Teaching with Games:

The Minesweeper and Asteroids Experience

Abstract

The value of games as a vehicle for teaching concepts while inspiring students is now well accepted at almost all levels of education. Video games, arcade and computer games are rarely given the same kind of attention. This paper will describe the value of computer games as a motivator and some of the benefits that can be realized by using known computer games as programming applications, even in the first year of a CS program. The use of two such games as assignments in CS1 and CS2 is outlined and some feedback on the experience is offered.

Introduction

The first year of a CS program offers many challenges both for students and instructors, largely because students must get accustomed to University in general and workload and lectures in particular. Courses at the post−secondary level often take a much more self−directed approach to learning than the one to which new students are accustomed. Lectures come in a full range of approaches and students are often expected to cope with vastly different teaching styles on their own. Instructors in some classes are truly spellbinding and have justly earned the admiration and respect of their colleagues while in other classes that perception exists solely in the mind of the instructor who seems to believe that emitting utterances is all that is needed in order for students to learn. Most of us fall somewhere in between. Almost all of us would be pleased to have a few new tricks up our sleeves that we could use to inject some new life into our classes. The following is one of those tricks − almost guaranteed to get a positive reaction.

Motivating Students

For many students the first year sets the tone for the rest of their university career and in some cases, affects their professional career as well. For better or worse, first year instructors can have a lasting impact on the students they teach. Computer Science is often described as one of the most demanding disciplines on campus. If students can be inspired and excited by the things they learn and the work they do early on, these attitudes will often carry them through the difficult times and the "all−nighters". It has been said that much of what is learned in computer science is learned by doing, particularly when it comes to programming, so if students can be motivated to do more it seems reasonable to conclude that students will end up learning more as well [RUBE99]. While *having fun* is not typically high on the list of teaching goals, its value should not be underestimated. Students who are having fun work harder, longer, and are more apt to expand on what is taught than those who simply wish to get it over with and pass the course.

CS Students in the 21st Century

Although many CS1 courses offered throughout North America assume no prior programming experience on the part of the student, almost all students entering college now have some form of experience with using computers. They were born into a world of PCs and video games. They went through school with varying amounts of exposure to computers used for educational purposes. They've used word processors, surfed the net while doing research for projects, and been exposed to all manner of courseware designed for learning and practice. The wide−spread acceptance of the value of experiential learning has resulted in the emergence of many fine examples of educational programs that inspire and motivate [SOLO91][RUBE99], and the value of games in particular as vehicles for teaching various concepts and skills is both well documented and accepted [FALT99]. In fact, the vast majority of the educational software available today is presented in the form of games of one sort or another.

Until recently, the use of educational software to actually teach computer science has lagged far behind its use in other disciplines, but the scene is changing. With the accessibility provided by the Internet, more and more instructors are realizing the value of animations for presenting various concepts [FALT99] and many have combined this approach with the use of games.

Games Are Not Just For Fun

While educational games have gained wide acceptance and a certain degree of respect, those produced for entertainment are generally believed to be devoid of educational value. These games are still largely ignored by the academic community and are often treated with disdain. To many academics, the entire games industry is considered to contain little scholarly merit, and computer games are generally not believed to be the result of serious work or worthy of serious attention. The image of undisciplined hackers producing computer games in their basements lingers. However, while we were engaged in perhaps more noble pursuits, the games industry has grown into a multi−***billion*** dollar industry [GORR00]. It remains the fastest growing software segment and it is estimated that 80% of all after purchase hardware upgrades are for the purpose of game play [TEPP99].

To try and ignore the influence of such a powerful force is a mistake. It may even be one of the factors in the perception (echoed by many students) that CS departments are out of touch with "the world out there". Like it or not, the games industry influences many of those who enter our field. There is evidence suggesting that the nature and target audience of the majority of the commercially available games may even be a factor in the decline of women pursuing CS degrees in the last 10 years [GORR00]. The reality is that most of our current CS students grew up with computer and video games. In fact, when students are asked, many will say that game playing is what got them interested in computers in the first place. Regardless of where they end up in their careers, many start off with a desire to become games designers. We can tap into this energy and use it to our advantage.

