CS1230 Student Guide

Introduction to Computer Graphics, Fall 2019

1 Course Information

Computer Graphics is a flourishing field within Computer Science. In this field, we study methods for digitally synthesizing and manipulating visual content. Today, this field touches many aspects of our daily lives: from animation, computer games, art and special effects to graphical user interfaces, information visualization, industrial design and education, computer graphics plays an increasingly important role in our lives, both practically and culturally.

This course will introduce you to fundamental concepts in 2D and 3D Computer Graphics. The topics covered will include image processing, rendering geometric primitives, 2D and 3D transformations, color theory, 2D image filtering, simple illumination models, user interfaces, virtual and augmented reality, and real-time graphics on the GPU. The Brown University Computer Science Department has a world renowned computer graphics program, and several of our faculty members are pioneers in the field.

The primary text for this course is the latest edition of Computer Graphics: Principles and Practice, by John (Spike) Hughes, Andy van Dam, Morgan McGuire, David Sklar, Jim Foley, Steve Feiner, and Kurt Akeley. Although not required it is highly recommended and is currently available through Amazon.

2 Lectures

Every year, we receive comments from students in questionnaires noting that sometimes our lectures seemed unnecessary, given the detailed nature of the slides. Nevertheless, a strong correlation has been shown between those who do well in CS1230 and those who come to class. Lectures are enriched by class discussion, live demos and other content beyond the slides themselves. We strongly encourage class attendance.

We don't have nearly enough lecture time to teach everything we wish we could. The lecture topics have been carefully selected by Andy and the TAs, taking into account the feedback we receive from former CS1230 students. If you feel like you aren't getting enough out of the lectures, we encourage you to talk to Andy or the TAs rather than abandoning them altogether. We are constantly revising CS1230, and we take all feedback under serious consideration during our weekly staff meetings.

Sick? Please do not come to lecture or use public computer labs if you have a communicable illness (like the flu). Ask a friend to fill you in on details from class, and review the lecture slides online. We would like to keep everyone in good health.

3 Help Sessions

There will be several CS1230 help sessions throughout the year. There are two on C++, another on linear algebra, and a third on the shader programming language GLSL. Dates and times will be on the course calendar.

If you feel that a particular help session would be useful to clarify a topic, you may contact the Head TA (at cs1230headtas@lists.brown.edu) to discuss your proposal.

4 Programming

Because graphics work can be computationally intensive, CS1230 is one of the few remaining courses at Brown to use C++, and the only remaining course to teach it. In computer graphics, the performance implications

of managed language features like garbage collection and memory bounds-checking can be unacceptably high. We promote object oriented coding practices to ensure maintainability and extensibility as your graphics system grows. One of the key techniques you'll take away from CS1230 is the ability to carefully balance raw performance with excellent code maintainability and style.

4.1 Toolkit

Creating GUIs in C++ is often a hassle. We use Qt to achieve cross-platform UI functionality.

Because Qt depends on special preprocessors and compilers, we recommend that you use Qt Creator to author your C++ code. Qt Creator features automatic code completion, integrated debugging, and automatic Makefile management. Both Qt and Qt Creator are free (under the LGPL license) and cross-platform.

Where is Qt Creator installed?
For your convenience, Qt Creator is installed in several locations throughout the CIT.

- All department Linux machines
- All department Windows machines
- All CIS cluster Windows machines on the second floor of the CIT

You may also install the Qt SDK on your personal computer to work from home.

Regardless of your choice of development platform, you must ensure that your code compiles and runs properly on the Linux machines in the Sun Lab.

4.2 Software Engineering

We expect that you have all had a thorough grounding in the principles of good software design by now. Most of the assignments are relatively small, and shouldn't require hours of design work. We care mostly about the functionality, stability, and speed of your implementation; however, we will also grade partially on code quality. Good engineering practices early on will greatly help you on future assignments.

Basic engineering concepts
Avoid repeated code by thinking about good class design in advance.

Don't do anything grossly inefficient. For example, factor repeated computation out of loops.

Always remember to free your memory.

4.3 Don't know C++?

Students are able to take the course without knowing C++. However, these students will need to spend extra time at the beginning of the course learning C++. The Intro to C++, Intermediate C++, and Advanced C++ help session slides are available on the CS1230 documents page. These help sessions will be run by the TAs at the start of the semester. See section 3 above.

TAs are also ready and willing to help students with C++ questions at office hours.

Aside from the help sessions and TA help, the course has no special provisions for students learning C++. Programs will be due at the same time for everyone and will be graded on the same scale.

