ABSTRACT
This case study reveals Daniel Webster College's experience igniting student enthusiasm for the computer sciences by combining complex game development with innovative classroom management techniques during the freshman year. In this paper we discuss CS1 and CS2 game motif exercises which have dramatically accelerated student progress at DWC by stimulating independent study, cooperative learning, rapid knowledge assimilation and teamwork. Accelerated learning at DWC appears possible because game development creates enthusiasm and a strong work ethic among DWC students. We share student testimonies of their extraordinary freshman year experience.

Categories and Subject Descriptors
D2.0 Software Engineering, D3.2 C#, C++

General Terms
Design

Keywords
Software engineering, game development, GM-method, cooperative learning, independent study.

1. INTRODUCTION
At Daniel Webster College students create complex graphics-based games throughout their college experience. This philosophy begins in the freshman year during which students develop complex multi game, multi client, client-server gaming systems with centralized control and player asset management subsystems. (Figure 1) This level of accomplish by teams of 4 programming novices in the CS2 course is possible because of the enthusiasm young people have for game play and their interest in pursuing employment in some facet of the gaming industry.

Many educators and corporations hope that student attraction to games can re-energize interest in the computer science field. This has been confirmed at DWC which has experienced a 240% increase in enrollments since the adoption of complex game projects and the creation of the GM-method educational method discussed in this paper.

We have also found that the complexity of game development provides an opportunity to improve the preparedness of graduates for roles in engineering and information technology careers. This is particularly important to parents of DWC students. They are often surprised by the rigor of a program which has game development as its central focus. However, the successful placement of DWC graduates in top graduate schools, government laboratories and corporations allays parental fears and undoubtedly influences their willingness to enroll their children in one of the DWC computer science programs. Thus, this paper presents a case study revealing one potential strategy for attracting students to the computer sciences.

Figure 1
Freshman 2007
Limehouse Casino System
Multi-Game System with Account Management and Control

2. THE CS1 COURSE
The CS1 programming exercises in C# .Net (formerly C++/MFC) are primarily designed to develop basic programming skills at an accelerated rate. All the special CS1 exercises are “game like” in as much as they are visually appealing and targeted toward having students solve the types of complex problems encountered in game development. Within the first two or three weeks, beginning DWC programmers in CS1
are using nested loops, arrays, managed code, and a whole range of Object Oriented principles to attack very knotty but visually interesting problems.

Even the Hello World exercise (Figure 2) assigned in Week 1 is utilized to motivate students to quickly reach beyond beginning concepts to find programming solutions that create more elegant and efficient solutions. All CS1 exercises are designed to motivate independent research, to foster rapid absorption of knowledge and to stimulate immediate application of advanced concepts. In Week 3, the Cellular Automaton (CA) exercise (Figure 3) is used to encourage the students to craft advanced solutions. For example, once a CA solution is found using two arrays the students are encouraged to craft a solution using one single dimension array.

Figure 2:  
CS1 Hello World  
Week 1 Exercise

There currently exist 14 special CS1 special exercises. Approximately half involve console output and the other half visual output. Many are illustrated below. Our experience with the 14 existing CS1 special exercises has proven that this number should be greatly expanded. Currently, a typical CS1 programming session lasts 2 weeks. During that one or two week period a DWC student will be charged with programming one or two special Game motif exercises along with three to five traditional exercises selected from the excellent programming textbook used in the course. [2]

Figure 3: 
CS1 Cellular Automaton  
Week 3 Exercise

Under the GM-Method developed at DWC, students are shown no faculty code solutions nor are faculty lectures given on any technical subject until the students have independently crafted their own solution. The students are provided a written specification and an executable for each special exercise. They are required to independently research, master and apply whatever knowledge is necessary to complete the CS1 exercise. We believe this fosters independence, develops self confidence and trains the students to function in a manner similar to that which will be required of them when they are employed.

What role do faculty members then play when using the GM-Method in freshman courses? Faculty members work with students individually in a laboratory setting. Difficulties, code solutions, software engineering principles and problem solving are discussed mostly one on one. Students are also encouraged to collaborate with one another. We have found that students, in fact, prefer to discuss and
solve problems amongst themselves rather than with a faculty member. Under the GM-method the students quickly exhibit remarkable independence and rarely ask of or require guidance from an instructor. [4]

Examinations are also unnecessary under this method. In a laboratory setting faculty are continuously reviewing student programming exercises for functional completeness, good coding practice, adherence to professional documentation standards and adoption of sound software engineering principles.