Some colleges are already beginning to view the games industry in a new light. Senior level courses that deal with games design are being offered at a few institutions and instructors are realizing the potential that games hold for the integration of almost ALL of the concepts and techniques taught in a typical CS degree program [JONE00]. The University of Calgary (Canada) is offering a capstone course in games design for the first time in the winter session of 2001 that includes direct input from a successful games company, and already there are first year students planning their next three years specifically around this one, senior level offering. This institution will also offer a degree concentration in Games Design beginning in September 2001 and student interest has been high. The motivation to succeed seems made to order.

Back to the First Year in CS

First year CS classes are typically populated with students who come into first year with a wide range of experiences and abilities. Meeting the needs of all is a tremendous challenge. There are many skills to acquire and concepts to conquer and the less experienced students often feel overwhelmed with the sheer volume of work and the time commitment we require of them. As instructors, we naturally want all of our students to master the concepts and be prepared to succeed in later years but we also want them to finish first year with a sense of accomplishment and confidence. In a perfect world, they would be inspired, excited, and enter second year feeling enthusiastic about their careers as CS students.

We try many different approaches to inspire and hold their interest. Some instructors have spent a great deal of effort in designing assignments that integrate the concepts taught in class. Occasionally, when we find a good one (or a whole set), we use the same assignments again and again. The well–known John Conway creation: The Game of Life has withstood the test of time for good reason. It exercises several useful concepts (like the use of random numbers, 2–D arrays, functions). It is well within the realm of "do–able" by a first year class with little programming experience, and it's ***fun***.

Unfortunately though, most programming assignments are what most students would classify as boring: mathematical problems excluding games (for a discussion of some more interesting mathematical games, see [BERL00]), business oriented problems, text processing, list processing, and others all succeed in helping students to learn the concepts but few are met with any real enthusiasm and fewer still inspire true creativity. Add to that the general complexity and difficulty of learning to program in the first place [MOSE97] and it's not hard to see why many students are somewhat less enthusiastic at the end of their first year than they were at the start.

Games to the Rescue

While certainly not the solution to all our problems, computer games and games design provides us with a tremendous untapped resource of ideas that will inspire. By mixing a few games in with our other tried and true assignment offerings, most instructors will find that students respond with new energy and enthusiasm. Many instructors are now using games of their own creation in both junior and senior level courses [ADAM98] [MOSE97] [JONE00] with encouraging results.

Games are *fun*. This one fact alone sets them apart from the majority of the projects assigned. But computer games have other advantages. They hold an almost universal appeal for the current generation of students. *They are cool* (Who of your students would recognize Alan Turing or Donald Knuth? – but try asking them about "Asheron's Call" or "Vampire: The Masquerade". They can discuss the relevance of movies like "12 Monkeys" and "Matrix" with an insight that demonstrates a much more sophisticated grasp of the social context of our discipline than most of us realize.) It is the games developers rather than the revered cornerstones of our discipline that our current students aspire to be like. Computer games have become a significant part of today's popular culture [FITC00] [BOWM82]. ***We can use this.***

Games are undeniably a non–trivial application for computer programs and so it provides us with a ready–made connection between what we are trying to teach them and something they already know (and care about). Games come in a full range of complexities, adaptable to all levels of experience. Many forms of games lend themselves nicely to (nay, *insist on*) a modular or object oriented approach for their solution, allowing us to form a natural match between the problem and the concepts. Virtually all have a graphical output, which even when crudely implemented can provide easily comprehensible feedback to the programmer. For example, if the game involves drawing objects on a grid and then moving them, students can actually watch their indexing efforts and loops go awry! With games, students have a clear and identifiable goal and because they already know how it's supposed to work and find the task at hand interesting, they are more motivated to actually get their programs running – not just for the marks but also for their own satisfaction.

