

L01 Introduction

Jarek Rossignac

www.gvu.gatech.edu/~jarek

1 - Overview

Welcome to this series of Lectures In Graphics.

This chapter discusses the scope and the organization of the course.

1.1 Objectives of this course

In this course, you will learn how to represent and process shapes and animations in a computer so as to support interactive design, useful rendering, and analysis. The course has 4 intertwined threads:

Model: How to represent curves, meshes, and surfaces

Control: How to allow the user to edit shapes and views

Animate: How to represent and display behaviors (move, bend, morph, bounce) of shapes

Render: How to simulate light, use the GPU, and enhance visual clarity or aesthetics

The course is focused on the mathematical and algorithmic underpinning of computational techniques for creating, processing, and rendering shapes and animations. It is designed for undergraduate students who are interested in developing software systems, modules, and applications in which users create, manipulate, visualize, and interact with shapes or animations. It is also designed to be used as support for graduate students who are interested in a research or R&D career in graphics or who want to learn modeling, animation, rendering, and visualization techniques to help them in a different discipline.

The course uses the Processing language for projects and demonstrations and discusses the programming of shaders, does not teach how to use commercial or public domain modeling or rendering tools or packages. These lectures are also useful to graduate students as a basis for further exploration.

 Search for and make a list of the most popular packages for designing shapes and animations.


1.2 Motivation and applications

I expect that most students take this course because they are interested in a career as a graphic or animation software developer and wish to either work directly for the entertainment or manufacturing industries or for companies that develop tools for these industries. However, I also expect that students from various application fields, such as medicine, architecture, or data visualization take this course to obtain important skills, which they will use to prototype custom tools for visualizing complex shapes and structures.

 Make a list of the 10 largest companies likely to hire students with the skills taught in this course

1.3 Relation to other courses and prerequisites

A variety number of courses, textbooks, and online courses on this topic are available. Students are encouraged to explore these and compare their scope and explanations with those of this course. The course is designed to cover a broad set of modeling and animation techniques and solutions and emphasizes intuition, simplicity and elegance rather than mathematical completeness and precise formulation. Most topics are illustrated with images, animations, or hands-on interactive demos. Students are expected to be comfortable in using an object-oriented programming language and standard algorithmic concepts, such as arrays, loops, stacks, and recursion. Familiarity with Processing or Java is an advantage, but is not required.

 Identify 3 other online or video courses on this topic

Prerequisite skills (students need to **know** these tools before taking the class):

Linear algebra in 2D and 3D:

Points, Vectors, Matrices, simple geometry (line, circle), Solving Linear Systems...

Math basics:

trigonometry, complex numbers, polynomials, roots, exponentials, derivatives, integrals, Boolean logic

Programming:

Processing or Java, classes and objects, arrays, iterations, recursion, top-down design, debugging

Logic and intuition:

Aptitude for understanding abstractions and for solving logic problems via algorithms

http://en.wikipedia.org/wiki/How_to_Solve_It

<http://algorithmicproblemsolving.org/aps/what-is-algorithmic-problem-solving/>

Students are assumed to know about vectors and their operators and the basic concepts covered standard Algebra and in Linear Algebra courses. I will briefly go over all algebra that is needed here, but many students may need to review what they have learned in these classes. I suggest that they use online video lectures.

 Identify a popular video lecture explaining how to solve a quadratic equation and watch it

 Identify a popular video lecture explaining how to solve a linear system of 2 equations and watch it

1.4 Instructor

My name is Jarek Rossignac. I worked as a Research Staff Member and then Manager, and Senior Manager at IBM Research and then as Professor at Georgia Tech and Director of the GVV Center. My research interests include: efficient representations, algorithms, and GUIs for designing, processing, visualizing, and disseminating shapes and animations. I published over 150 papers, obtained 20 patents, delivered 20 Keynotes & Distinguished Lectures, received 24 Research or Best paper awards, chaired 20 international conferences, 6 Program Committees, and 4 Best Paper Award juries, and served as Editor-in-Chief of GMOD and as Associate Editor of VTCJ. I also Chaired the Solid Modeling Association, Served on 82 International Program Committees, and am a Fellow of the Eurographics Association.

