1. GOALS AND OBJECTIVES

Many Computer Science (CS) and engineering students seek an education with a direct applicability. This is especially true in a high-profile field like Computer Graphics (CG), where students are familiar with many popular applications (e.g. graphical editors, games, and special effects). Many students are motivated and enthusiastic about the CG field because they want to understand how to build such fascinating applications.

As educators, we would like to concentrate on fundamental principles and competencies where there is potential applicability throughout students' careers. Many educators align these needs by relating fundamental principles to real-world applications. It has been demonstrated that in many CS domains that this pedagogical approach is effective and welcomed by students. For example, relating knowledge in introductory programming courses to real-world case studies [1,2]; or relating programming projects to real-world experiences in Software Engineering and Databases [3]; or relating algorithms to internet applications [4].

In alignment with these observations, we have developed CG courses based on analyzing the design and implementation requirements of familiar interactive CG applications [5,6]. In these courses, functional modules of moderately complex CG applications are studied, and students synthesize the concepts learned by building their own applications based on these modules. The learning outcomes of these courses are for students to understand the essential CG concepts, to gain practical hands-on implementation experience, and to be able to relate concepts learned to real-world applications.

Over the past few years, the enrollments in these CG courses have increased steadily while student populations in our program remained somewhat constant. In addition, an increasing number of students are pursuing careers in the CG field after these courses. Our approach has been well received by the CG education community [7,8], where recently we have been invited and published our extended findings in a full-length journal article [9]. Currently, we are working with other educators in organizing public forums to compare and contrast our different approaches to teaching CG [10].

This ***DUE CCLI-EMD Proof-of-Concept*** project proposes activities to assess, refine, and facilitate the dissemination of our course materials, and to assist other educators facing similar challenges adapting this new approach. More specifically, we seek funding to support: (1) refining our existing course materials such that they can be suitable for public access; (2) independently assessing the effectiveness of the refined course materials by other educators; (3) updating the course materials based on these assessments; and (4) disseminating the top-down approach and associated course materials.

2. BACKGROUND AND MOTIVATION

When we taught the introductory CG course following the traditional textbooks (e.g. [11-15]), our students were disengaged. According to our student population, we have found the cause of the problems to be: **(1)** unclear relevancy of knowledge; **(2)** unrealistic hands-on exercises.

2.1. Relevancy of Knowledge

''while I appreciate that the DDA line drawing algorithm is part of the foundation building blocks of 3D applications like Maya, I also notice the complexity of the software system, and wonder if time may not be better spent learning other more interesting aspect of the graphics system'' – CSS450 Student, Fall 1999.

Among CG educators there is a collective understanding of the essential fundamental principles and concepts that define the field [17,18]. Constrained by the time limit in academic terms, traditional CG courses cover subsets of these essential principles and concepts based on respective student learning outcomes (e.g. [19,20]). In the course of a term, these courses present to students most of the foundation building blocks of modern CG systems and students gain knowledge of what is "under the hood" of modern graphics coprocessors. These approaches ([18,21-23]) are characterized as bottom-up because they present the CG field focusing on the individual low-level foundation building blocks [7]. While the strength of the bottom-up approach is that it teaches the basic mathematics and methodology of graphics engine design, this approach seldom addresses application-level issues and does not enable students to use a powerful graphics API to design a complex application. In the near term, the bottom-up approach may seem to lack practical relevancy.

2.2. Practicality of Hands-on Exercise

''while it was interesting to practice multi-way symmetry in mid-point ellipse algorithm, I still do not understand how these exercises relate to the graphics programs I use'' – CSS450 Student, Fall 1999.

This sentiment is shared by many educators. Kubitz suggested that the traditional (bottom-up) approach to teaching CG is ''*mostly wrong*'', that CG courses should study higher level issues based on latest APIs [23]. To provide students with a holistic understanding of the CG field, [24] traded depth for breadth of coverage and described strategies to cover Rendering, Modeling, Animation, and Postprocessing in one introductory CG course. Others [25-29] point out that API-based CG courses that cover high-level issues reach a wider audience and thus are more suitable for students who only take one CG course in their undergraduate education.

