambridge, MA: MIT Press.

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM, 50*(7), 59–64.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on human factors in computing systems*. San Jose, California, USA.

Kelly, J. F. (2007). *LEGO MINDSTORMS NXT-G programming guide*. Berkeley, CA: Apress.

Kenway, J., & Gough, A. (1998). Gender and science education in schools: a review 'with attitude'. *Studies in Science Education, 31*(1), 1–30.

Kessels, U. (2005). Fitting into the stereotype: how gender-stereotyped perceptions of prototypic peers relate to liking for school subjects. *European Journal of Psychology of Education, 20*, 309–323.

Kurkovsky, S. (2007). Making computing attractive for non-majors: a course design. *Journal of Computing Sciences in Colleges, 22*(3), 90–97.

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*, 159–174.

MacArthur, C. A., Graham, S., & Fitzgerald, J. (2008). *Handbook of writing research*. NY: Guilford Press.

McClay, J., Mackey, M., Carbonaro, M., Szafron, D., & Schaeffer, J. (2007). Adolescents composing fiction in digital game and written formats: tacit, explicit and metacognitive strategies. *E-Learning, 4*(3), 273–284.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Papert, S. (1991). Situating constructionism. In I. Harel, & S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex Publishing.

Penner, D. E. (2001). Cognition, computers, and synthetic science: building knowledge and meaning through modeling. In W. G. Secada (Ed.), *Review of research in education* (pp. 1–35). Washington, DC: American Educational Research Association.

Prensky, M. (2002). The motivation of gameplay: the real twenty-first century learning revolution. *On the Horizon, 10*(1), 5–11.

Resnick, M., Berg, R., & Eisenberg, M. (2000). Beyond black boxes: Bringing transparency and aesthetics back to scientific investigation. *Journal of the Learning Sciences, 9*(1), 7–30.

Rice, J. W. (2007). Assessing higher order thinking in video games. *Journal of Technology and Teacher Education, 15*(1), 87–100.

Roberts, E. (2007). Stanford university news release. news-service.stanford.edu/pr/2007/pr-robertsaaas-021407.html Retrieved June 10, 2009.

Simpkins, S. D., Davis-Kean, P. E., & Eccles, J. S. (2005). Parents' socializing behavior and children's participation in math, science, and computer out-of-school activities. *Applied Developmental Science, 9*(1), 14–30.

Singh, K., Allen, K. R., Scheckler, R., & Darlington, L. (2007). Woman in computer-related majors: a critical synthesis of research and theory. *Review of Educational Research, 77*(4), 500–533.

Szafron, D., Carbonaro, M., Cutumisu, M., Gillis, S., McNaughton, M., Onuczko, C., et al. (2005). Writing interactive stories in the classroom. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning, 7*(1). imej.wfu.edu/articles/2005/1/02/index.asp.

U.S. Department of Labor Employment & Training Administration. (Thursday, November 15, 2007). www.doleta.gov/BRG/Indprof/ITprofile.cfm Retrieve January 6, 2009.

Vilner, T., & Zur, E. (2006). Once she makes it, she is there: gender differences in computer science study. In *Proceedings of the 11th annual conference on innovation and technology in computer science education*. Bologna, Italy: ITiCSE.

Vygotsky, L. S. (1978). *Mind and society: The development of higher mental processes*. Cambridge, MA: Harvard University Press.