The Minesweeper Experience

In the winter and fall of 1999 we had one assignment (out of 6) in each of the introductory courses that was more involved than the rest. Students were given three weeks rather than the usual two to complete this assignment. In CS1 (CPSC 231) students were to implement the Game of Life (to practice with random numbers, 2–D arrays, and functions) and in CS2 (CPSC 233) they were given a drawing program (OO, inheritance). Students generally found them interesting but were not particularly inspired. In the following year the assignments were changed. In place of The Game of Life, the CS1 class created an ASCII version of "Minesweeper", the well–known Windows game, and in place of the drawing program in CS2 they created a simple version of "Asteroids!". The difference in their responses was striking. Students were enthusiastic, they worked much harder and some truly inspired creations were submitted. During the weeks they were working on these projects, they frequently came by the instructor's office for the sole

```
MINESWEEPER (with the "cheat code" turned ON for testing purposes):

gromit% ./mines -d
GAME OF MINESWEEPER

Enter play level:
b : Beginner [10X10]
i : Intermediate [16X16]
e : Expert [16X30]
>> i
    1 2 3 4 5 6 7 8 910111213141516        1 2 3 4 5 6 7 8 910111213141516
 1  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     1        1 1 1 1 X X 1       1 X 2
 2  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     2  2 3 2 3 X 2 1 2 2 2 1 2 1 2 2 X
 3  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     3  X X X 4 X 3 1 1   1 X 3 X 2 1 1
 4  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     4  3 5 5 X 2 2 X 1 1 2 2 4 X 3
 5  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     5  1 X X 2 1 1 1 1 1 X 1 2 X 2
 6  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     6  1 3 4 3 1     1 2 2 1 1 1 1
 7  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     7    2 X X 2     1 X 2 1
 8  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     8  1 3 X X 2     1 2 X 1
 9  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~     9  X 3 3 2 2 1 2 1 2 1 1
10  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~    10  2 X 1   1 X 2 X 2 1 1 1 2 1 1
11  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~    11  1 1 1   1 1 2 2 X 1 1 X 3 X 2
12  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~    12  1 1           1 1 1 1 1 3 X 2
13  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~    13  X 1                 1 1 2 2 2 1
14  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~    14  1 1                 1 X 1 2 X 2
15  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~    15  1 1               1 2 3 2 4 X 3
16  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~    16  X 1               1 X 2 X 3 X 2

Score: 0 Mines Left: 40
[i,j]: 12 4
```

purpose of showing off some of the better features of their programs. Many students were inspired to teach themselves about tools and approaches that weren't even part of the curriculum just because they wanted their games to be better. The additional marks they got for the extra work didn't even begin to cover the extra time they spent but they didn't care – they were happy to do it.

An unexpected benefit of this approach has been that word is now spreading through the student body that assignments in first year are fun and interesting. Students approach the instructor before the semester begins to find out what they ***get*** to do this term. They look forward to the assignments. We've even had some senior students comment that they wish they could take the first year courses now.

Writing games inspires and excites students. Using games they have played or heard about has several added benefits. "Minesweeper" is a game that almost all students have tried – ***it is part of their PC.*** This puts the problem they are to solve in a context with which they are already familiar [SOLO86]. While the implementation is no more difficult than the Game of Life (it is in fact marginally simpler), the difference is that they knew this game when they were mere computer users. Writing it themselves and seeing their efforts behave just like the real thing forces them to cross a significant perceptual boundary:

they become the "creators". They go from experiencing the "magic" to being the "magicians". There is a ***real*** connection here and it doesn't feel like school.