2.3. API-Based Approaches to Teaching CG

The existing API-based approaches to teaching CG use case studies to demonstrate individual CG concepts. The two widely accepted API-based CG textbooks [25,26] present these knowledge from opposite perspective. Hills [25] extends the traditional bottom-up approach by relating the low-level algorithms to higher-level CG concepts found in a popular 3D API. While Angel [26] takes the alternate approach where he first presents individual high-level CG concepts in simple case studies, and then proceeds to describe the underlying low-level algorithms required to support these concepts.

When compared with the traditional approaches to teaching CG, API-based approaches traded covering the depth of fundamental principles for the concept-level issues and the associated case studies. The API-based approaches are pragmatic bottom-up because they concentrate on studying individual issues (*bottom*) of general CG applications (*up*) based on modern APIs. Real-world CG applications require the collaboration of many concepts. For example, a *practical* interactive CG application must integrate CG concepts with independent user and timer events based on the best practices in software design and architecture. API-based pragmatic bottom-up approaches to teaching CG do not attempt to demonstrate the complex interaction of the CG concepts in practical settings.

3. TOP-DOWN APPROACH TO TEACHING CG

Our top-down approach to teaching CG [7,9] turns the pragmatic bottom-up approach around by identifying a category of applications and decomposing the applications into *functional modules*. The course would then cover the modules while relating each module back to the target applications. In this way, students learn the foundations and structure of graphics applications, practice implementing the more visible application-level skills, and more importantly, study and experiment with issues involved in integrating multiple CG concepts in complex applications.

Ideally the *functional modules* from the top-down approach should be continuously decomposed into smaller units until the units become the *essential concepts* identified in the bottom-up approach. However, given the sophistication of the modern graphics applications, this ideal decomposition process is non-trivial and typically cannot be accomplished in a 16 week semester (or 10 week quarter). This is the same reason why a typical bottom-up CG course does not have time to complete a moderately complex graphics application starting from the foundations. The popular graphics API libraries can serve as a convenient convergence point for the two approaches. A top-down approach would teach students to implement functional modules based on the popular graphics APIs. Besides serving as a practical skills training, using an API extensively in building a moderately complex system helps students understand the design and appreciate the pros and cons of the API.

Top-down is complementary to the traditional bottom-up approaches [30,31] because it trades high-level system architecture understanding (e.g. scene graph design) for low-level foundational knowledge (e.g. rasterization algorithms). Of course, the two approaches are not strictly mutually exclusive. For example, the bottom-up approach often uses a simple target application framework (case studies) for students to investigate the implementation of different algorithms, whereas, in the top-down approach, it is possible to cover some basic low-level algorithms.

One of the difficulties in designing a top-down syllabus for introductory CG courses is in identifying an appropriate *top*. The *top* in this context refers to a *target software system*. As mentioned, in the bottom-up approach CG educators have a collective understanding of what are the essential philosophy and concepts that defines the field [18]. These concepts are the basic building blocks of *general CG systems*, whether it is a hardware CG system, a batch software CG system, or an interactive CG system, etc. The key is that the basic building blocks are suitable for building any of these CG systems. In a top-down approach, where a *system* is decomposed for identifying the