Students seem to prefer being graded on the number, quality and software sophistication exhibited in their programs. Higher scores, result from the use of advanced programming concepts. However, we believe students are primarily motivated by the game like exercise rather than because they want a higher score. For this reason we make every attempt to create exercises which are relevant to game creation. [5] The rectangular, circle and pixel collision exercises exemplify how this is done (Figures 4, 5) in the CS1 course.

Figure 4

GM-Circle Collision

GM-Rectangular Collision

Figure 5

GM-Pixel Collision Game

Pixel collision is one of the games that very few first semester DWC students can complete successfully under the time constraints given to the students. Nevertheless, we assign the game because it serves the purpose of showing the students that game programming is a complex, mathematically rich and challenging undertaking.

3. THE CS2 COURSE

The development of a C# .NET multi-game client-server gaming system with centralized control and account management is a complex undertaking under the best of circumstances. Nevertheless, six years of experience has shown that it can be done; however, the game programming exercises in CS2 must be carefully crafted and sequenced. The exercises in CS2 must include much greater functionality and, of necessity, must be much more complex than those found in the CS1 course. The CS2 exercises also must build knowledge in a manner that supports rapid application development (RAD) of a video casino system. [6]

The CS2 course involves 10 to 12 Sessions which typically last from one to three weeks. Each session centers around one special programming exercise or a project integration activity. Collaboration is much more evident in the CS2 course since CS2 students always work in teams involving four or five young men and women. [6] Every student in CS2 will not solve each exercise. The team members will typically divide the various subsystems amongst themselves in order to adhere to the very tight development schedule.

Figure 6 illustrates one of the more complex CS2 exercises. In Session #8, students at DWC are required to implement a server side Casino floor management subsystem with client control and player asset tracking. This two week exercise requires students to learn relational data base concepts, master the implementation details of ADO.NET, deploy a Microsoft Access or mySQL data base, implement log files, parse TCP/IP packets and finally provide a casino manager appropriate management functions. In the case of Session #8 each member of the team will specialize in some aspect of the task. One might specialize on the client server problem and another may become the data base expert.

So while the Session #8 exercise would be a monumental undertaking for a single student, it is quite manageable assignment as a team effort. The knowledge gained through the exercise is also directly applicable to the system they will create for their final project. Therefore, DWC students will often develop solutions which far exceed that specified in the Session specification. Students look ahead toward the functionality they want to introduce into their own game system. We have found that students prefer creating their own technical solutions and developing their own artistic ideas.
We believe that some of our instructors who are new to game complexity or those inexperienced in rapid application development might be more comfortable using the CS2 exercises if they are simplified and much more numerous (Figures 6, 7). Therefore, we expect to increase the number of CS2 exercises. We also will develop project motifs for CS2 that do not involve the casino theme. We believe student groups under-represented in the computer science field could be attracted to the discipline if we develop game motif exercises that resonate with a broader range of ethnic, gender and cultural populations.

Eventually, we will program each exercise with two different code implementations, one using elementary code constructs and the other using advanced code concepts. This will provide an instructor greater flexibility when using game based exercises. This expanded set of exercises will also provide DWC enough programming material to create a game based course sequence that entails more than two semesters. This could help us reach freshman student populations that move at a slower pace.

4. WHAT ABOUT THE SLOWER STUDENTS
We believe a collection of exercises with more primitive code implementations will preserve the attractiveness of the game motif in the beginning exercises without requiring every student to learn at an accelerated rate. For example the Hello World program depicted in Figure 2 could be crafted with simple write statements and escape characters. The number of lines of code would expand greatly, but the implementation would be easy to craft by students who are getting off to a very slow start.

We have found that approximately 15 to 20% of freshman students at DWC are unable to learn under the GM-method discussed in this paper. Either the pace is too fast, the complexity too great or the emphasis on independent study and massive knowledge assimilation too demanding. Nevertheless, even weaker students can survive the freshman year because of the cooperative learning atmosphere created in CS1 and the team based scoring of the CS2 project. This gives some of the slower starters an opportunity to develop roots. On the other hand, team based activities allow some students to incorrectly conclude they can succeed in the field without commitment or fundamental skills.

While a lack of basic skill or commitment is observable during freshman team projects, it becomes very evident during the first term of the sophomore year. During the fall term every sophomore must independently develop a fully functional WWII German
encryption system. The ENIGMA is recreated in assembly language on a 16 bit microprocessor called the lc-3. Students who struggled in their freshman year are unlikely to survive this course and we recommend that these students consider a less technically demanding course of study.