2 - Syllabus

The course is divided into 24 lectures. Part 1 (the first 12) deals with the interactive design, display, and animation of planar shapes. Part 2 deals with 3D modeling and rendering of triangle meshes, with the GPU organization and use, and with colors, light, and perception.

PART 1: 2D GRAPHICS, MODELING AND ANIMATION

1. **Introduction:** scope, expectations, grading
2. **Processing:** 2D graphics, transforms
3. **Points:** Affine combinations, coordinates, barycentric, GUI
4. **Vectors:** polar, complex, and Cartesian representation, dot & det products, steady interpolation, rays, lines
5. **Frames:** (isometric, conformal, general) affinity, registration, steady motions
6. **Polygons:** half-space, triangle, NM, holes, containment, intersections, Booleans, CSG, active zone, BSP, BList
7. **Curves:** line, circle, parabola, Bezier, Hermite, Neville, NUBS, continuity
8. **Polyline:** curvature, resampling, cubic smoothing, constrained, subdivision, J-splines
9. **Motions:** translation, rotation, spiral, driving/skidding, physics
10. **Collisions:** particles, disks, offsets, polygons
11. **Deformations:** space warp, skeleton bending, skinning, area preservation
12. **Morphing:** linear, curvature, Minkowski, and ball morphs

MIDTERM

PART 2: 3D MODELING AND RENDERING

13. **Graphics:** primitives, reflections, camera, lights, pick, drag, see-through
14. **Geometry:** dot, cross, mixed, primitives, intersections, projections
15. **Triangulations:** ops, swingRings rep, files, tessellate NUBS, pick, modify
16. **Rendering:** normals, smooth shading, silhouettes, see-through, texture mapping, strokes
17. **Segmenting:** components, loops, genus, isolation, skeleton
18. **Smoothing:** borders, interior, subdivision, adaptive
19. **Skinning:** resampling, Frenet-trihedron, twist compensation, bifurcations
20. **Lighting:** reflection/refraction, optics, surface properties (lambertian, specular),
21. **Rasterization:** scanline fill, Gouraud, Phong interpolation, Z-buffering

22. **Perspective:** vanishing points, projection, transform, properties, inverse
23. **Shaders:** GPU architecture, languages, functionality, limitations, texture mapping, shadows, CSG rendering
24. **Perception:** eye, acuity, color resolution, Gestalt, Pregnanz, illusions, memory, aesthetics

FINAL



Look at the syllabus of 2 textbooks in this area and identify the major differences with this one.

3 - Resources and study process

3.1 Material needed

T-square account

Top Hat account: <https://tophat.com>

Top Hat quick started guide for students:

[https://s3.amazonaws.com/thm-corporate/Support/Guides/Student+Quick+Start+Guide+\(M\).pdf](https://s3.amazonaws.com/thm-corporate/Support/Guides/Student+Quick+Start+Guide+(M).pdf)

In the Top Hat Lobby, search the course using my name (Rossignac) or course number (CS3451)
or use the short-cut number 244264 .

Remember the phone number 315 636 0905 in case you need to enter answers using your phone.

PowerPoint presentation of Top Hat:

<https://s3.amazonaws.com/thm-corporate/Support/Guides/Intro+To+Top+Hat+Slide.pptx>

Textbook: **Shirley: Fundamentals of Computer Graphics, 3rd Edition**

Laptop with wifi access in class

Processing: download and install two versions of processing

<https://processing.org/download/?processing>

Version 2.1 or later, so that you can use the latest version

Version 1.5.1, because some of my older demos are not compatible with the new version

So, when using my older applets, if they do not work in the latest version,
try running them in 1.5.1

Paper and pencils/pens of different colors

3.2 What you need to study before each lecture

I plan to run this class in reverse mode:

- you study at home, before class
- you work on your project, explore applications, and practice solving problems during class

Here are all the things, which you will have to study BEFORE each lecture (plan for 3 hours):