The Asteroids! Experience

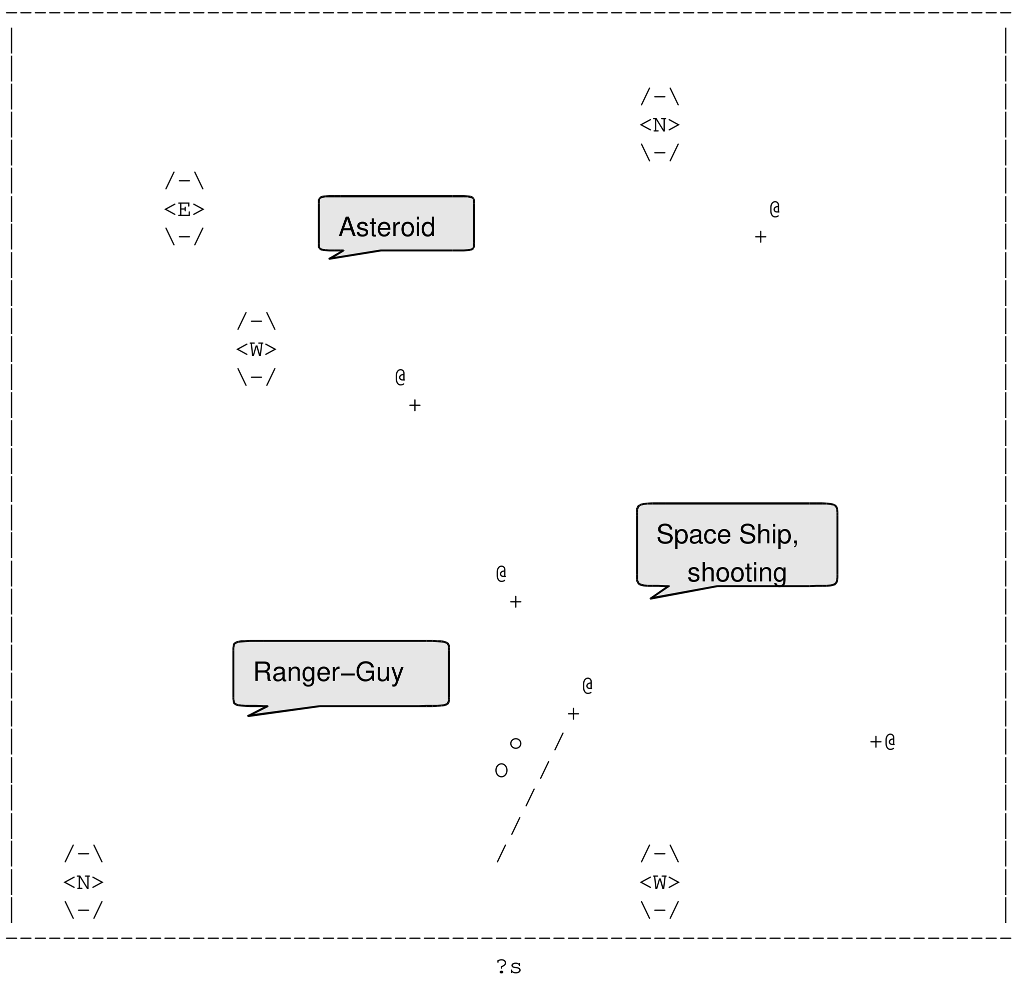
In our CS1−CS2 pair of courses we currently teach C++ on SUN Workstations running UNIX. This makes access to reasonable GUIs rather difficult. In order to make this project one at which all students could succeed the game was somewhat modified. We displayed the game using ASCII graphics (with the help of a simplified version of the "curses" library) and game play was turn−based rather than real−time. At each point in time the static display is presented and the player may choose to move, shoot, move−and−shoot, or pass (<return>). The request is received and processed, and then all objects are moved according to their pre−determined behaviours before the next display.

As stated before, students in first year often begin with dramatically different backgrounds and abilities so in order to address the needs of all, several levels of completion (stages) were described in the assignment specifications. The minimal requirements could be awarded a maximum grade of C+ and required that students implement just the asteroids themselves. Asteroids were to be placed at random and simply moved E−W or N−S and never changed direction (so they had to wrap around the display). A collision between two caused both asteroids to be destroyed completely. The B−stage solution required the implementation of a "Ranger−guy" that could be moved by the player and that could shoot (and destroy) the asteroids. The full−blown solution introduced randomly placed spaceships that moved towards the "Ranger−guy" and fired upon the ranger. All three kinds of objects had to look different but this could be done with a few cleverly placed ASCII characters. Since this assignment involved the use of inheritance, each of the "objects" on the screen had to be implemented as a class derived from a common base class. The concepts related to inheritance, which typically cause confusion amongst many students seemed to pose little difficulty on this project because the C++ objects related so closely to the tangible ones they could actually see on the screen and the inheritance pattern of attributes and behaviours appeared obvious to them. Students were provided with a pre−written base class that they could use if they wished and they were given an introduction to "game−AI"s including a rough outline of what the game AI might do in this game.

