# Programming for Media Arts

AME 230

School of Arts, Media and Engineering
Herberger Institute for Design and the Arts
Arizona State University
Spring 2019

Tuesday/Thursday, 10:30am-11:45am, Stauffer B125


## Loren Olson

loren.olson@asu.edu


## SPRING 2019 OFFICE HOURS

Location: Stauffer B260
Time: M/T/W/Th 2:00 - 3:00pm
During office hours you are welcome to walk in, no appointment is necessary.
If you need to see me, and you can't make my office hours, please send me an email to make an appointment at another time.

### Teaching Assistants


Jennifer Weiler
jjweiler@asu.edu
Location: Stauffer Gallery
Time: T/Th, 9:30 - 10:30am

Daisy Nolz
dnolz@asu.edu
Location: Stauffer Media Lab
Time: T/Th, 11:45am - 12:45pm


*This syllabus is subject to change. I will let you know when it does, always refer to this online version.*

## OVERVIEW

Computing is everywhere. However, not that long ago, computing was something only found in a large institutional machine room. Today it is ubiquitous, integrated into devices we use throughout the day. It's not surprising that there are now a wide range of jobs/professions that include programming work. As computing spreads, the range of programming tasks has also increased in variety.

Programming is more than a job skill. Programming can actually be a fun activity, a creative outlet, and a way to express abstract ideas in a tangible form. The goal of this class is to teach you how to program. A natural outcome of learning how to program, is learning a lot more about computing, and how a computer works. Designing programs also helps to develop a variety of valuable skills: critical reading, analytical thinking, creative synthesis, and attention to detail.

If computing has become ubiquitous, and programming so varied, what kind of programs should we teach you how to make? Aren't they all different? Some people want to add interaction to a web page, others want to make a script to automate a complex modeling task in a 3d animation system, others want to create a console video game, others want to make an app for iPhone. What should we choose for learning how to program?

We will use the Swift Programming Language. Swift is a relatively new programming language, developed at Apple, and introduced in 2014. Swift is a general purpose language, it is modern and progressive in its approach, with very ambitious goals for its scope. In December 2015 Apple published Swift under an open source license, and moved the project to an open development model. A good place for general Swift information is the official open source project website at https://swift.org/.

The goal will be to teach programming using Swift. Not to teach you "just Swift." Our goal is that you leave the class understanding how to program. Once you have learned how to program (using Swift), you will be ready to easily move on to other horizons if you so desire, whether that be for programs for web pages (javascript), 3d animation systems (python), games (C# or C++), or any other areas of interest no matter the platform.

**Course Objectives**

Understand the elements and structure of computer programs.
Understand and use common programming elements.
Analyze, break down, and solve a computing problem by creating a program.
Use, control, and synthesize common media types algorithmically.
Understand interactive, event driven programming environments.
By the end of this course, students will be able to:

- Create a complex program using the Swift programming language.
- Create a dynamic, interactive Swift program.
- Use various media types in an interactive Swift program including images, movies, and sound.
- Use object oriented programming techniques to develop flexible, extensible interactive Swift programs.

# TEXTBOOK

There is no required textbook to purchase. There will be readings I distribute that go with assignments. We will also use the free online book "The Swift Programming Language" as a reference. [https://docs.swift.org/swift-book/]

# ASU Spring 2019 Important Dates

| | |
|---|---|
| Session Dates and Deadlines | Session C: 15 weeks<br>(Jan 7 – April 26, Finals week is April 29) |
| Classes Begin | January 7, 2019 |
| Drop/Add Deadline | January 13, 2019 |
| Tuition and Fees 100% Refund Deadline | January 20, 2019 |
| Course Withdrawal Deadline | March 31, 2019 |
| Complete Session Withdrawal  Deadline | April 26, 2019 |
| Midterm Exam | February 28, 2019 (Thursday) |
| Spring Break | March 3 - 10, 2019 |
| Final Exam | April 30 (Tuesday), 9:50-11:40 |

For additional university deadlines and important dates for the spring 2019 term, please visit: students.asu.edu/academic-calendar.

# EVALUATION

### 60% Assignments

Assignments will involve writing a short program using Swift, assignments are turned in using Critviz. Always check Critviz for updated Assignment information and due dates.

### 30% Midterm and Final Exam

There will be a **midterm exam on Thursday February 28th**. The **final exam will be on Tuesday April 30th at 9:50am**.

### 10% Class Participation, Attendance

There will be some class quizzes, critique assignments, or other activities.

We will use an online system called Critviz to turn in assignments. Critviz can be found at http://critviz.com. I will show you how to use the system in class, and we will do an example project before a "real" assignment is due.

