Nifty Assignments: A Team & Project Oriented Undergraduate Computer Graphics Course

Ronald Marsh, Ph.D.
Computer Science Department
University of North Dakota
Grand Forks, ND
rmarsh@cs.und.edu

Abstract

This paper discusses the requirements, course content, student assignments, and outcomes of a senior-level computer science graphics course (CSci-446 Computer Graphics I) offered by the Computer Science Department at the University of North Dakota (UND). The course is a typical lecture styled course where theoretical topics are covered in class and where homework assignments center around the incremental development of a semester long and team oriented project (four minute animation) using the OpenGL graphics API, the RayGL API, and the Persistence of Vision Raytracer (POV-Ray). The overarching goal of the course is for students to master the theoretical concepts of computer graphics and to gain experience with the OpenGL API. However, OpenGL alone is not able to produce scenes with the realism required by ventures such as the movie industry. For such ventures raytracing is popular. However, there is no accepted standard mechanism that would allow programmers who are already familiar with OpenGL to smoothly transition between raster graphics and raytracing. Thus we created RAYGL, an OpenGL like API that allows the OpenGL programmer to easily generate Persistence of Vision Raytracer (POV-Ray) SDL files from OpenGL and OpenGL Utility Toolkit (GLUT) programs. RAYGL allows our students to easily extend their OpenGL project such that realistic animations are possible. Therefore, as the semester progresses, each student contributes to the team's animation by developing a specific component (or character) for the movie using OpenGL. Towards the end of the semester each student should have a component that includes a texture mapped 3D solid component, that exhibits some form of 3D motion, and that is lighted by ambient and specular lights. At this point the teams merge their individual components into a single executable culminating in a 4 minute computer animation. However, as noted above, the rendering quality is really not very good. So the teams then incorporate the RayGL API into their code; rerun it producing thousands of scene description files. They then run POV-Ray on our cluster to render those scene description files into PNG files. The PNG files are merged using MEncoder creating an AVI file that exhibits near "Hollywood" quality. Prior to developing animations, students were required to develop a computer

game as their project, however several issues surrounding the development of a game became apparent and the project component was changed to require a computer animation instead. While our experience in requiring animation projects is limited, our opinion is that the change improves the students understanding of graphics, provides another perspective of the application of software engineering principles, and exposes then to parallel processing.

1 Introduction

This paper discusses the requirements, assignments, and outcomes of a senior-level computer science graphics course offered by the Computer Science Department at the University of North Dakota (UND). The course of interest here, CSci-446 (Computer Graphics I), is like many computer science graphics courses. CSci-446 surveys computer graphics and includes a coverage of display technology, light and color, user interfaces, 2D raster scan graphics (line drawing, polygons, and line clipping), geometrical/affine transformations, 3D graphics (rendering, shading, texture mapping, curves, surfaces, hidden surface removal, and ray tracing), and provides an introduction to image processing.

In addition, the Computer Science Department expects certain courses (including this course) to include team oriented projects. While not as well defined as the approach taken by California Polytechnic State San Luis Obispo [1], the goal is similar – to apply software engineering concepts to a real problem for a real customer. However, given the time and resource constraints of the college curriculum, projects are not always able to incorporate software engineering concepts with the desired rigor nor are the students always able to find real customers. In our case we have settled for a customer who is "pseudo-real" - an instructor who expects the final product to be delivered on time, yet whose interest and expertise is not software engineering (like a real customer). We cannot forget what the goals and expected outcomes of each course are either, in the case of CSci-446, the primary goal is for students to become proficient in computer graphics and to demonstrate that proficiency via class projects developed using OpenGL [2] and either Glut [3] or the Simple Direct Media Layer (SDL) [4]; incorporating software engineering concepts with rigor is a secondary goal.

2 Background

Originally, projects were computer games developed by individuals or small teams (two or three students). Since CSci-446 is an elective having average enrollments of 20-25 students, an average of seven games were developed each semester. Having manageable enrollments, CSci-446 allowed the instructor to have frequent and cordial contact with the individual teams. The resulting informal discussions with students revealed several concerns associated with game development. The most frequently cited were:

1. The design of a game requires a significant amount of creativity and students would frequently struggle to devise an idea for a game that could be completed within the

allotted time. Most of the students deemed their own games as being simplistic. Many students expressed disappointment with their own lack of creativity.

