A. SLIDE Laboratory Assignments Text References

"Computer Graphics: Principles and Practices, Second Edition" by Foley, van Dam, Feiner, and Hughes [11] has been used as the text book during the last two offerings of our course. The sequence of the SLIDE laboratory assignments make reference to sections of this book in a rather non-linear sequence. The following list orders these references by the SLIDE assignment which first makes use of them:

1. **Interactive 2D Polygon Builder**
   - Geometry Construction Techniques, Section 8.3.2, p. 382-385
   - Sampling vs. Event-Driven Processing, Section 2.2.3, p. 42-48
2. **2D Polygon Editing and Morphing**
   - Pick Correlation, Section 2.2.6, p. 48-50
   - Dynamic Manipulation, Section 8.3.3, p. 386-388
   - Interpolation, Section 21.1.3, p. 1060-1064
3. **2D Dynamic Hierarchical Scene Renderer**
   - Mathematics for Computer Graphics, Appendix A, p. 1083-1111
   - Standard Rendering Pipeline, Section 18.3, p. 866-873
   - Geometric Transformations and Homogeneous Matrix Representation, Chapter 5, p. 201-226
   - Geometrical and Hierarchical Modeling, Section 7.1, p. 286
   - Extents and Bounding Volumes, Section 15.2.3, p. 660-663
4. **2D Bounding Box Rejection and Polygon Clipping**
   - Visibility Hierarchy, Section 15.2.6, p. 665
   - Elision and LOD, Section 7.13, p. 340-341
   - Cohen-Sutherland Outcodes, Section 3.12.3, p. 113-117
   - Sutherland-Hodgman Polygon Clipping, Section 3.14, p. 124-127
5. **3D Parallel Viewing and Crystal Ball Interface**
   - Projections and View Orientation Matrix, Section 6.5, p. 258-271
   - Plane Equation, Section 11.1.3, p. 476-477
   - Back Face Culling, Section 15.2.4, p. 663-664
   - Transforming Planes, Section 5.6, p. 216-217
   - Crystal Ball, Section 8.2.6, p. 376-387
6. **3D Perspective Viewing**
   - 4D Polygon Clipping, Section 6.5.3, p. 271-278
   - Stereopsis, Section 14.7, p. 616-617
7. **Polygon Scan Conversion and Software Z-Buffer**
   - Coherence, Section 15.2.1, p. 657
   - Polygon Filling, Section 3.6, p. 92-99
   - Filling Rules, Section 19.2.8, p. 964-965
   - Z-Buffer, Section 15.4, p. 668-672
   - Scan-line Algorithms, Section 15.6, p. 680-685

8. **Lighting and Shading**
    - Local Illumination Pipelines, Section 16.14.1, p. 806-809
    - Illumination and Shading, Chapter 16, p. 721-741
    - Diffuse or Lambertian Illumination, Section 16.1.2, p. 723-727
    - Specular or Phong Illumination, Section 16.1.4, p. 728-731
    - Gouraud Shading and Average Vertex Normals for Polyhedrons, Section 16.2.4, p. 736-737
9. **Procedural Modeling and Photo-realistic Rendering**
    - Representing Curves and Surfaces, Chapter 11, p. 471-529
    - Primitive Instancing and Sweeps, Section 12.3, p. 539-541
    - Procedural Models, Sections 20.2-20.4, p. 1018-1030
    - Global Illumination Pipelines, Section 16.14.2, p. 809
    - Visible-Surface Ray Tracing, Section 15.10, p. 701-715
    - Recursive Ray Tracing, Section 16.12, p. 776-793
    - Radiosity Methods, Section 16.12, p. 793-804
    - Aliasing and Antialiasing, Section 14.10, p. 617-646
10. **Final Project: Interactive Visualizition and Simulation**
    - Animation, Chapter 21, p. 1057-1080

B. SLIDE Student Survey

At the end of the Spring 1999 semester of CS184: Foundations of Computer Graphics, 44 students filled out surveys about the SLIDE graphics language, rendering system, and set of programming laboratories. The following shows their responses in a condensed form.

1. The things I liked best about SLIDE and the programming labs are:

Category	Repeats	Student Comment
Fun / Rewarding		
	8	Cool and Fun Assignments
	8	Visual aspect allowed to view results of efforts
Educational		
	9	Focused lab assignments closely followed lecture material
	1	"see formulas and concepts from class plug directly into code"
	5	Progressive or incremental learning of Rendering Pipeline
	5	Hands on approach / Practice of concepts
	2	Cohesive overall picture -- get to see total package from day one

	6	Learned a lot about the Rendering Pipeline
	1	Challenging
	2	Good interactions with TA's and fellow students
Ease of Use		
	7	SLIDE language simple yet powerful
	4	SLIDE objects, instances, and groups made it easy to build simple scene hierarchies
	1	Primitive objects
	4	Tcl procedural object generation more convenient than flat data files
	5	Renderer program was easy to use
	6	On line documentation was good
	2	Good example .slf files for labs
Interactivity		
	3	Renderer runs with real time interactivity
	1	Crystal Ball UI
	6	Tk widgets useful for controlling object parameters
Labs' Source Code		
	11	Extensible and Modular C++ framework was easy to code in
	2	Learned about software engineering and good coding style
	1	Liked the Hungarian variable naming convention
	9	Fresh start for each lab assignment
	5	Areas of modification are well commented in the code
	2	Portability of code (can work at home)
	1	OpenGL is useful for debugging
	1	Fast to compile
	1	Much of the tedious coding done in advance