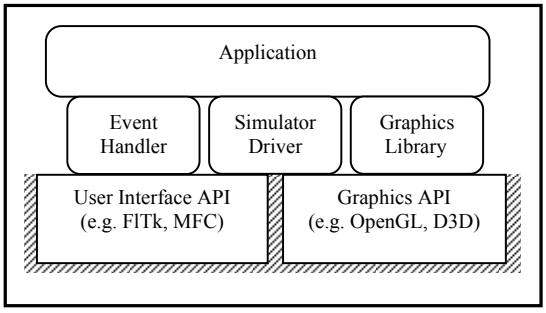

Figure 1: Components of Interactive Applications

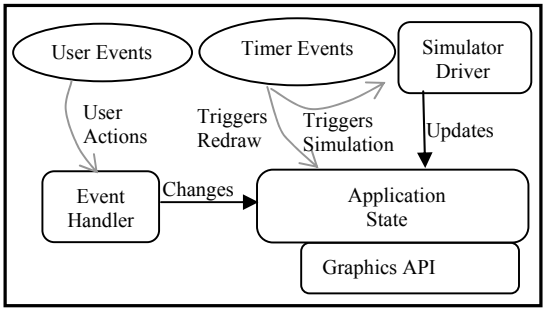

Figure 2: Architectural Framework

supporting requirements, the ''*target software system*'' must be well defined. It is important to identify a target system that demands a sufficiently large set of common supporting requirements shared by many CG applications.

As cautioned [27], courses based on popular applications (e.g. Maya) and/or APIs (e.g. DirectX, OpenGL) run the risk of teaching students to be *users* rather than *practitioners* of CG. As an example, the top-down approach should identify major functional modules in Maya (e.g. user interaction module, or hierarchical modeling module) and examine how to design and implement these modules based on functionality supported by OpenGL or DirectX. This is different from *learning how to use Maya* [32], or *learning how to use specific APIs* (e.g. [33-36]). As newer versions of the software and/or APIs are released, the knowledge students gained must continued to be valid and applicable.

4. PRIOR WORK AND PRELIMINARY RESULTS

We have developed and refined a sequence of two 10-week-Quarter CG programming courses: a 2D introductory course [5], and a more advanced 3D Programming course [6]. The objectives of both courses are for students to gain knowledge and experience in designing and implementing "*popular interactive graphics applications*". Being a small department in a preliminary undergraduate institution, our students have very limited opportunities in taking elective courses in CG. Our choice of "*popular interactive graphics applications*" as the *top* in the top-down approach is based on the fact that most students are familiar with such software systems. This choice assists us in relating students' personal experiences to CG concepts in the limited time.

4.1. CSS450: 2D Introductory CG Course

We observed that most of the "*popular interactive graphics applications*" (e.g. Microsoft PowerPoint) can be described as applications that allow users to interactively update their internal states. These applications provide real-time visualization of their internal states with the graphics subsystem. In addition, these applications typically support some mechanisms that allow the user to define simple animations. Figure 1 shows one way of decomposing these types of applications into major functional modules. Figure 2 depicts an implementation architecture based on the identified functional modules. The syllabus of our introductory CG course is a mapping of the requirements to understand and implement these functional modules into specific topics in CG.

Three general topic areas are identified: **(1)** Event and Simulator Driven Programming; **(2)** Graphics API Abstraction; and **(3)** Transformation. Topic 1 allows students to learn the external-control programming model while practicing new Graphics User Interface tools. Topic 2 introduces students to the idea of interaction with behavior of abstract objects independent from Graphics API, and prepares students for large-scale software development. Topic 3 covers coordinate transformation pipeline and leads naturally to hierarchical modeling and scene graph design. These topics are covered in the first 6 weeks' of class. At this point students commence to work on their final project. The remaining 4 weeks are divided between discussions of topics related to students' final project development (e.g. collision detection algorithms, texture mapping, etc.) and the fundamental algorithms in CG (e.g. color models, raster level scan conversions, etc).

4.2. CSS451: 3D CG Programming Course

By this point, from the 2D course, students understand and have synthesized event/simulator-driven interactive applications that contained hierarchical models defined in appropriate coordinate systems, and supported displaying with multiple viewing windows. The 3D course is designed around topics in Viewing, Rendering, and Modeling that allow students to continuously and seamlessly bring their applications into 3D world.