2. The logic/code managing the game play (score keeping, collision avoidance, user interface control, etc) of many of the games proved difficult to engineer, hence they did not "play well" or the students spent all of their time on the "game engine" and not on the graphics. Thus, the 3D graphical component of their games frequently was not well developed.

3. Given the limited understanding/exposure of software engineering, software development, and software development management, many teams struggled to maintain the required schedule and reverted to an ad hoc development process in an effort to adjust for the unexpected problems they encountered. As reported by Hilburn [5], students whose software development experience is limited are unprepared for larger complex projects. They have the technical skills and scientific preparation, but not the necessary people skills and process skills.

While the objectives of the course were being met, a personal goal of the instructor was to make the course fun. After each semester, in addition to the regular course evaluations submitted by all students, students were also queried for ideas on how to make the course more enjoyable. One of the ideas that gained in popularity was to allow the development of a computer animation instead of a game. Thus, in 2004 the option to make computer animations was offered and eight students (four individuals and two teams of two students each) opted to make short (one minute or less) computer animations instead of a game.

Since the development of computer animations could take the course in a slightly different direction (requiring a lecture section on animation concepts), the progress of the eight students was closely monitored; they were asked why they developed the computer animations they did and they were asked to evaluate their own success. Their progress and answers suggested that two of the three game development issues cited previously were much less of a factor in developing computer animations. For example:

1. Lack of creativity was of little concern to the students. They seemed content to simply recreate "some" event. Seven of the eight students seemed genuinely satisfied with their computer animations and said they felt the goal was to recreate the event with as much realism as time and their ability allowed.

2. With no game play logic to contend with, the students concentrated more on the graphics. This was obvious as the 3D graphical component was more sophisticated than that of any of the games developed. Furthermore, all eight students expressed a desire to produce higher quality animations than was possible with OpenGL and all eight students took up the instructor's offer of extra credit to "port" their OpenGL scenes into Persistence of Vision Raytracer (PovRAY) [6] scene description files for raytracing. The scene description files were then rendered with PovRAY and

assembled into a computer animated movie using the MEncoder component of MPlayer [7].

Several of these students even developed algorithms to automatically generate PovRAY scene description files based on their OpenGL code and in 2006, Kris Zarns, a UND computer science graduate student, began the process of piecing the individual algorithms together. He then added additional capabilities resulting in the library/API referred to as RayGL[8, 9]. Additionally, the instructor developed the GRID package GridRAM [10] that allows the PovRAY rendering to be spread across the 48 Linux computers in the department's student accessible labs or on the department's 20 CPU high performance computing cluster.

3 Assignments

Due to the popularity of the animations and the availability of RayGL and GridRAM, all course projects are now animations (games are not allowed) and expected to be of longer play length. While developing our undergraduate specializations we contacted Microsoft's Game Studios and Electronic Arts for recommendations on course requirements for our Game Development and Computer Animation specialization. Their recommendations were to require teams of at least three students, so we made that a requirement also. In an effort to reinforce software engineering principles and better manage the development process, each team is first required to develop a concept for a three to four minute computer animation that includes a specific 3D object (character, object, or scene) that each team member would be responsible for, describe the concept in a written proposal, and have it approved by the instructor. We loosely define this document as the "requirement". Once their concept is approved, each team is then required to develop a storyboard detailing individual scenes and the person(s) responsible, produce a written version of the storyboard, and have it approved by the instructor. This document is loosely defined as their "specification". The weekly assignments that follow (as detailed below) comprise the individual component developments that progress with the course (wire frame models, transformations, motion, solid models, color, lighting models, and texture mapping), culminating with the integration of each team member's 3D object, integration testing, raytracing, and encoding. To encourage students to investigate the more advanced capabilities of OpenGL and PovRAY, extra credit is made available to any student who wants to develop or incorporate additional features into RayGL. In an effort to foster creativity, extra credit is awarded to the team(s) developing the most creative animation.