- Review the notes you took in class of the **previous lecture**.
- Review the **solution**, presented in class, of the **previous project**.
- Study the assigned **course notes** for the (forthcoming) **lecture**. About 8 pages per lecture.
 - Read, study, understand, practice, and retain the information in these course notes. (It is not sufficient to know the name of a concept and where to find it. You need to understand the concept intuitively and remember the formulae or be able to re-derive them quickly.)
 - Examine referenced links to web pages or papers
 - **Answer** online **TopHat questions** dispersed in these readings
- Watch the **video Lectures** (if available) for the lecture
 - Note that the short video lectures (when posted) may provide a nice, high level overview and introduction to the topic, but do not replace the notes below (which you need to study on your own)
- Read the assigned sections from the **textbook**
 - These often provide additional information, a different perspective, or define a different notation
 - You need to read and understand the assigned sections
- Read additional notes or **papers** (when assigned)

- It is important for many of you to be able to read and understand contemporary articles in graphics.
 - Hence, I will ask you to read some during this course.
- Watch videos of demos provided for that lecture
- Download and play with the sketches provided as demonstrations for the lecture


In addition to all this **study work**, you will have to do a **microProject** for each lecture, see below.

3.3 What you need to do during each lecture

- Attend (email instructor if you need to miss a lecture for travel or family/health hardship)
- During the lecture, I will not attempt to re-explain or re-derive what is included in these notes.
 - Instead I will assume that you studied them and have retained the notions, notations, and formulae they teach.
 - I will test you on these first.
 - Then, I will answer specific questions about that material in these notes and try to clarify them and give you more intuition.
 - I will challenge your understanding of these concepts with lots of problems that you will have to solve in class, either individually or in teams of two.
 - Finally, I will discuss applications of these tools to some useful techniques in modeling, animation, rendering.
- Take lots of notes of what is discussed by the instructor and by other students
- Participate: ask questions, suggest answers, be active in team exercises
- Bring your laptop to class every lecture: We will start your next MicroProject in class!

3.4 Students' questions

In each chapter, I will list questions that students have asked or that I think students should have asked. I will also provide answers or suggestions for searching answers.


 Write down questions you may have and ask them during the next class.

3.5 Accompanying sketches and videos

To provide the reader with an intuitive understanding of some of the more complex concepts presented or introduced here, I will provide accompanying videos and interactive Processing sketches that the students can use to explore the concepts and variations of their implementation.

3.6 Links to additional resources

There is a wealth of complementary material that might benefit students who may have a difficulty grasping a particular point or students who are eager to go further and learn about the more rigorous mathematical formulations or extensions. I will include links to such material posted online. The students should anticipate that the naming and notation used in these sites may differ from one source to the other and from those used in these chapters.

 Test that you can watch the videos provided for this course and that you can unzip and run the Processing applets.

3.7 Suggestions for additional reading

For each chapter, I will refer to related sections of “Fundamentals of computer graphics”, 3rd Edition, by Shirley et al. That book exposes material in a different order and puts more emphasis on realistic rendering and filtering and much less on modeling, on shape and animation design, and on shape processing.

 Browse through the Textbook and identify sections that are not covered in this course

4 - MicroProjects

I plan to have a microProject due every lecture (except lecture 01). Although I am contemplating replacing one or two series of these with a larger project. I want your opinion on this, especially after the first few microProjects.

4.1 Guidelines for doing and submitting microProjects

Guidelines for writing the project reports and submission projects

- Download and unzip the base code and study it.
- Think about the project in advance and write down questions to ask during the lecture before the project's deadline.