We begin by studying 3D viewing and camera manipulations. After that students are able to upgrade and examine their multiple-view 2D systems with multiple cameras in 3D space. We then cover topics in rendering (illumination, shading, and texture mapping). This allows students to properly illuminate their hierarchical models. Lastly we cover polygonal and mesh modeling, which allow students to work with more interesting models in their applications. The simulator-driven component of their applications ensures students are aware of and constantly working with simple topics in animations (e.g. motion from continuously updated matrices, simple elastic-collisions, etc.). Similar to that of the introductory course, these topics are covered in the first half of the 10-week-quarter so that students can commence with their final projects. Once again, the remaining of the quarter is divided between discussions of topics related to students' final project development (e.g. search structures, levels of details, view culling, etc.) and the more advanced topics/effects in CG (e.g. transparencies, reflections, shadows, etc).

4.3. Results

We have experienced enthusiastic student comments and observed effectiveness in student learning in these courses. In the past four years the overall student population in our department has remained somewhat constant and yet the enrollment of the 2D CG course has increased over 100%: from 10 in 2000, to consistently more than 25[1]. Students' further interests in the field can be observed from the enrollments of the follow-up 3D course: from 1 in 2001, to consistently around 20. Of these students, about 20 are currently working or interning at local graphics/games companies. Although these numbers are based on relatively small sample size, they do reflect success and show encouraging and consistent trends. Our institution is impressed with our results, and is supportive of our efforts in refining the course materials. Recently we were awarded a seed funding [37] to complement this NSF proposal. This modest funding allows us to upgrade some of our laboratory development environments and begin engaging some student hourly as will be required in this project.

We have received positive feedback from the general CG education community. The initial publication of this work [7] was well received, where we were invited and published our extended findings in the Computer & Graphics Journal [9]. Recently we condensed our top-down philosophy into a book summary, and organized the outline of our lecture notes into an extended table of content. We sent these materials to colleagues in the fields and to a commercial textbook publisher (AK-Peters). The book summary, extended table of content, and endorsement/support letters are included in the supplementary section of this proposal. Currently we are engaging in and collaborating with other CG educators in healthy discussions of comparing and contrasting the different approaches to teaching CG [10].

Our courses materials should be formally assessed and refined to effectively support the defined student learning outcomes. So far, the majority of the materials were prototyped during the courses; and these materials were re-organized/refined based solely on students' course evaluations and the PI's self-reflection.

A textbook should be developed for our top-down approach. Traditional textbooks in the field (e.g. [11-15,38-41]) are designed for bottom-up approaches with focus on individual low-level algorithms. Recent API-based textbooks (e.g. [25,26,42]) exemplify these low-level algorithms with one popular graphics API (OpenGL). All these textbooks cover the details of foundational algorithms in isolation and seldom attempt to relate the concepts to modern CG applications. On the opposite end of the spectrum, existing application-level text/trade books (e.g. [43-46]) concentrate exclusively on discussing one software system and on "*how to become a proficient user*" of a system. A textbook for our top-down approach would cover CG knowledge on how to build a general *interactive-graphics-application* based on an *abstract-3D-API*.

5. PROJECT OBJECTIVES AND PLAN

Our project is designed to accomplish two major objectives: first, formally evaluate, develop, and refine our existing course materials to ensure effective support of the top-down approach; second, disseminate the refined course materials, share our results with other CG educators, and begin facilitating those that face similar challenges adopting a similar approach. Our ultimate goal is to develop a textbook based on the results of this project supporting straightforward adaptation of the top-down approach to teaching and learning CG.

[1] At the time of writing this proposal (June 2004), the 2D Computer Graphics course for Fall 2004 is fully enrolled (40), with additional students on the waiting list.