3.1 Weekly Assignments

HW-1 10 points Due: Tuesday, Jan. 11
Write a 1 page proposal describing your team's computer animation. Please note that you must form teams of 4 (or more), your movie must be about 4 minutes in length, and you must have at least 1 unique 3D animated object for each team member to develop as homework for the semester. I will review the proposals and use them to define the movies that we will tackle this semester. Please note that your proposal must be type-written (no

hand drawn figures or text allowed). Finally, assume that all movies will be 640x480 in size.

HW-2 10 points Due: Tuesday, Jan. 18
Write a program to draw a 640x480 window on the screen. The window must be filled with white. The program must also draw a 2D wire frame model of the first initial of your first name. You can use any color except white (duh). I suggest that you use the makefile from my website, but change the text "mesa:" to the name of your program (hw2). I also suggest that you use the simple example from my website as a starting point. All you should need is a "myInit" routine (see below) and the code required to draw your initial. Demo your program during office hours prior to noon on the due date.

Sample myInit routine:

```
void myInit(void) {
  glClearColor(?, ?, ?, ?);
  glPointSize(1);
  gluOrtho2D(0.0, 640.0, 0.0, 480.0);
  glLoadIdentity();
  glClear(GL_COLOR_BUFFER_BIT);
}
```

HW-3 10 points Due: Tuesday, Jan. 25
Each team must have its movie proposal in its final form by this date. The proposal must have a 1 page (single spaced) description of the movie, including a timeline of events. The proposal must also include a 1/2 page description of each team member's component, including a description of what it does and who the responsible person is. Each team must develop a story board for their movie. The story board must have a picture for each key frame in the movie (I expect dozens of key frames will be required). The story board can be hand written (but it must be legible or I will not accept it). Note that only 1 proposal per team is required, but each team member must be clearly identified.

HW-4 20 points Due: Tuesday, Feb. 1
Write a program to draw a 640x480 window on the screen. The window must be filled with white. The program must also draw a 3D wire frame model of your component. Be very careful to follow the CCW rule! Also, no transformations are required. Finally, I suggest that you use triangles/polygons. Do not use lines as lines cannot be filled in later. Demo your program during office hours prior to noon on the due date.

HW-5 20 points Due: Tuesday, Feb. 15
Modify your program (HW-4) to include animation of your object. I also suggest you consult with your team to insure that you all have the same main and myInit routines. Please note that your full name must appear as a comment at the beginning of your program. Demo your program during office hours prior to noon on the due date.

HW-6 30 points Due: Tuesday, March 1
Modify your program to perform the final animation of your object (must conform to the storyboard). Your object must be a solid model and should include hidden surface removal. Demo your program during office hours prior to noon on the due date.

HW-7 20 points Due: Tuesday, March 15
Modify your program to include a lighting model and shading. You will want to meet with your team to decide where/what type of lighting should be included. However, ambient and diffuse lights are required in all assignments. Demo your program during office hours prior to noon on the due date.

HW-8 20 points Due: Tuesday, March 29
Modify your program to include texture mapping. Also, each student WILL place the following comment at the very beginning of their code:

```
/************************************************/
/* Name: your name                              */
/*                                              */
/* description of this particular code          */
/*                                              */
/* CSci 446 / spring 2011                       */
/************************************************/
```

Please note that this is the last individual assignment. All future assignments will be team efforts. Demo your program during office hours prior to noon on the due date.

HW-9 40 points Due: Tuesday, April 12
Each team will combine their programs into a SINGLE executable that approximates the story board. The source will have at the top:

```
/************************************************/
/* Name: member #1 name                         */
/* Name: member #2 name                         */
/* Name: member #3 name                         */
/* Name: member #4 name                         */
/*                                              */
/* HW9                                          */
/*                                              */
/* CSci 446 / spring 2011                       */
/************************************************/
```

Demo your program during office hours prior to noon on the due date.

HW-10 60 points Due: Tuesday, May 3
Each team must submit (via email) their entire combined RayGL source code (as 1 program requiring NO user input). Each team must submit (via email) their source code,

your executable, and your 4 minute movie that is playable using MPlayer on Linux (in room 109). The source will have at the top:

```
/*************************************************/
/* Name: member #1 name                          */
/* Name: member #2 name                          */
/* Name: member #3 name                          */
/* Name: member #4 name                          */
/*                                               */
/* HW10                                          */
/*                                               */
/* CSci 446 / spring 2011                        */
/*************************************************/
```

Each student WILL place the following comment at the very beginning of their code:

```
/*************************************************/
/* Name: your name                               */
/*                                               */
/* description of this particular code           */
/*                                               */
/* CSci 446 / spring 2011                        */
/*************************************************/
```

The movie will have 1 second of aFrame.png (Figure 1) modified to display your movie's title. You can resize the frame (cut and paste the text into the new frame - do NOT simply rescale it), but you must leave the background black.