We score every assignment, exam, quiz or any activity on a scale of 0-100. Using that scale makes it very easy to always understand the performance result for any activity. It doesn't mean that everything has equal value! The assignment, exam, or quiz will be put into one of the above categories, and given a weight to determine how it contributes to the final course grade.

Not all exercises, assignments, or projects will be weighted equally, I may change the weights during the semester.

### Grading Scale

| | |
|---|---|
| A+ | 97-100% |
| A | 93-96% |
| A- | 90-92% |
| B+ | 87-89% |
| B | 83-86% |
| B- | 80-82% |
| C+ | 77-79% |
| C | 70-76% |
| D | 60-69% |
| E | 0-59% |

## LATE ASSIGNMENT POLICY

The concepts, techniques and ideas you learn in this class will be cumulative. They build on each other as the semester progresses, and you will use all of these things together throughout the semester as you learn more about programming. Therefore, it is important for your success to do all assignments in a timely fashion, in the order that they are given. The important concepts from one assignment will become foundations for subsequent coursework. If you miss an assignment, or for some reason you struggle with a particular assignment and do poorly - we want to make sure you don't miss those concepts and therefore struggle in future assignments. For this reason, we will accept late assignment submissions or resubmissions with a late penalty. Work submitted the week after due date is marked down 15 points. Each further week late means an additional 10 point penalty. After 4 weeks, late submissions will no longer be accepted.

# CLASSROOM CITIZENSHIP

Attend class, and pay attention. Participate in class discussion. Be curious. Be open to working hard and trying new things. Speak up. If you have a burning questions that you think is too simple to ask, it probably means several other people have the same question.

Be courteous to your fellow classmates, teaching assistant and teacher. Help us to create a positive and constructive learning environment that encourages everyone. Any disruptive behavior will be dealt with according to ASU policy. Please see the Student Services Manual for more details.

**Title IX**

Title IX is a federal law that provides that no person be excluded on the basis of sex from participation in, be denied benefits of, or be subjected to discrimination under any education program or activity. Both Title IX and university policy make clear that sexual violence and harassment based on sex is prohibited. An individual who believes they have been subjected to sexual violence or harassed on the basis of sex can seek support, including counseling and academic support, from the university. If you or someone you know has been harassed on the basis of sex or sexually assaulted, you can find information and resources at http://sexualviolenceprevention.asu.edu/faqs/students

As an employee of ASU, I am a mandated reporter and obligated to report instances of reported or suspected incidences of sexual harassment.

# ACADEMIC INTEGRITY

Is writing code like solving an equation in Calculus or is it more like writing an essay in English? If two students in English turn in essays that are nearly identical, it will be flagged as plagiarism. When two students have identical solutions in Calculus, not a second thought would be given - but everyone understands that copying answers from a neighbor's paper is cheating. In fact, I think that our projects should be more closely related to the English class essay, than the Calculus example. When you work on your project, please keep in mind the essay example - you must write your own unique essay.

What about code you found to help you? (Code at developer.apple.com, or stackoverflow, or github, etc) It can be ok to use other code in projects. It is critical that you acknowledge that code. You must cite where the code came from, and what code you have used. Also, I should add that you should never add code to a project that you don't understand. It is critical that you understand code that you add, it should not be

a "black box." Always cite code in comments, where you use it. Clearly mark what code is involved, and include a URL, and explanatory text. Failure to cite code in an assignment will result in an automatic 0 grade.

How much code, and what code is ok to use is also context dependent. Again, think about an English essay. You would not turn in an essay that consists of two sentences you wrote, then simply quote another writer for two entire pages of content. That wouldn't be your own essay. In this class, you cannot copy and paste many lines of code, change a few variable names, then submit the result as your own work. That would be a coding example of plagiarism.

What about study groups? A problem has arisen in the past when students work together, then turn in identical code. You should not be turning in identical code, since it becomes impossible for us to tell what is innocent coincidence due to a study group versus blatant unethical copying. I think study groups are very helpful, and do not want to discourage that community. Experience so far shows you can avoid this problem with a simple rule of thumb - always type all of your own code. Never copy a file from a friend, or allow a friend to copy one of your files.

**University policy regarding copyright**
Students must refrain from uploading to any course shell, discussion board, or website used by the course instructor or other course forum, material that is not the student's original work, unless the students first comply with all applicable copyright laws; faculty members reserve the right to delete materials on the grounds of suspected copyright infringement.

## Attendance and Participation

Students are expected to attend all classes. In the case of absence, please inform the instructor before the class if possible, and/or after the missed class. Classroom attendance and participation is 10% of the overall grade. Any student missing more than 2 classes without formal notes (Dr. Note etc) will fail the course.