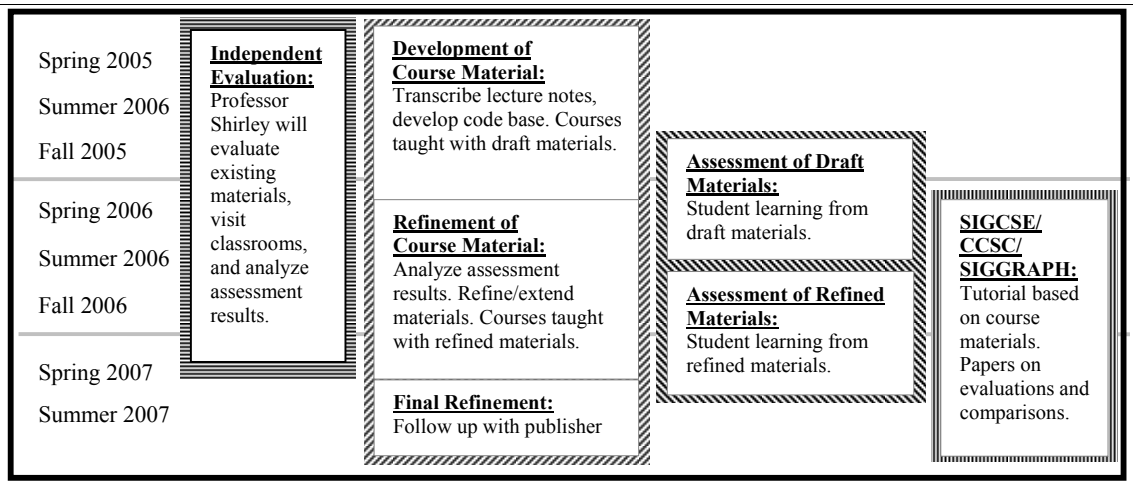

Figure 3: Project Schedule

Our procedures to achieve these objectives are informed by the most recent and comprehensive assessment scholarship [47-50]. As depicted in Figure 3, we have a four prong implementation plan: (1) **independent evaluation of course materials**: evaluate the organization and technical materials by an independent expert CG educator; (2) **development of course materials**: develop, refine and reorganize the course materials continuously based on evaluation and assessment feedbacks; (3) **assessments of student learning**: monitor the classroom teaching and student learning outcomes by expert scholars in student learning assessments; (4) **dissemination** of our results and course materials.

5.1. Independent Evaluation of Course Materials

The objectives are to independently evaluate the student learning outcomes, and effectiveness of the course materials. These objectives will be achieved via three phase assessments: pre-analysis, continuous monitoring, and post-analysis by an experienced educator in CG from outside of our institution.

Pre-analysis will be performed before the courses are offered where the experienced CG educator will critically examine existing course design and materials: including articulation of course goals, syllabi, past examinations/assignments/final projects, and the existing on-line concept-demonstrating-applications. These materials will be evaluated against students' demonstrated ability to design and implement moderately complex interactive CG applications. An evaluation report including recommendations will be provided to the PI for appropriate refinements to the materials. During the offering of the courses, the refined course materials will be continuously monitored and feedback provided for the PI to make further adjustments. As part of the monitoring effort, the experienced CG educator will attend some classes to obtain first-hand feedback from students. After the courses, post-analysis will be performed based on the refined course materials, and a final report with recommendations will be provided to the PI.

Peter Shirley, a campus-wide teaching award winning [51,52] CG Professor from University of Utah has agreed to serve as our independent evaluator. Professor Shirley is the author of two books [15,53] in the CG field. In his most recent bottom-up approach CG textbook [15], Professor Shirley pointed out that both top-down and bottom-up approaches have their merits. Professor Shirley's distinguished teaching experience, pedagogical position, and book writing experience uniquely qualify him to be our independent evaluator.

5.2. Development/Refinement of Course Materials

Over the past four years, our implementation of the top-down approach has been developed into: (1) lecturing the CG concepts; (2) demonstrating the concepts with applications; (3) analyzing the source codes involved in implementing the concepts; (4) having students synthesize the concepts into their own applications.

These 4 steps are followed for each of the CG concepts covered in both of the courses. The lecture notes for these courses are unique because CG concepts are presented based on two 3D APIs (Direct-3D and OpenGL). In addition, the lecture notes include extensive references to the source code of the accompanied library modules highlighting issues involved in integration with other CG concepts and techniques in implementation. Most of the existing lecture notes for the courses are hand-written and organized in an ad hoc manner. These lecture notes are

hand-edited, updated, amended, each time the courses were taught. After four years of updating the lecture notes contents have somewhat stabilized. However, they have also become relatively difficult to parse. These lecture notes should be transcribed into electronic format for public access.