Figure 1: Title frame (aFrame.png).

The movie will have 1 second of aFramb.png (Figure 2) unmodified! You can resize the frame (cut and paste the text into the new frame - do NOT simply rescale it), but you must leave it as is otherwise.

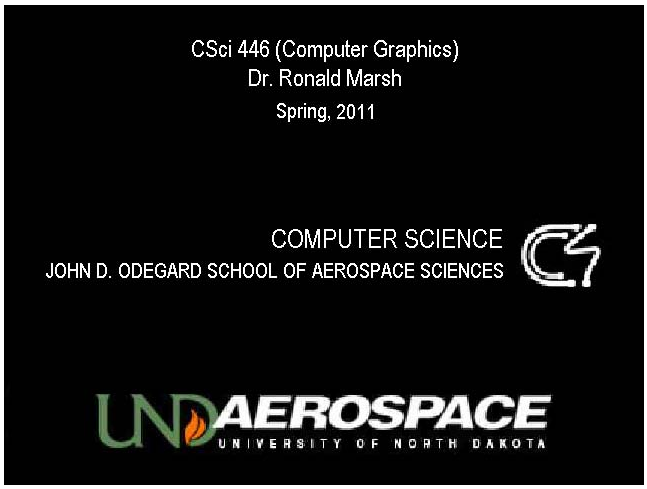

Figure 2: UND Computer Science frame (aFramb.png).

The movie will have 1 second of yFrame.png (Figure 3) modified to display your movie's credits (who did what). Include your team members full names (First name, Middle initial, Last name). You can resize the frame (cut and paste the text into the new frame - do NOT simply rescale it), but you must leave the background black. Note that the this frame (the credits frame) must include the text "Rendered on the CrayOwulf in the Computer Science Department using POVRAY (www.povray.org)."

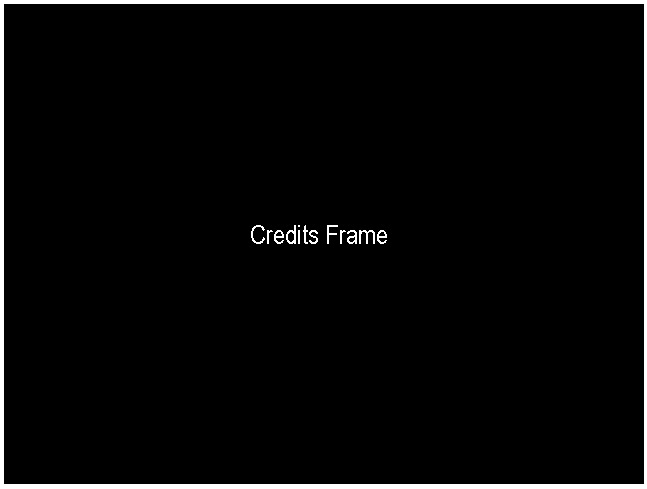

Figure 3: Credits frame (yFrame.png).

The movie will have 1 second of zFramb.png (Figure 4) unmodified! You can resize the frame (cut and paste the text into the new frame - do NOT simply rescale it), but you must leave it as is otherwise.

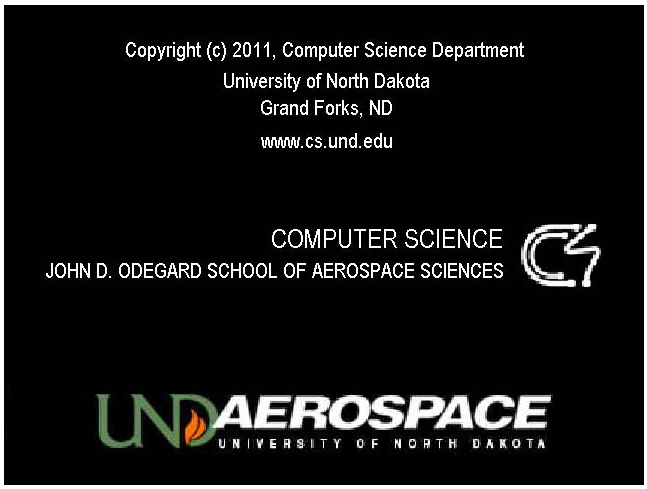

Figure 4: Copyrights frame (zFramb.png).