- Fill in the file header of the main tab with your name, project number, title, and date.
- Make sure that you replace the data/pic.jpg file with a file of the same name showing a clear picture of your face.
 - o Your pic.jpg file should have roughly same dimensions as mine: about 320x320 pixels or so.
 - o Your picture should be recent and should clearly show your face.
- Edit the title and name at the bottom of the main tab to make sure that the project canvas shows the proper project number, title, and your name, so that they appear clearly and are readable on the canvas.
- Write your own code as procedures or functions in a separate tab. Make sure to put a header in that tab with your name.
- Be concise. Most of the projects require that you write only a few lines of code!
- Write elegant and readable code. You do not need to use the functions provided in the base code or their names and are welcome to change the names of variables, classes, methods... and to reformat the code to suit your coding style. But realize that doing so will make it hard for the TAs to understand your code and help you. So, your code needs to be very easy to read.
- Put brief comments as needed to show that you understand perfectly what the code does and to help TAs follow it.
- These may include discussions with colleagues (give names of colleagues and be specific as to what was discussed), help from TA (state which part they helped you with), cite books, papers, websites ... that were helpful (be specific which part was helpful and has influenced your implementation). If you find a small piece of code online that does exactly what you need, you may use it in your code, but you have to (1) ask the TAs whether this is acceptable, (2) reference it in the code and in your report, (3) change it to make it simpler, more elegant, or clearer, (4) put comments on each line to show that you understand perfectly what it does, and (5) explain in the report what the overall idea of that code and justify why it is correct.
- Add commands to draw() and to the various action procedures to call your procedure(s) and support your GUI and animation.
- Edit the 'guide' string as desired to explain the GUI of your program.
- Capture images of the canvas (!) showing results of your program and include in the write-up for your submission.
- Follow the indication in the 'utilities' tab on how to make a movie showing your sketch in action.
- Make sure that your movie is short (5 to 20 secs) and that the resolution is modest, so that the file is small.
- Make sure that in the first frame of the vide one can see the project number, title, and your name and face.
- Write a very short report containing:
 - the title: "CS3451-Fall 2013, Pxx", where xx represents the microProject number (01, 02,...)
 - author: First LASTNAME, clear image of your face.
 - project title and short description of what was asked,
 - a short explanation in English of what you implemented and how you coded it
 - (include a code snippet showing the key pieces of what you have programmed),
 - include one or two images showing interesting results produced by your program,
 - explain clearly what functionality you were not able to implement and why,
 - clearly identify EXTRA CREDIT contributions and include explanations and additional images showing results.
 - (Only up to 25% of the extra credit points can be earned per project. The amount earned will be decided by the TA, based on the average quality of all submissions, on the intellectual merit of the added functionality and on the elegance of the implementation or of the results.).
- Include the source code sketch tabs, the data folder, the movie, and a PDF of your report inside the sketch folder.
- Make sure that you DELETE the FRAMES folder and the PICTURES folder.
- Compress the sketch folder into a .zip file and submit the zipped file on Tsquare before the deadline.

No extensions! (The project solution will be discussed on due date.)

Your 3 worse microProject grades will be discarded. If we have larger projects (not microProjects) that span several lectures, these will not be discarded.

4.2 Software style

Students are expected to implement and debug one microProject per lecture. However, they will be provided with a code base for each microProject that supports most of the unessential functionality, so that they can learn from that code base and focus on the key point. Hence, students will typically have to write only a few lines of code for each microProject.

The provided code is written using a compact style for naming and formatting, which was developed by the author to structure rapid prototyping, to facilitate debugging and to improve productivity. Most functions and methods are either presented as a single line of code or divided into steps, each of which is either presented in a single line or as an iteration of such steps. Students who are more comfortable with different naming and formatting conventions are encouraged to reformat the code and to use a more self-explanatory naming convention.

- 🔗 Read a description of the Processing language and environment online and check some demos.
- 🔗 Download and install the latest version of Processing and learn how to run it.

4.3 Notation

I will attempt to use a consistent notation and typesetting to distinguish scalars, points, vectors, frames, and geometric shapes. Each chapter will contain a notation section where the meaning of new symbols is summarized. Furthermore, many of the types and operators used in mathematical expressions will correspond to Processing classes and functions.

- 🔗 How would you document the correspondences between concepts, names, math symbols, and class/function calls so that it is easy to remember?

4.4 Plagiarism and forbidden collaborations

Plagiarism: Acceptable use of publicly available material (printed or posted): you must cite the source, explain it to prove your total understanding, customize or improve the source material somehow, and document your customization.