The students loved it! It was the first time we had ever experienced students continuing to work on their projects well ***after*** the assignment had been submitted for grading – just for the fun of it. The admittedly crude GUI seemed to do nothing to detract from their enthusiasm for the task. They would discuss their latest features and proposed enhancements with each other at every opportunity. In fact, some later admitted that their other courses were allowed to slip a bit during those three weeks because of the time they willingly spent on this one. While I would not advocate a survival of the fittest approach to courses, there is no higher praise than to have students choose your course over all their others.

In each class (especially large ones) there are always a few students of superior ability who are often bored by the work assigned to the class as a whole. It has been the author's habit to try and address this by including a bonus category with each assignment. These are enhancements that students can implement for extra marks but are not required for an "A". I occasionally include very

ASTEROIDS! ("Ranger" is made of O's; note the "spaceship" shooting at the "Ranger-guy")



advanced enhancements as suggestions, usually without any expectation that anyone will actually rise to the challenge. For the Asteroids assignment, the bonuses included:

1. [2 points] Allow asteroids and spaceships to appear at random once the game has begun.
2. [2−3 points] Detect "Game Over": when Ranger dead; or when all asteroids and aliens gone.
3. [2 points] Keep score. (one point per thing or allow a certain number of points for certain kinds of things).
4. [2−5 points] Allow different levels of difficulty (different #'s of things, different ranges, etc.)
5. [2−4 points] Asteroids break into two when hit − both of which are smaller and go off in different directions.
6. [2 points] Set up "key bindings" so player can use the arrow keys, etc.
7. [4−? points] Add sound.
8. [5−? points] Make it real−time
9. [5−? points] Implement a full GUI
10. [5−? points] Make it 3−D

A majority of the class implemented the first 4 bonuses and nearly 25% tried to implement some or all of 5−8. Doing the key−bindings, running the game as an event−driven application, and incorporating sound were all things not covered by the curriculum and students implementing these were left to puzzle out how to do this on their own.

After the three weeks were up, we had one submission that had implemented ALL of the bonuses. In addition, this game was played from the perspective of the ranger − the screen displayed the scene as the ranger would see it while sitting in his own ship traveling through space − in full colour no less! While this student is a truly exceptional individual (he'd had no prior experience with either UNIX or Linux systems before beginning first year university, and taught himself − X−Windows from a book while writing his game all in the three week time frame) − he credits this course and this assignment in particular for his decision to stay at this institution. His friends at some of the ivy−league colleges weren't doing anything nearly as interesting (no offense intended).

Conclusion

This experience has been most gratifying. Students pushed themselves harder and learned more in the process than they would have had this been a more traditional assignment like a bank account or payroll system. They gained experience with all the concepts that the assignment was designed to exercise, and in many cases exceeded expectations, learning concepts more thoroughly than they would have had they not been so keen. Everyone wins. The students learn the concepts they need to continue on (so the instructor wins) and they are excited and look forward to what the following year will bring (they win too). Using known games has provided a hook to capture their imagination and their energy. They already know what it's *supposed* to do and have plenty of ideas of what they'd like to be able to make it do. Through the use of these games we have also been able to expand our foundation in first year and give them a glimpse of what is possible now and what they may be able to make possible in the future.