I will base the grade strictly on the movie (AVI file). Therefore, each team member should be a part of the movie making process to ensure that your component is included. There are no second chances. I will grade these once and only once!

4 Conclusion

As RayGL and GridRAM have evolved, the animations developed have also become more sophisticated (screen shots shown in figure 5). Students have frequently commented on how the development process followed made their Software Engineering course "make sense to them now" and many students have commented on how important is was to have a good specification before starting development.

However, interpersonal/team dynamics, specification creep, and the grading policy are still issues that must be continually addressed. Interpersonal/team dynamics was aggravated by the random assignment of students to teams; in a few cases, team members simply did not get along. To reduce the interpersonal/team dynamics issues, we now allow students to form their own teams. While an unrealistic model of the post-graduate professional setting, we are forced to realize that we are not working with experienced professional developers, but students who are still developing their skill sets and who can become overwhelmed by the scope of the project.

The problem of specification creep stems from the fact that students do not realize the amount of work that it takes to create by hand (animation tools such as Maya [11] are not allowed) the many components/actors required in a computer animation. Several students became infatuated with their creation and obsessively put in whatever time it took to create their "grand vision." However, this would frequently create discord within the team when their other, not so infatuated team members, would complain about the ever changing specification (storyboard) and the inevitable increased interactions among the component parts required to achieve their "grand vision." We address specification creep by now penalizing teams that vary from the specification (storyboard). We have also considered the adoption of a "change control mechanism" requiring additional paperwork

to request approval for any/all changes to the original storyboard. This will allow for changes (and extra credit), but in a manner consistent with industry.