5 Assignments

5.1 Projects

5.1.1 Brush

The first assignment is designed to get your feet wet in the world of graphics programming. In this assignment, you will be implementing various different airbrushes, similar to ones found in many commercial painting programs such as Adobe Photoshop. This assignment should give you a good introduction to the kind of C++ programming you will be doing in this course, as well as gently familiarize you with Qt user interfaces.

5.1.2 Shapes

This assignment covers one of the earliest steps in the 3D rendering pipeline: object tessellation. For this assignment you will be constructing simple 3D objects (e.g., spheres and cylinders) out of triangles and then displaying them on the screen. All you need to do for this assignment is compute the necessary triangles; OpenGL handles the task of drawing them for you.

5.1.3 Filter

Ever wonder how programs like Photoshop generate all those cool special effects? This assignment represents a subset of the functionality that photo editing programs have. It is designed to teach you the basics of image processing and anti-aliasing. You will implement various image manipulation operations like edge detection, blurring, and image scaling.

5.1.4 Sceneview

By this assignment you will know how to display, transform and view 3D objects. The next step is for you to put these tools together to create a scene consisting of arbitrarily positioned objects viewed from an equally arbitrary location. In this assignment, you will build your own viewer for moderately complicated scenes.

This assignment builds upon the code from Shapes and from the Camtrans lab.

5.1.5 Intersect

In this assignment you will compute intersections between the common objects you tessellated in shapes and rays (of light). You will then use these classes to create somewhat photorealistic images of scenes.

This assignment builds upon your Camtrans and Sceneview code.

5.1.6 Ray

Ray tracing is a method for rendering realistic pictures of geometric objects. It uses available information about lighting and optical effects like light reflection and refraction. It may sound complicated, but it is a relatively simple technique, and the cool pictures you get are well worth it! You will essentially be taking your code from Intersect, adding in the ability to illuminate the objects, and applying textures from 2D images.

This assignment builds upon your Camtrans, Sceneview, and Intersect code.

5.1.7 Final Project

Lastly, you will be creating a final project. The final project can be anything you want that includes one or more of the concepts we covered during the semester, plus a technique you research on your own. Final projects must involve GPU shader programming. Examples include a short game, a cool OpenGL shader program, a demo scene, an implementation of an advanced rendering system, or some combination of the

above! Note you're not restricted to these ideas alone. We'll cover the final project in greater detail in late November.

5.1.8 Labs

In addition to homework assignments, a series of labs will provide hands-on experiencing using OpenGL, a popular real-time graphics programming library. The concepts you learn in lab will prove quite useful for implementing your final project (see section above).

5.2 Algos

5.2.1 Algorithm worksheets

Each programming assignment will be accompanied by a written algorithm ("algo") assignment to get you started thinking about how to approach the assignment mathematically and algorithmically. These hand-ins will contribute 10% of the final grade. There is also an algorithm assignment for the Camtrans lab. See the assignment handouts for more details and exceptions. Algorithm worksheets are returned the same day as the deadline so you can begin coding with confidence right away!

Algorithm answers should be clear and succinct. Don't just start writing. Think first, and then write up the clearest answer you can. If we ask you to describe a section of the project's algorithm, you should probably go with a description, or pseudo code, rather than actual C++. That said, some incredibly simple parts, such as loops, may be easier to read in actual code.

5.2.2 Tips for Algorithm worksheets

When you think you're done with the algorithm worksheet, begin coding immediately while it's still fresh in your mind. Don't wait for the solutions to begin coding. The solutions to the algorithm worksheets are designed to help you to debug your program, not to help you write the code for the first time.

Take it seriously! Expect to spend between 1 to 2 hours on each algorithm worksheet. Late algorithm worksheets aren't accepted, so be sure to turn them in on time.

5.3 Optimization

While we don't expect you to go overboard with optimization, we do expect your programs to perform well. Be sure to factor repeated computation out of loops, but don't feel compelled to write assembly code or special SSE instructions.

Memory Management: Rather than repeatedly allocating bits of memory, allocate a large chunk all at once. Use smart pointers and STL or Qt data structures that automatically manage your memory when approrpiate.

Orders of Magnitude: Be aware of the differences (especially with regard to O(n) performance and memory overhead) between different data structures. We'll dock points for big memory leaks and things you obviously should have factored out of your loops.

Accuracy: Above all, don't sacrifice accuracy for performance! You will not lose points for an efficient, yet accurate, program. If you ever have a question about optimization, ask a TA on hours.