Appendix I: Games to try:

Game	Simple Solutions [CS1]	Intermediate Solutions [CS2+]	Advanced Solutions [3rd Yr+]	Can be Staged (varying levels of completion)	Some Possible Advanced Skills & Techniques
Adventure	√	√	√	√	*limitless*
Amnesia ("Memory")	√	√		√	
Asteroids!		√	√	√	
Attaxx ("Reversi")	√	√	√	√	
Battleship	√			√	AI Search
Blocks / Break Out		√	√	√	Trajectories
Box World	√	√	√	√	
Checkers	√	√	√	√	Decision Trees
Connect−4	√	√		√	
Doom		√	√	√	*limitless*
Frogger		√	√	√	
Gold Monkey	√	√		√	
Donkey Kong		√	√	√	*limitless*
Joust		√	√	√	
Leap Frog	√	√		√	
Life	√	√	√	√	Genetic Algorithms?
Lunar Lander	√	√	√	√	Physics, Diff. Eqn.
Mario Bros.		√	√	√	*limitless*
Minesweeper	√	√		√	Flood Fill
Missile Command		√	√	√	Physics, Diff. Eqn.
Pac Man		√	√	√	Physics
Pinball		√	√	√	Trajectories

Pong		√	√	√	Trajectories
Slither / Nibbles / Snake	√	√	√	√	Genetic Algorithms?
Solitaire	√	√	√	√	
Space Invaders		√	√	√	
Tetris	√	√	√	√	

References

[ADAM98] Adam, Joel C. "Chance−It: An Object−Oriented Capstone Project for CS−1", Proceedings of the 29th SIGSCE Symposium in Computer Science Education,1998, Atlanta, GA. p10−14

[BERL00] Berlekamp, Elwyn R., John H. Conway, Richard Guy, "Winning Ways for Your Mathematical Plays" Vol. 1, 2nd Ed. 2000, A.K.Peters, Natick Massachusetts

[BOWM82] Bowman, Richard F. Jr. "A 'Pac−Man' Theory of Motivation: Tactical Implications for Classroom Instruction", Educational Technology, Vol.22 #9, Sept. 1982 p14−16

[FALT99] Faltin, Nils "Designing Courseware on Algorithms for Active Learning with Virtual Board Games", Proceedings of the Conference on Integrating Technology in Computer Science Education, June 1999 Cracow, Poland p135−138

[FITC00] Fitch, Crosbie, "Cyberspace in the 21st Century: Mapping the Future of Massive Multiplayer Games", Gamasutra (http://www.gamasutra.com) Jan. 20 1999

[GORR00] Gorriz, Cecilia M. and Claudia Medina, "Engaging Girls with Computers through Software Games", Communications of the ACM, Vol.43 #1 Jan. 2000 p42

[JONE00] Jones, Randolph M. "Design and Implementation of Computer Games: A Capstone Course for Undergraduate Computer Science Education", Proceedings of the 31st SIGSCE Symposium in Computer Science Education, March 2000, Austin, TX p260−264

[MOSE97] Moser, Robert, "A Fantasy Adventure Game as a Learning Environment: Why Learning to Program is so Difficult and What can be Done about it.", Proceedings of the Conference on Integrating Technology in Computer Science Education, June 1−5, 1997 Uppsala, Sweden p114−116

[RUBE99] Ruben, Brent D., "Simulations, Games, and Experience−Based Learning: The Quest for a New Paradigm for Teaching and Learning", Simulation and Gaming, Vol. 30 #4, Dec. 1999 p498−505

[SOLO86] Soloway, Elliot, "Learning to Program = Learning to Construct Mechanisms and Explanations", Communications of the ACM, Vol.29 #9, Sept. 1986 p850−858

[SOLO91] Soloway, Elliot, "How the Nintendo Generation Learns", Communications of the ACM, Vol.34 #9, Sept. 1991, p23

[TEPP99] Teppe, Michelle, "The Power of Play", netWorker: The Craft of Network Computing Vol. 3 #4 1999

Minesweeper Assignment:

(note: both sets of notes are password protected. For access please send email to the author.)

http://www.cpsc.ucalgary.ca/~becker/231/Asst/Functions/MineSweeper.html

Asteroids Assignments:

http://www.cpsc.ucalgary.ca/~becker/233/Asst/BigGame/Asteroids.html