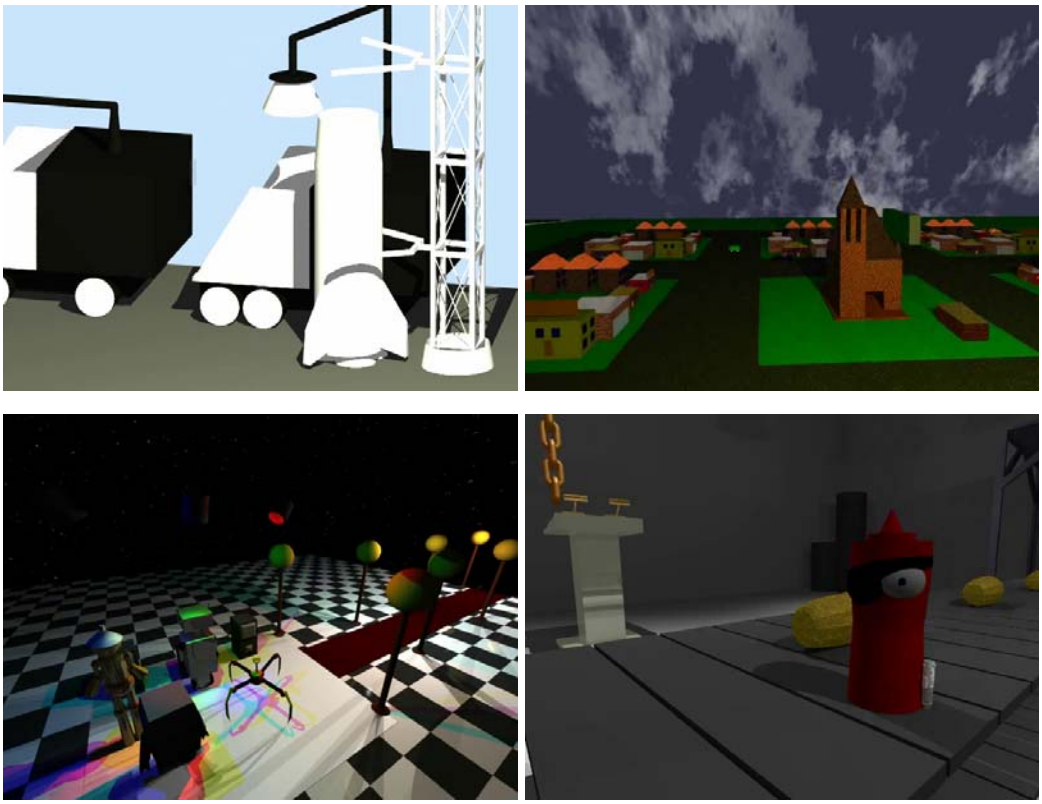

Figure 5: Clockwise from top-left “2.56 ... Crash,” “OS_Wars(),” “Dancing robots,” and “Persistence Of Defiance.”

While it is very unlikely that any student would plagiarize another’s work, it can be a challenge for the instructor to fairly assign points given the wide range of animations that can be developed. This is addressed by requiring instructor/team face-to-face meetings as part of the weekly homework reviews (our loosely defined “design reviews”) where each student demonstrates their component. These meetings also facilitate student/instructor interactions as to where improvements can be made or how to solve problems that inevitably arise. These meetings also allow the instructor to determine how well each component conforms to the original storyboard and what extra credit work was done and by whom. It is also important for the instructor to stress that any overall “lack of creativity” would not diminish their grades as long as their animation conformed to the original storyboard.

We found that animation projects eliminated many of the issues (creativity and coding) associated with developing games and made it easier for students to be creative and take ownership of their project. We found that following the film industries approach to developing a movie reinforced software engineering concepts.

Finally, a review of the course convinced the Computer Science Department to develop a degree certificate in computer animation and game development. In accordance with

UND requirements, the “specialization” requires that certain computer science electives and general education electives to be taken in order for the specialization to be noted on their transcript. In addition to the core computer science program, the specialization requires Art 110 (Introduction to the Visual Arts), Art 112 (Basic Design)**,** Linear Algebra, Physics, Human Computer Interface Design, Computer Graphics I, and Computer Graphics II. Computer Graphics II concentrates on game development using Microsoft’s XNA [11] development kit. While it may seem that we are reversing ourselves by going back to game design in Computer Graphics II, our belief is that once students complete Computer Graphics I they will have enough experience with a graphics library, enough experience on a team oriented project, and enough experience in developing a larger (full semester) project to be able to manage the added complexities of game development.

References

[1] E.E. Villarreal and D. Butler, “Giving Computer Science Students a Real-World Experience,” in *Proceedings of SIGCSE* 1998, 1998.

[2] OpenGL, http://www.opengl.org/. Last accessed March 1, 2011.

[3] Glut, http://www.opengl.org/resources/libraries/glut/. Last accessed March 1, 2011.

[4] Simple DirectMedia Layer, http://www.libsdl.org/. Last accessed March 1, 2011.

[5] T.B. Hilburn, “Software Engineering Education: A Modest Proposal,” *IEEE Software,* 1997.

[6] The Persistence of Vision Raytracer (PovRAY), http://www.povray.org/. Last accessed March 1, 2011.

[7] MPlayer, http://www.mplayerhq.hu/design7/news.html. Last accessed March 1, 2011.

[8] K. Zarns and R. Marsh, “RAYGL: An OpenGL to POVRAY API,” in *MICS 2006: Proceedings of the Midwest Instruction and Computing Symposium*, Mt. Pleasant, IA, 2006.

[9] R. Marsh and K. Zarns, “RAYGL: An OpenGL to POV-Ray API,” in *CATE 08 - Computers and Advanced Technology in Education*, Crete Greece, 2008.

[10] R. Marsh, “GridRAM: A Software Suite Providing User Level GRID Functionality for University Computer Labs,” in *MICS 2007: Proceedings of the Midwest Instruction and Computing Symposium*, Grand Forks, ND, 2007.

[11] Autodesk Maya, http://usa.autodesk.com/maya/. Last accessed March 1, 2011.

[12] Microsoft XNA, http://msdn.microsoft.com/en-us/library/bb200104.aspx. Last accessed March 1, 2011.