Graphics Software Architecture

Syllabus

Bulletin Description

Fundamentals of modern software graphics, and how these can be efficiently implemented in C/C++. Core topics include geometric primitives, scan conversion, clipping, transformations, compositing, image sampling. Advanced topics may include gradients, antialiasing, filtering, parametric curves, geometric stroking. Students will develop a graphics engine, evaluated for functionality, clarity, and performance.

Textbooks and Resources

There is no textbook for this class; (we have the internet).

Course Description

Computer graphics are everywhere, so much so that we often lose sight of the fact. Desktops, set-tops, laptops, tablets, book-readers, phones, and now watches, all have displays to communicate with us, and the applications they host (web browsers and mobile apps mostly) rely on graphics engines to do the actual work of pushing pixels.

In this course we will create a 2D graphics engine from the ground-up. It will begin simple, but over the course it will be extended to add new APIs and capabilities.

- colors
    - blending
    - textures
    - advanced : filtering, dithering
- geometry
    - filling rectangles, polygons
    - triangles
    - advanced : splines
- transforms
    - matrices
    - clipping

Our engine will also be the test-bed where we learn to flex our Computer Science muscles, revisiting core ideas through the lens of graphics.

- integer and floating point numerics
- memory organization and alignment issues
- compression
- performance
- debugging

Target Audience

The course has been designed for advanced undergraduates and first year graduate students, who are already comfortable coding in C/C++, and are ready to apply and extend those skills in the domain of computer graphics. Graphics is an old field, but one with constant pressure from the "real world" to push software techniques to match the latest HW developments. If you want to know how the CPU really works, what low-level operations are slow or fast, or just what goes on when your phone screen lights up, this may be the class for you.

Prerequisites

Basic System Architecture, Algorithms, Data Structures
Comfortable coding in C/C++
Basic Linear Algebra and Differential Calculus

Goals and Key Learning Objectives

Students will learn to write a 2D graphics engine from the ground-up. They will learn how to structure a library's API. They will learn to test the correctness, improve the readability, and measure the performance of their code, and they will also encounter the inevitable tradeoffs that can occur when they try to optimize for all three.

Course Requirements

Course assignments are organized around designing and refining a graphics library in C++. Some time will be devoted in class to coding/debugging, but much of that work will need to occur outside of class.

Students are encouraged to discuss ideas and techniques with their peers, but each student is to build their own library, writing all of the code themselves.

Grading Criteria

Your grade in this class will be based on:

- Programming assignments (approx 90%)
- Final examination (approx 10%)

Course Policies

All programming assignments are due at 6pm on their due date. Each 24-hour period following the original deadline will reduce the assignment's score by 15%.

Exceptions to this requirement must be approved by the instructor in advance.

Honor Code

Students are encouraged to discuss assignments and various strategies and techniques to accomplish them. They are also encouraged to share general coding techniques (esp. around debugging and performance). In all cases, however, the organization and implementation of the assignments must be the sole work of the student.

Unacceptable collaboration on assignments includes:

- Copying or sharing (verbatim or substantially similar) code.
- Working with individuals who are not presently members of the course.
- Code that is jointly authored or authored by entirely or in part by another individual.

If you copy or transcribe one or more lines of code from any other student/individual, or copy any code from any website or internet source, or base any code found on any website or internet source without attribution, you are in violation of the Honor Code.

Course Schedule

Approximate sequence of topics, with 6 programming assignments. Specifics will vary as needed.

Day 1	Course Introduction
Day 2	Color Theory
Day 3	Framebuffers
Day 4	Blending

Day 5	**PA1** Due -- review in class
Day 6	Coordinates
Day 7	Polygons
Day 8	Clipping
Day 9	More Blending
Day 10	**PA2** Due -- review in class
Day 11	Matrices
Day 12	Shaders
Day 13	Sampling
Day 14	**PA3** Due -- review in class
Day 15	Concave polygons
Day 16	Concave polygons 2
Day 17	Paths
Day 18	Gradients
Day 19	**PA4** Due -- review in class
Day 20	Curves
Day 21	Gradient Tiling
Day 22	Image Tiling
Day 23	**PA5** Due -- review in class
Day 24	Quads
Day 25	Triangles
Day 26	Compose Shaders
Day 27	Triangles 2
Day 28	**PA6** Due -- review in class

Disclaimer

"The professor reserves the right to make changes to the syllabus, including project due dates and test dates. These changes will be announced as early as possible."