2. The things that should be redesigned or fixed are:

Category	Repeats	Student Comment	Response
Documentation			
	11	More documentation (i.e. Assignment 3)	
	4	More examples	
	1	Better OpenGL documentation	
	1	Better Tcl/Tk documentation	
	2	Teach a Tcl/Tk programming primer in the beginning of the semester	
	1	Better explanation of Scene Hierarchy	
	1	Assignment 3 is too hard	
	1	Better presentation of relevance of pure linear algebra	
	8	Matrix Transformations are hard	
Source Code			
	15	More comments in the code	More commenting of fields and procedure requirements will be added, but a balance must be set so that the students will be forced to reason out the problems themselves instead of simply following a list of instructions.
	4	Comment all flags better	
	2	Pre- and Post-condition comments for all Procedures	

	2	Some method names are counter intuitive (Set which returns reference which can be assigned a value)	
	2	Interface to Vector and Matrix package is unintuitive	
	2	Remove all Hungarian notation	Hungarian naming convention gives a short hand for type information. This is a valuable form of commenting and will not be removed.
Portability			
	5	Precompiled binaries and libraries for more platforms	We were only supporting the UC Berkeley instructional machines this semester, although the software is portable to any system which supports OpenGL and Tcl/Tk. In the future, SLIDE will be supported for arbitrary computers over the web.
	2	Dependence on environment variables and initialization scripts (window.tcl)	This needs to be addressed for packaging the system for download
SLIDE Language			
	1	True mirroring transformations	
	1	Switch cameras without totally modifying transformations	This is fully supported by having multiple render statements for the same viewport. Need to document this feature better.
	1	Give SLIDE viewports Tk path names for the bind command to use	As we move away from the SLIDE render statement controlling input more to having it all done in Tcl, we need to address this.

	1	Annoying that SLF_MOUSE_X and SLF_MOUSE_Y only update with mouse button down	Again need to revisit issues of data input to the system.
SLIDE Renderer			
	5	Memory leaks when reloading files	There are issues having to do with multiple windows which we need to redesign to fix this problem.
	3	Too slow	Profiling needs to be done, but it does work reasonably fast for reasonably sized scenes.
	1	Default for cylinder Z-slices seems to be 8, should be 1	
	1	Two sided lighting for SLF_HOLLOW	
	1	LOD flags don't work	It is a known bug that SLF_EDGES and SLF_BOUND will not work on static trees under OpenGL because the OpenGL display list mechanism will not allow it.
	3	Lights and Surface Reflectivity	There are problems having to do with translating SLIDE values into OpenGL light descriptions, but there may be implementation bugs on top of that which still need to be fixed.
	1	Point and Spot Lights do not illuminate much	This is a result of the parameters chosen by the student for the Phong lighting model.
	2	Error messages easier to read by sending to a file	Redirect output of SLIDE renderer to a file.
	5	Better error messages especially for Tcl blocks	
	1	Better debugging of .slf embedded Tcl code	

	1	Better error checking of data, viewer sometimes assert fails on bad values	It is difficult to do full checking when the .slf file is allowed to write any random value it wants. I am sure there are bugs to fix of this kind.

Based on the comments collected in this survey of CS184 students in Spring 1999, the consensus is that SLIDE is a beneficial tool for learning Computer Graphics. The following is a summary of the positive and negative aspects of SLIDE as told by the students.

The students had many positive comments about SLIDE. They were motivated to keep up with the pace of the assignments because the subject matter is fun and the results of the assignments are rewarding and interactive. The labs have well defined descriptions and keep pace with the materials presented in lecture. The labs incrementally build up a complete software rendering pipeline. Taken as a whole, the labs form a well structured, cohesive system built in C++ on top of OpenGL and Tcl/Tk. Each lab starts with the solution to the previous lab and incrementally adds new rendering functionality. This fresh start policy was a welcome safety net for students, because it prevented them for being penalized all semester for any mistakes made in early assignments.

The SLIDE language is simple enough such that students could readily learn and apply it to build scenes. At the same time, it is powerful and flexible enough for students to build interesting, interactive 3D projects with it.

The negative comments of the students will be taken into account in refining the SLIDE system. Many students complained that the on line documentation was confusing and incomplete at times. A concerted effort was made through the course of the semester to provide enough information for the students to learn and complete their assignments, but more work needs to be done to refine the documentation and make it easy to reference. There is a common complaint that there were not enough comments in the provided skeleton code. The suggestions to write pre- and post-conditions for all the procedures and to describe all the flags better will be taken into consideration on the next iteration of the assignments. However, it is important not to over-comment or else the students will end up just following a list of instructions instead of reasoning out their own solutions. It is important to make sure the students become self sufficient programmers even though they are working within a frame work. This is a difficult balance to strike.

Many students complained that the work load was too heavy. Others complained that they were so bogged down by coding and implementation that they did not feel that they were learning the concepts. Based on these comments, it appears that we need to make a greater effort at providing conceptual materials in an easily accessible, streamlined manner. We also need to do a better job at motivating the students to digest the material before they start coding. We believe that if we could get students to follow these practices, then they would find that the actual implementation time would be greatly reduced.