Collaboration: You may discuss ideas with colleagues and TAs and help each other address technical problem (for example how to make a movie from a Processing animation). But you should **never show, share, or copy source code!**

Let me say that again: All the code that you add should be written entirely and exclusively by you! No collaboration on the code is allowed. You are not allowed to copy code from any other sources unless explicitly permitted by the instructor or TA. You are not allowed to show your code to any other student.

4.5 What you have learned in Lecture 01 to help you with microProject 01

1. How to open, run, stop, modify, and save a sketch
2. Mouse actions, press 'a' for free-fall animation, drag mouse and release to set velocity
3. How to switch tabs, edit the code, add comments, check commands in References, look at examples
4. Structure of the sketch: setup, draw, keyPressed...
5. How are mouse and key actions defined
6. How to record pictures of the canvas for inclusion in your microProject report
7. How to make a movie
8. What you need to modify in microProject 01
9. What to do to earn extra credit points
10. How to submit the project

Grades

Grade formula: Projects (50%), Midterm (10%), final exams (20%), Quizzes (20%)

Quizzes: Most quiz and exam (midterm or final) questions will be administered online using TopHat. Most will be multiple-choice, but some will involve writing short algorithms in pseudo-code, formulae, proofs, or definitions. Some may require producing a figure by hand on a piece of paper.

TopHat questions will be given:

Before the lecture (to be answered from home while studying for the lecture)

At the beginning of a lecture (to test whether the material for the previous lecture and the pre-class self-study material have been understood and retained)

During a lecture (to test student's understanding of what was just presented or discussed by the instructor or by other students).

During exams (midterm and final), which will contain large number of questions.

Time allocated to each question will range from 1 mn (for simple questions that test your understanding), up to 10 mns for complex questions where algorithm or formula must be derived and written or applied.

Regardless of the complexity, each question will be worth 2 points: partial credit (1 point) will be given for written answers that are almost correct, except for silly mistakes, typos, or numeric errors, or for interesting, although incorrect, solutions.

The 12 worst questions for each student will be discarded. This will help students who need to travel or experience personal or technical hardships and need to miss some lectures. But the best use of these is to keep them in reserve for the midterm or final.

Projects: Each MicroProject will be graded out of 10 points. Points will be allocated as follows:

- 5: Correct implementation of the assignment (partial credit for partial implementation or interesting but wrong solution)
- 2: Nice (short, but informative and original) video
- 2: Report (short, informative, correct), with snapshots showing the program running and explaining the solution)
- 1: Details (comments, headers, student's name and photo in the code, canvas, video, and report)

Some microProjects may not require any programming, others may not require writing a report. Instead, the students will be asked to provide a short video of a miniLecture given by them explaining a concept or solution.

Larger projects, which span n lectures will be graded out of $10n$ points. All large projects will count (i.e., none will be discarded).

Extensions: There will be **no extensions** of project deadlines and no late submission, but **your 3 worse microProject grades will be discarded for all students**. So, reserve these in anticipation of days when you need to travel or have a hardship, health issue, or computer problem.

Extra credit: There are several ways to earn extra credit.

In projects: implement cool extensions or variations or add comparative studies with alternatives. Please submit a separate .zip folder for the extra credit part or make it clear in the GUI how to run the extra credit and what it does. Also document (briefly) in the report that you have produced an additional implementation for extra credit, say what it does and how.

In project reports: Produce a very nice report with clear explanations, additional info/links, cool pictures. Think of writing it as a tutorial for other students.

During class: Participate actively by asking interesting questions and volunteering answers or on the white board corrections during class. Offer to give a short demo of an interesting extension of a project (email a TA and ask if they approve that your extension/solution) is interesting and correct. (This extra credit is left to the discretion of the instructor. Honestly, it is a highly subjective process and people who sit up front may have an advantage.)

At home: Be the first to find a major (previously unknown) mistake in the class notes, reading material, or base code provided and email to the instructor and the TAs: a brief explanation of the problem, its location, and a suggestion for a good fix. You may also discover online a better explanation or a different but interesting solution to a problem addressed in the lectures. If so, email the instructor and explain why you think it may benefit other students.