5.4 Final Grades

Your final grade will be determined based on the programming assignments, algorithm hand-ins, and labs.

You must complete all the assignments (including 10 out of 11 labs) to pass the course. As in all other

computer science courses at Brown, you must hand in a working solution for all programming assignments in order to be eligible for a passing grade. Please note the adjective working: if you receive a grade of NC (no credit) on any of the assignments (before late penalties are deducted), you will be expected to revise your program and hand in an acceptable version if you want to pass the course. Also note that the reverse implication is not intended: handing in all assignments does not guarantee that you will pass the course; your accumulated points will determine that.

Andy doesn't use a curve, and would be delighted to hand out As to the entire class. Indeed, the majority of students traditionally have worked hard and gotten As. In borderline cases (e.g. 89-91), Andy will take attendance and class participation into account, as well as your perceived effort and dedication. We all love to both give and receive good grades, but do understand that merely working hard doesn't guarantee you an A. Your grade will reflect primarily upon the quality, correctness, and timeliness of your hand-ins.

5.5 Extra Credit

There is ample room for bells, whistles, and other credit-garnering efforts on the part of ambitious programmers. You are invited to get creative, as long as it does not make you late. Rewarding bells and whistles with extra credit is left to the discretion of the TAs, so we strongly encourage you to discuss your creative plans with a TA before you forge ahead to make sure that they are considered appropriate for credit. Also, keep in mind that bells and whistles should only be done after the standard assignment is fully working since they won't count in lieu of missing or buggy features! CS1230 offers many opportunities for extra credit: if you finish a program a little early, seek appropriate inspiration and add something fancy.

5.6 Half-Credit

Students wishing to receive an extra half-credit for taking this course must also register for CSCI-1234 and complete specific requirements for each project. More details can be found in the student half-credit guide found in the docs section of the course website. Students who wish to earn 2000 level graduate credit will need to use the half-credit option. See the half-credit missive for more details.

5.7 Workload

This set of assignments probably looks like a burden, but in fact, if approached sensibly (i.e., working steadily), you will have sufficient time for each and every one. The normal load is about 15 hours per week. By the time you take CS1230 you are expected to be a competent programmer, with good design and debugging habits, and able to turn assignments in on time. Good time management will make this course much more enjoyable!

5.8 Rescoring Requests

Sometimes you may feel that you have been graded unfairly. If you ever feel this way, please talk to the TA who graded your assignment by visiting them during office hours. If you are not satisfied with the TA's explanation, talk to the Head TA. If there is still a problem, Andy is the final word in grading and will be happy to hear what you have to say.

If you decide to challenge a grade, you must do so within two weeks of its receipt. In the past, students have tried to get points back on all of their assignments in the last week of classes. Our first priority is fairness, both to us and to you! You may ask questions about your grades at any time. You can't change the contents of your hand-in after your grade is returned. If you discover that you handed in the wrong work after you get your grade back, or if you fix your program after getting its grade back, we are unable to take that "external" content into account for grading purposes.

5.9 Support Code Tips

5.9.1 Vectors and Matrices

The support code uses a library called glm for vector and matrix operations. Glm contains vector types called glm::vec2, glm::vec3, and glm::vec4, as well as matrix types such as glm::mat4.

Examples:

```
glm::vec2 pos; // Default constructor initializes x and y to 0
pos.x = 1;
glm::value_ptr(pos)[1] = 3;
std::cout << glm::to_string(pos) << std::endl; // Outputs [ 1 3 ]

glm::vec3 v = glm::vec3(1, 2, 3);
glm::vec3 n = glm::vec3(5, 2, 0);
float dotProduct = glm::dot(v, n); // dotProduct == 9
for (int i = 0; i < 3; ++i)
   glm::value_ptr(n)[i] = 2 * i;
glm::vec3 piecewiseProduct = v * n; // piecewiseProduct == [0, 4, 12]
```

5.9.2 Qt Creator

Qt Creator has lots of shortcuts and tricks that can make your life a lot easier. Here are some of the most helpful shortcuts:

- F2 Go to the definitions of the symbol under the cursor.
- F4 Switches between the header and cpp file.
- Ctrl+Shift+R Rename and refactor the symbol under the cursor.
- Ctrl+Shift+U Find all usages of the symbol under the cursor.
- Ctrl+L Go to a specific line number.
- Ctrl+K Godmode - Search for ANYTHING. If you learn one new shortcut, this should be it.