We choose to custom develop the concept-demonstration-applications (CDAs) of step 2 and 3, even though there are other existing systems. There are three reasons for our choice. First, as *practitioners of CG*, it is important for students to understand/experience with implementation of CG concepts, and not be *mere users of* CG concepts. Second, we want to ensure students' familiarity with the source code base. In the beginning of the course the CDAs are relatively simple and the source codes are straightforward to understand. The complexity of the CDAs increase gradually as each new concept is introduced. In this way, students can follow the development of the source codes closely and concentrate on analyzing the implementation of each new concept. Third, we want to demonstrate CG concepts with same CDAs based on different graphics APIs. For example, we have implemented 2 versions of the same application demonstrating "*coordinate transformation pipeline*" using "*matrix stacks*" where the first version was based on OpenGL and the second one was based on Direct3D. We find this to be a valuable tool in emphasizing the independence of CG concepts from the underlying supporting APIs.

Maintaining the CDAs is becoming very difficult and time consuming. We have designed an API independent Object Oriented Graphics Library (OOGL) to support the development of the CDAs. However, in practice, more than one version of OOGL is needed because the sophistication of the CDAs varies greatly throughout different stages in the two courses. In addition, many existing CDAs were rapidly prototyped during the courses on per-need basis. For these reasons, individual CDAs are typically developed based on slightly modified OOGL interfaces. Currently the existing 100+ CDAs have similar and yet completely independent source code bases with an average size of about 2000 lines of C++ code per CDA.

The effective reorganization and consolidation of these CDAs is very important to the continual success of our approach. The exact work involved will depend on the recommendations from Professor Shirley after the pre-analysis. We anticipate this work to include: (1) Designing a development framework to support the consolidation of similar CDAs. For example, controls should be incorporated into a single application to support dynamic-run-time loading of different APIs to demonstrate the same CG concept. (2) Refining the OOGL definition into different versions with appropriate degrees of sophistication. For example: a. flat hierarchy for simple primitives; b. hierarchy with inheritance for interaction with abstract interfaces; c. transformation support for hierarchical modeling and coordinate pipeline; d. camera support for 3D viewing; and e. additional classes for lighting, materials, textures, etc. (3) Consolidating and porting CDAs based on the new development framework and appropriate versions of OOGLs. Currently, about 30-40 of the 100+ CDAs uniquely demonstrate CG concepts, the rest are duplicated source code supporting interface with different APIs (e.g. OpenGL vs Direct3D, FlTk vs MFC, etc.).

The PI of this project, Professor Kelvin Sung, designed and developed the current OOGL and CDAs. He will work with hourly programmers to accomplish the above tasks. While as a graduate student, Professor Sung worked extensively with Computer Graphics Standards and APIs including: large scale design and development [54-56], and scholarly evaluations [57]. Although the works are somewhat dated, the underlying principles of API design remained the same and the development experiences apply directly to this project. Prior to joining CSS, Professor Sung taught CG to traditional Computer Science students at the National University of Singapore [58]. Afterwards, he became one of the chief designers of the Rendering module of the academy award winning [59] Maya software system [16]. Professor Sung's familiarity with teaching both traditional and non-traditional students coupled with his hands-on experiences from designing/developing large-scale API and commercial CG systems uniquely qualify him for designing teaching materials based on practical applications, which is the top-down approach.

Because of the amount of materials involved, it is expected that the first year's effort will be focus on higher-level organization and prototyping. We expect the lecture notes to be largely transcribed (probably lacking details), and the OOGL/CDAs prototyped. Based on the results from the assessment program (please refer to Section 5.3), in the second year we expect some re-organizations while majority of the efforts will be spent filling in the details and extending the lecture notes and documenting the OOGL/CDAs.