The total amount of extra credit per student will be capped to 20%. In general, extra credit per microProject and Project will also be capped to 20% of the points allocated for that project, although significantly extended extra credit may be given for outstanding discoveries, extensions, or studies.


What you need to do before the next lecture

1. Learn how to download/install Processing, edit, run (demo): <http://processing.org/tutorials/gettingstarted/>
2. Download/install/run two versions of Processing (1.5.1 and 2.1 or later) onto your laptop
3. Download the 01Pbase.zip file, unzip it, and make sure that you can run it
4. Find a clear, low-res picture of your face in .jpg format or use 01DliveVideoFromCam
5. Copy it into the data folder as pic.jpg
6. Do microProject 01 (see below) and submit on T-Square **before** the beginning of the next lecture
7. Study the course notes for this lecture and write down any questions you may want to ask in class
8. Register with TopHat and make sure that you know how to use it from your laptop (at GaTEch)
9. Download and read the assigned reading for this lecture (01RIntroduction.pdf)
10. If you find errors in the notes or reading, email the instructor (first report earns extra points)
11. Answer all the TopHat questions posted in these notes and in the assigned reading
12. Study the course notes (below) for **lecture 02**
13. Download 02Demos.zip and play with rotateAroundPoint and with swirl. Understand how the code works.
14. Download the 02Pbase.zip for microProject 02 onto your laptop and make sure that it runs

15. Download the 02PsolutionVideo.m4v and watch it to see what your project 02 should do
16. Read the deliverables for microProject 02 (below) and plan how you will implement them (write down questions)
17. Make sure that you have wireless access from your laptop in class
18. Bring your laptop to class for the next lecture (with charger if needed)
19. Practice answering sample questions provided about this and about the next lecture

Ideas for short projects

In each chapter, I will list ideas about short or microProjects, but the projects may vary from one year to another. Students are encouraged to read these ideas and implement additional projects to solidify their understanding of the material and to establish a portfolio of results.

 Write a brief description of 2 microProjects that you would like to implement and bring it to class on a piece of paper with your name on it

Relevant research questions

For each lecture, I will list a series of research questions. Some of these will be explored in class discussion or in projects by the graduate students in CS6491. Advanced undergraduate students are encouraged to read them and think about them. If time permits, we will investigate some of them in the CS3451 lectures.

Practice exercises

I will provide example questions and solutions. The students are encouraged to try and do all of them prior to looking at my solutions and to inform me if they detect mistakes in my solutions or if they have discovered a more elegant or simpler solution.

I will also provide additional exercises without solutions. Some of these will be used in exams.

Most of the TopHat questions will be multiple choice. I will offer some examples so that you can get used to them.

Here are the practice questions for lecture 01.

4.6 Practice questions with solutions

What is the core structure of a sketch?

- Global variables and classes
- setup() executed once at initialization
- draw() executed at each frame
- user actions procedures executed when the user acts on the keyboard or mouse

What does background(white); do?

- Erase the canvas before drawing the next frame

How to show on the canvas, next to the mouse, which key is pressed?

```
void draw() { ... if (keyPressed) {fill(black); text(key,mouseX-2,mouseY); }
```

How to write some text on the canvas?

- Use one of the scribe procedures provided in the utilities tab

How to set the color and width of the lines or shape borders to be drawn afterwards?

4.7 Additional practice questions, without solution

- How many quizzes will not be counted?
- When is the microProject due?
 - Use the pen(c,w); procedure provided in utilities

4.8 TopHat questions to answer online before the next class

Q01-1) Which of these implementations increments x when the mouse is pressed and dragged right:

1. `void mouseDragged() { ...x+=mouseX-pmouseX; ...}`
2. `void draw() { ... if(mousePressed) x+=mouseX-pmouseX; ...}`

A: Only 1, B: Only 2, C: Both, D: Neither

Q01-2) Which of these is suggested in the base code to make a movie showing how your sketch runs:

1. Press '!' repeatedly to snap pictures of the canvas
2. Press '~' to toggle the *filming* state variable which controls whether each frame is saved on file
3. Use your iPhone to take a video of the screen

A: Only 1, B: Only 2, C: only 3, D: None of these, E: All of these