## Religious Accommodations for Students

Students who need to be absent from class due to the observance of a religious holiday or participate in required religious functions must notify the instructor in writing as far in advance of the holiday/obligation as possible. Students will need to identify the specific holiday or obligatory function to the faculty member. Students will not be penalized for missing class due to religious obligations/holiday observance. The student should contact the class instructor to make arrangements for making up tests/assignments within a reasonable time.

### Special Accommodations

To request academic accommodations due to a disability, please contact the ASU Disability Resource Center (http://www.asu.edu/studentaffairs/ed/drc/# ; Phone: (480) 965-1234; TDD: (480) 965-9000). This is a very important step as accommodations may be difficult to make retroactively. If you have a letter from their office indicating that you have a disability which requires academic accommodations, in order to assure that you receive your accommodations in a timely manner, please present this documentation to me no later than the end of the first week of the semester so that your needs can be addressed effectively.

### Stauffer Media Lab

The Stauffer Media Lab - Stauffer B135 - is available for all students in this class, weekdays 8am to 8pm. All the computers in the lab have Xcode installed. There may be weekend hours this semester, check the lab to find current hours.

## REFERENCE

Apple Developer Web Site. If you want to get serious about writing software for Apple platforms, you should become familiar with this site.
https://developer.apple.com/

There are many other books about Xcode and Swift available, some of them are very good. There are enough that keeping a current list here isn't practical.

If you have an iPad, Apple has released an iPad app called Swift Playgrounds. See this (marketing) web page, and take a look - its free. The app includes links to educational material. There is a Learn to Code iTunes U course that uses the Playgrounds app.

## OUTLINE

1. Regarding programming.
    1. What is programming?
    2. Where do you find programs? Who programs?
    3. Viewing Assignment: SXSW talk, Program or Be Programmed (optional book link if you want to know more), Douglas Rushkoff (talk at SXSW 2010)
2. Getting Started in Xcode
    1. What is Xcode?
    2. Creating a Swift Playground.
    3. Instructions, data and writing code.
3. Simple drawing with Swift and Tin.

4. The Tin Framework.
    1. A coordinate system for drawing.
    2. Basic shapes.
    3. Color for drawing shapes.
5. Variables
    1. Constants and Variables.
    2. Integers and Floats.
    3. Types.
    4. Booleans.
    5. Optionals.
    6. Names.
6. Interactive programs.
    1. Project structure.
    2. Event-loop program flow.
    3. setup()
    4. update()
    5. Frame rate.
    6. Mouse position.
    7. Mouse clicks and key presses.
7. Operations
    1. Expressions and operators.
    2. Assignment.
    3. Math operators.
    4. Comparison operators.
    5. Logical operators.
    6. Boolean expressions
    7. If conditional statements
    8. Else
8. Strings
    1. String literals.
    2. Value types.
    3. Characters.
    4. Unicode.
    5. Working with Strings
9. Xcode.
    1. Elements of Xcode.
    2. Xcode projects.
    3. Creating a macOS app project.
    4. Development flow in Xcode.
10. Control Flow statements.
    1. While loop.
    2. For-in loop.

    3. Switch.
    4. Control transfer statements.
11. Collections.
    1. Arrays.
    2. Dictionaries.
12. Functions.
    1. Organizing code.
    2. Defining and calling functions.
    3. Parameters and return values.
    4. Argument labels and parameter names.
    5. Modularity.
    6. Values from caller to function.
13. Enumerations
    1. Defining an enumeration.
    2. Matching enumeration values in statements.
14. Structures and Classes.
    1. A short introduction to object oriented programming.
    2. Elements of a Class.
    3. Class as a template.
    4. Comparing structures vs. classes.
15. Properties
    1. Stored properties.
    2. Computed properties.
    3. Global and local variables.
    4. Type properties.
16. Methods
    1. Instance methods.
    2. Type methods.
17. Initialization
    1. Setting initial values for properties.
    2. Customizing initialization.
18. Algorithms.
    1. Process of programming.
    2. Recipes.
    3. Divide and conquer.
19. Grab bag of technique.
    1. Modulus and a loop trick.
    2. Probability.
    3. Perlin noise.
    4. Angles, radians and polar coordinates.
    5. Recursion.
20. Transformations.

    1. Translate, Rotate, Scale.
    2. Graphics state, pushState, popState
    3. Creating a hierarchy of movement.
21. Images.
    1. A simple class for images.
    2. Animating an image as a drawing primitive.
    3. Getting pixel data from an image
    4. Pixels and image processing.
22. Debugging
    23. Writing programs is hard, some thoughts about fixing code.
24. Data visualization.
    25. Loading a text file from disk.
    26. Parsing a CSV file.
    27. Creating a time-series graph.
    28. Saving Strings (data) to a file on disk.
29. Sound.
    30. AVFoundation
    31. AVAudioPlayer
32. Where to go next.