time on this one in grading my students' CS2 assignments) is entitled "Submission" and the behaviors at each performance level were specified as follows.

Attribute	Unacceptable	Meets Standard	Exceeds Standard
Submission	[0.0 points] TA is unable to figure out how to compile and run after 3 minutes.	[1.0 point] Takes 2-3 minutes to figure out how to compile & run.	[2 .0 points] Takes less than 2 minutes to compile; run; test.

Becker's sample rubrics are divided into two sets. Rubric One, entitled "Style", included guidelines that would apply to most any program (such as Presentation, which included the Submission attribute discussed above, and Documentation). Rubric Two, entitled "Function & Design", documented attributes specific to the particular program assignment being assessed (this example was for an English to Latin Translator). In addition to the many English-Latin Translator attributes, Rubric Two contained a Bonus section that allowed for the awarding of extra points for functionality beyond that required of the assignment, which encourages students to go beyond the requirements.

In addition to being a great tool for consistent grading, rubrics allow, or rather force, teachers to clarify their definition of excellence and can be useful in planning how to assist students in achieving excellence. [3] If the rubrics are developed in advance of making an assignment, this may allow us to refine and perhaps "perfect" some assignments before they are handed out.

Rubrics can also be used to provide detailed feedback to students. Rubrics might be provided to students before assignments are submitted, so that students know what the instructor/grader considers most important. However, some students may use the guidelines to guide their performance and perhaps to do the minimum necessary to achieve a certain grade. In some instances, this is not be desirable. For assignments where creativity is desirable, providing rubrics in advance could actually deter creativity. Also, for many assignments, it may not be possible to determine all levels of performance and how they can be demonstrated in advance. Thus, creation of the scoring rubrics can occur as an assignment is graded.

According to Performance Links in Science Assessment, technically sound rubrics are continuous, parallel, coherent, high descriptive, valid and reliable. Each of these terms is defined in [3]. This web resource also provides guidelines for developing rubrics, common errors in developing rubrics, and examples of good and poor rubrics.

For more on rubrics, I point you to [1], [2], and [3] for general information; to [4] for a computer science/programming project examples.

References

[1] Holzberg, C. "Assessment, Assessment Rubrics and Evaluation Guidelines," http://www.techlearning.com/db_area/archives/WCE/archives/evalguid.html

[2] International Society for Technology in Education, "Educational Computing and Technology Programs - Computer Science Rubrics," http://cnets.iste.org/ncate/n_cs-rubrics.html

[3] Performance Assessment Links in Science, "Rubrics & Scoring" http://pals.sri.com/guide/scoringdetail.html

[4] Becker, K. "Grading Programming Assignment using Rubrics," in Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education, page 253.

[5] Becker, K. "General Programming Assignment Rubric," http://pages.cpsc.ucalgary.ca/~becker/233/Asst/ProgramRubric.htm

Classroom Issues

Do Computer Games Have a Role in the Computing Classroom?

Henry M. Walker

For several years, I have been struck by two conflicting themes that arise regularly at conferences. Some speakers promote games as excellent motivators that provide opportunities to introduce modern technology and encourage creativity. Other speakers describe games as discouraging underrepresented groups and distracting students from fundamental principles, structures, algorithms, and methodologies. This column explores several common arguments on each side.

Due to space limitations, this column focuses on the use of games within introductory courses, although similar issues may apply at other levels as well. As with previous columns, I invite feedback on any side of the argument.

Types of Games

Of course, games encompass many styles and subjects. For example, games may be competitive or cooperative, be played by individuals or groups, and touch on numerous themes, such as adventure, education, social interactions, science fiction, violence, and sexual circumstances. Simulations sometimes are considered games as well. With this diversity, discussions of advantages or disadvantages

of games may apply only to some types or application areas; and one must be careful not to over generalize in making comments.

Positives

Several arguments are commonly cited for the use of games in the computing classroom. While the following list is NOT ordered according to importance, I have numbered the main points for later reference.

1. Motivation: (a) Some (many?) students find games very motivating, and (b) Many students have prior experience with a variety of computing games, so using games in courses may connect with students' background.
2. Fancy graphics can capture students' interest and imagination.
3. Games are often easy to understand, so developing programs that play games can highlight problem solving, data structures, classes/methods, and other high-level skills.
4. Games provide options for creativity in assignments, possibilities for extensions, and opportunities to develop projects through a sequence of assignments.
5. Games allow assignments to be described in layers, where a moderate level of functionality is required for a "C", additional features constitute a "B", and extensive refinements yield an "A".
6. Games provide opportunities for the early introduction of elements of modern technology, such as client/server computing, concurrency, and object-oriented programming.

Negatives

While games have various constructive elements within the classroom, various reports suggest that many positive elements also have counterbalancing negatives. Here are some commonly cited problems for the use of games. (Numbering is keyed to the points in the list of positives.)