5.3. Assessments of Student Learning

We are adopting a mixed method approach to assess student learning, as recommended by the NSF's Directorate for Education and Human Resources [60]. Beginning with a clear statement of student learning outcomes, multiple assessment instruments will document student learning (through pre- and post-test and evaluation of staged assignments), student perceptions (through logs students will complete along with the staged assignments), peer

review of the course (as discussed in Section 5.1), and faculty self-assessment (based on reflections on in-class activities, student work and student feedback). The data collected will be analyzed on an ongoing basis, so that the results can be fed back into continuous revision of the course.

Pre- and Post-Assessment: An assessment will be administered to all students at the beginning of the course. This assessment will cover all the computing concepts identified as desirable prerequisites for the course as well as outcomes of the course. The same instrument will be administered at the end of the quarter in order to determine change and to correlate student performance in the course overall with command of CG concepts.

Student logs: Students will complete logs associated with assignments every other week of the course. The logs encourage students to reflect on their understanding of concepts presented in the course, to evaluate their ability to apply those concepts to assignment problems, and to identify areas in which they still have questions.

The responses to the pre- and post-assessment instruments and the student logs will not be shared with the course instructor during the quarter, but the assessment coordinator will be able to associate responses with particular students in order to correlate them with performance and grades on course assignments.

Homework assignments: For the past few offerings of the courses, one assignment was created/revised for each topic covered. These assignments were in the form of technical specifications where students must demonstrate concepts learned by designing and implementing interactive CG applications. For the final project, the technical specifications required students to demonstrate the ability to independently synthesize all concepts learned in an interesting interactive CG application. The final project involved formal written and oral proposals, in-class progress demonstrations, and at the end, formal written and oral report with user manual and final in-class demonstration. We will modify the assignment structure based on pre-analysis feedbacks.

Faculty self-assessment: A journal has been maintained that records the observations of the progress of the course with particular attention to insights from grading assignments and exams, reflections on in-class activities and questions, and ideas for further revision of the course.

By using these data in concert, we will be able to: (1) determine how fully the learning outcomes for students are being achieved (through assessment of the homework they have completed); (2) identify concepts that are challenging and are not adequately addressed in course materials (through student logs); (3) track the course of student learning across the quarter (through homework and logs); (4) correlate learning in the course to prior preparation (through the pre-test); (5) locate stages in the course when changes would be most useful (through all the data sources).

Dr. Rebecca Reed Rosenberg, interim director of the UWB Teaching and Learning Center [61], will design and administer this student assessment program. She will work in consultation with the staff of the University of Washington Seattle's Center for Instructional Development and Research [62], and specifically with Dr. Wayne Jacobson, who specializes in assessment of technical and computing instruction; and with Dr. Cinnamon Hillyard, director of the UWB Quantitative Skills Center [63].

We will complete the draft development of the course materials prior to the next offering of our CG courses. In this way we could administer the assessment program based on the draft course materials. We will begin refining and extending the materials as soon as assessment results becomes available. In the following year, we will teach our CG courses with the refined materials. The follow up assessment will allow us to verify the effectiveness of the refined/extended course materials.

5.4. DISSEMINATION

The top-down idea [7] is well received [8,9]. We plan to continue to take advantage of the different presentation formats (papers, tutorials, forums, etc.) in technical conferences for educators (e.g. SIGGRAPH Educators Program, SIGCSE conferences, and/or CCSC regional conferences) to present the different/appropriate aspects of our work. For example, panel discussions on different approaches to teaching CG [10]; forum discussions on the top-down approach; tutorials/workshops based on our course materials; and scholarly papers based on assessment results (e.g. [64]). The goal of these presentations would be to raise the awareness of the top-down approach. In addition, we will continue to post results of our work at our courses' web-sites [5,6]. Throughout this project, we will keep the commercial publisher (AK-Peters) update-to-date of our progress. At the end of this project, we will follow-up with the publisher and further refined our course materials into a textbook. With the above dissemination efforts, educators facing similar challenges would be aware of our results, and the textbook will provide a straightforward adaptation pathway.