Examples:
Type Ctrl+K followed by "m paintGL" to see a list of all methods named paintGL. Hit Enter to go to the implementation of the selected method.
Type Ctrl+K followed by "c Canvas2D" to see a list of all classes named Canvas2D. Hit Enter to go to the Canvas2D class definition.
Type Ctrl+K followed by "? QList" then hit Enter to view the Qt Help on the QList class.

For more information, hit Ctrl+K and read what each letter prefix does.

5.9.3 Floating Point Calculations

Keep in mind that floating point numbers have limited precision, so floating point calculations may lead to rounding errors. To compare two floating point numbers x and y for equality, use abs(x-y) $<$ EPSILON. You can define an EPSILON that works well for your program (usually 1e-4 to 1e-8).

5.9.4 Settings

The support code defines a global object named settings to hold values entered through the GUI. This object is automatically updated when things are changed in the GUI, though the GUI is not updated when the settings object is changed programmatically.

6 Getting Your Questions Answered

For most course-related questions, you should ask a TA during TA hours.

You may not ask TAs course-related questions when they are not on hours. All TAs are prohibited by department and university policy from answering course-related questions when not on official TA hours. TAs may answer general administrative questions when not on TA hours.

Thank you for helping us comply with departmental and university regulations (which reflect applicable labor laws). If you have any questions about this policy, please contact mta@cs.brown.edu.

6.1 TA Hours

The TA hour listing is available from the CS1230 home page. We will make every effort to maintain these hours without exception. On rare occasion, we might need to cancel or reschedule a TA hour session. When TA hours are rescheduled or exceptions are made, these will be announced on the CS1230 web page.

TA hours are often very busy, and many times there will be a waiting list on the board. You may sign up on the waiting list when TA hours are in progress, but not before TA hours start. The TA will erase the waiting list at the start of TA hours each day. If the waiting list is very long and the end of TA hours is approaching, the TA may close the waiting list so that all people on the waiting list can be helped before TA hours are over. You can help alleviate this problem of crowded hours by simply starting early. While there is no bonus for doing this, starting early often means finishing in less time, since you won't have to wait as long to get your questions answered.

Before asking a TA for help, please check the lecture slides to see if your question is answered there. If you still do not understand something, you are welcome to ask questions and clarifications about the lecture slides.

Please note that taking any source code (or a TA's modifications to your source code) away from TA hours is a violation of the CS1230 collaboration policy. The TAs are not here to debug your code, but rather to clarify your understanding of the concepts taught in class and used in the assignments.

6.2 Sending mail to the TAs

If there is a problem with an assignment that affects the entire class (such as a bug in the support code), you may e-mail the TAs at cs1230tas@lists.brown.edu. TAs will respond to course-related e-mail during their office hours. For administrative questions or if there is a problem with a TA, you can e-mail the head TA at cs1230headtas@lists.brown.edu.

Questions related to coding the projects or concepts explained in class should be asked at TA hours and TA help sessions. We are not able to respond to these types of questions via e-mail.

6.3 Course webpage

The CS1230 web page, which you are responsible for checking and reading daily, is located at http://www.cs.brown.edu/courses/cs123. It is a useful source of online course material. A wide variety of things like lectures, handouts, the syllabus, TA hours, and software guides can be found on this site.

You are responsible for checking the web site frequently, as we will be updating it with important information as the semester progresses. Most updates to the website will be accompanied by an announcement to the course mailing list and Piazza.

7 Accomodations

Students requiring special accommodations should see Andy or the Head TA at the beginning of the semester (or as soon as an extenuating circumstance arises) to make arrangements. We are unable to retroactively apply special accommodations, so be sure to contact us as soon as possible!

8 Finally, welcome!

As we mentioned before, CS1230 is a continually evolving course. It has been updated in recent years to use and teach modern version of OpenGL. In addition, tweaks and fixes have been made across the board. As such, we are bound to have our own 'bugs' hiding in the corners. Please read everything we hand out very carefully. If there is something which you do not understand, or which is not stated very clearly, please let us know so we can fix it right away.

This applies to the material discussed in lecture as well. Give us constructive criticism on all aspects of the course. The more feedback there is, the better we can make this course for you as the semester progresses, and the better we can make it for the next twenty-five years of Brown computer graphics students!

We just went through a lot of heavy talk. Please take all of it seriously, but also remember that we're not trying to scare you. We are here to help you! We're just clearing preliminaries and establishing the ground rules. With that done, we hope you'll have as much fun as we did in CS1230.