Thus, we have not found a completely satisfactory solution for the slower students.[7, 8] We feel we must keep the pace of the freshman courses brisk and demanding for the majority. Passion seems to be the most important success factor since it is usually correlated with a strong work ethic. [11] However, even desire is sometimes insufficient. Problem decomposition skill, logical thinking and persistence are all essential. [12] As a general rule we do not encourage continuation in a computer science major unless passion, work ethic as well as problem solving skills are evident. Students at DWC with these attributes quickly become independent learners and adept at deploying knowledge on an accelerated basis. Since the workplace requires these attributes we do not feel it is appropriate to provide encouragement to students who have little hope of developing these skills.

Even though very demanding the DWC classroom strategy is based upon a large body of academic research which supports the notion that non-traditional classroom methods can radically improve student performance. [9, 13]

5. Socratic and Cooperative Learning Methods
R.L. Moore was one of the 20th Century’s strongest and most successful proponents of a Socratic method called the discovery method. [10] Well over 1600 Ph. D mathematician descendants have emerged to be a part of the enduring legacy of Professor Moore’s work. During the 40 year period 1915-1954 descendents of R. L. Moore were producing journal articles at a 50 to 150% higher rate than graduates from virtually every other educational institution [Jones] How was this possible? Moore summarized very succinctly the philosophy leading to this amazing performance by his students: “That student is taught the best, who is told the least.” [3] Under the discovery method, “the instructor plays the role of coach, mentor, collaborator, guide, and occasional cheerleader”. [13] We have incorporated many these ideas into the GM-Method.

Cooperative learning is a defined instructional design in which small groups work together toward a common goal. [1] In the case of the GM-Method the common goal is the development of a complex software system. In more traditional classrooms settings information is developed and analyzed by team members for presentation to other classmates and an instructor. Nevertheless, in both cases, the goal of the cooperative learning classroom is to develop student dialog which increases mastery of course material. Our experience confirms the truth discovered by many other researchers. Students who discuss and explain solutions to one another learn by doing so. [16]

6. STUDENT RESPONSE TO THE EXPERIENCE
Student surveys (Table 1) yield interesting insight. They highlight student perception of learning and retention of course content, as well as reveal the student perception of effort expended, and the motivation to perform. Approximately 100 freshmen have studied beginning programming under the GM-method. While the sample size is small, the student feedback has always been encouraging to us. In the most recent survey the freshman class provided the following responses.

<table>
<thead>
<tr>
<th>Table 1: Student Opinions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparison to Traditional Classroom</strong></td>
</tr>
<tr>
<td>Expended More to Much More Effort</td>
</tr>
<tr>
<td>Learned more to much more</td>
</tr>
<tr>
<td>Liked Focus on Real Product Development</td>
</tr>
<tr>
<td><strong>Confidence &amp; Content</strong></td>
</tr>
<tr>
<td>Improved Confidence in learning</td>
</tr>
<tr>
<td>Absorbed great quantities of knowledge</td>
</tr>
<tr>
<td>Increased Confidence in Teamwork</td>
</tr>
<tr>
<td><strong>Motivation and Preferences</strong></td>
</tr>
<tr>
<td>Material Mastered is very useful</td>
</tr>
<tr>
<td>Worked hard without examinations</td>
</tr>
</tbody>
</table>

7. CONCLUSION
Gaming is clearly a magnet for young people. Exercises and projects with game themes keep students interested in their CS1 and CS2 courses. The strong work ethic that emerges clearly has a direct impact on student success. Passion about games creates an extraordinary incentive to succeed.

The demanding nature of game exercises and gaming system development is a great benefit to instructors at DWC. Strong problem solving skills and
rapid absorption of advanced programming concepts are a pre-requisite to student success. Nevertheless, novices seem to rise to the demands when gaming is involved. Our experience reveals that programming exercises based on banking, administrative or manufacturing applications simply do not create the energy or enthusiasm necessary for accelerated learning.

The CS1 and CS2 exercises are yielding encouraging results; however, we believe that much work can be done to broaden and expand game motif exercises so they can be used to attract ethnic, cultural and gender populations which are traditionally absent from the computer science field. It is also apparent that there is value in creating solutions with both simple as well as sophisticated code constructs. A minority, but significant percentage of freshmen, cannot learn by reading finished, polished and elegant programming solutions which omit all of the thinking, experimentation and test code that precedes the creation of faculty solutions or solutions seen in textbooks.

Finally, we believe that game themed exercises alone are not sufficient to cause accelerated learning and extraordinary outcomes at DWC. The work of researchers in non-traditional instructional methods has been critically important to our efforts. The GM-method, requires a significant departure from tradition, but the research of others encouraged us to believe far more in our students than in the importance of our own skill in the classroom.

8. REFERENCES