1. Motivation:
   (a) Some groups, particularly women and other underrepresented groups, are often turned off by competitive games. Students in these groups often want an emphasis on socially-constructive applications. (See note 1.)
   (b) An emphasis on games may reinforce the popular misconception that video games represent a major component of computer science. (See notes 2 and 3.)
   (c) Games and game playing can be quite addictive, so emphasizing games in the classroom can reinforce anti-social behavior.
   (d) Assignments utilizing games can encourage distractions during class sessions, as students show off their programs.
2. Graphics:
   (a) Much class and/or student time can be devoted to graphics and I/O. If there is not a corresponding emphasis on HCI, game interfaces could focus on personal idiosyncrasies rather than principles and analysis.
   (b) Extensive time devoted to I/O can limit time available for such fundamentals as algorithms, data structures, and software engineering.
4. Extensions and Creativity:
   (a) Encouragement to add features may undermine a sense of writing to specifications and considering actual customer needs.
   (b) Options for extended functionality may encourage program bloat and unnecessary complexity.
5. When grades depend on multiple levels of functionality, true beginners (who often are students from underrepresented groups) can be at a significant disadvantage for the best grades and building self-confidence.

Overall, this discussion suggests strong counter arguments may apply to at least four of the six arguments cited previously in favor of the use of games in computing courses.

Additional Notes

The remaining pro-games arguments involve (3) high-level problem solving and (6) opportunities for the early discussion of modern technology. In both cases, other applications may be equally effective as games in addressing desired objectives. Sometimes a simple rewriting a problem can broaden its appeal. For example, consider the following problem for a CS1/2 course:

Original Problem

Use a random number generator and a conditional loop count the minimum number of tosses of a coin needed to get at least one head and one tail. Then embed this code in an outer loop that prints this minimum number of rolls for 20 simulated experiments.

While this problem provides helpful experience with random number generators, conditionals, loops, and simple simulations, my students found the problem somewhat contrived. The following restatement generated much more student interest.

Restated Problem

A couple decides to have children until they have at least one boy and one girl. Then theywill stop having children. Assume the likelihood of having a girl is 50% and the probability of the next child being a girl is independent of the genders of previous children. Use a random number generator to simulate the gender of a child, and a conditional loop to count the number of children for a specific couple. Then embed this code in an outer loop that prints the number of children for each of 20 couples.

While the two resulting programs are functionally

identical, students often found the context of family size considerably more interesting than flipping a coin.

Example A

As a second example, the new Java-based Marine Biology Case Study of the Advanced Placement Computer Science course utilizes simulation on several levels–not games–to provide a wonderful context for problem solving, data structures, and object orientation.

Example B

As a third example, the SIGCSE Project grant for a client/server framework for CS1/2 supports both games and multi-user applications—promoting the use of modern technologies in many contexts, not just games.

Concluding Comments

While space prevents this column from touching all arguments either for or against the use of games in computing courses, it suggests that some arguments for games have corresponding counter arguments, while other arguments apply to various simulations and other contexts as well as games. With the potential negatives—particularly related to the discouragement of women and underrepresented groups—this preliminary discussion might suggest applications other than games be used in courses; at least, instructors should be careful before incorporating games into their courses. Of course, faculty on either side of this debate might bring other points forward as well to swing the debate to one side or the other.

Endnotes and Bibliography

1. Negative perspectives by women of computer games are discussed in several articles in the *Women and Computing* special issue of the SIGCSE Bulletin *inroads*, June 2002.
2. Educational Foundation Commission on Technology, Gender, and Teacher Education, *Tech-Savvy: Educating Girls in the New Computer Age*, American Association of University Women Educational Foundation, 2000.
3. While gathering comparable statistics is challenging, here is one estimate of the relative size of the computer gaming industry. The Interactive Digital Software Association reported that "2001 U.S. sales of computer and video games grew 7.9 percent year-on-year to $6.35 billion, ..." (See http://www.idsa.com/2001SalesData.html). Also, the Information Technology Association of America combines information and communications technology (ICT) products and services within its definition of the information technology industry and reports that "U.S. spending in ICT has increased almost 70 percent since 1992, to almost $813 billion in 2001." (See http://www.itaa.org/news/gendoc.cfm?DocID=120). Putting these numbers together, computer and video games made up 0.78% of total IT sales for the year 2001.

Community College Corner

ACM Two-Year College Education Committee Report

Robert D. Campbell

In spring 2003, the ACM Two-Year College Education Committee organized a task force to produce a new information systems report for the two-year colleges, as both an update to its 1993 *Computing for Information Processing* report and a bridge to the *IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems* report. The *Guidelines for Associate-Degree Programs in Information Systems* report - now available in reviewer's draft format - is the result of the work of the task force. The report provides guidelines for the implementation and sustenance of an associate-degree program in information systems (IS). The *Program Considerations* section provides context for two-year college programs in the field of IS; the *Body of Knowledge* section addresses curriculum content. The content is separated into course-based groupings.

The report provides a framework for the development, support, and updating of associate-degree programs in the computing discipline of Information Systems. Graduates of such programs will be able to transfer into the upper division of an IS baccalaureate-degree program or pursue a career in the IS field in an entry-level position. The guidelines will assist colleges in educating potential IS workers with technical computing competencies, as well as necessary workplace skills. Graduates of programs crafted from the guidelines should be able to:

- solve problems within the context of business systems;
- communicate effectively within an organization;
- work productively in team and individual settings;
- demonstrate professionalism; and
- adapt to emerging technologies and new environments.

Colleges should make students aware at the onset of their studies of the distinctions between career and transfer programs, the academic requirements of each, and the associated employment options. Students graduating from a career-oriented associate-degree Information Systems program may enter jobs with titles such as:

- database programmer, database administrator, database manager, records manager;
- systems administrator;
- technical writer, documentation specialist;
- technical support specialist, help desk specialist;
- e-business specialist, e-commerce specialist;