ARIZONA STATE UNIVERSITY

GENERAL STUDIES COURSE PROPOSAL COVER FORM

Course information:

*Copy and paste **current** course information from Class Search/Course Catalog.*

College/School: Herberger Institute for Design and the Arts | Department: **Arts, Media + Engineering**

Prefix: **ame** | Number: **230** | Title: Programming for Media Arts | Units: 3

Is this a cross-listed course? No | If yes, please identify course(s)

Is this a shared course? No | If so, list all academic units offering this course

*Note- For courses that are crosslisted and/or shared, a letter of support from the chair/director of **each** department that offers the course is required for **each** designation requested. By submitting this letter of support, the chair/director agrees to ensure that all faculty teaching the course are aware of the General Studies designation(s) and will teach the course in a manner that meets the criteria for each approved designation.*

Is this a permanent numbered course with topics? No

If yes, all topics under this permanent numbered course must be taught in a manner that meets the criteria for the approved designation(s). It is the responsibility of the chair/director to ensure that all faculty teaching the course are aware of the General Studies designation(s) and adhere to the above guidelines.

Chair/Director Initials _____ (Required)

Course description: **Introductory-level course in practical aspects of programming systems for media arts. Explores techniques in programming in the context of images, sounds, interaction, networking and data visualization. Uses the processing programming language, an accessible yet powerful environment for learning how to program. Students create images, animations and interactive programs, and learn how to use images, movies, sound files, microphones and cameras in their programs.**

Requested designation: Mathematical Studies-CS **Mandatory Review:** Yes

*Note- a **separate** proposal is required for each designation.*

Eligibility:

Permanent numbered courses must have completed the university's review and approval process.
For the rules governing approval of omnibus courses, contact Phyllis.Lucie@asu.edu.

Submission deadlines dates are as follow:

For Fall 2016 Effective Date: October 1, 2015 For Spring 2017 Effective Date: March 10, 2016

Area(s) proposed course will serve:

A single course may be proposed for more than one core or awareness area. A course may satisfy a core area requirement and more than one awareness area requirements concurrently, but may not satisfy requirements in two core areas simultaneously, even if approved for those areas. With departmental consent, an approved General Studies course may be counted toward both the General Studies requirement and the major program of study.

Checklists for general studies designations:

Complete and attach the appropriate checklist

- Literacy and Critical Inquiry core courses (L)
- Mathematics core courses (MA)
- Computer/statistics/quantitative applications core courses (CS)
- Humanities, Arts and Design core courses (HU)
- Social-Behavioral Sciences core courses (SB)
- Natural Sciences core courses (SQ/SG)
- Cultural Diversity in the United States courses (C)
- Global Awareness courses (G)
- Historical Awareness courses (H)

A complete proposal should include:

- ☒ Signed course proposal cover form
- ☒ Criteria checklist for General Studies designation(s) being requested
- ☒ Course catalog description
- ☒ Sample syllabus for the course
- ☒ Copy of table of contents from the textbook and list of required readings/books

It is respectfully requested that proposals are submitted electronically with all files compiled into one PDF.

Contact information:

Name: Althea Pergakis | E-mail: althea.pergakis@asu.edu | Phone: 480.965.1010

Department Chair/Director approval: *(Required)*

Chair/Director name (Typed): Xin Wei Sha Date: _____

ARIZONA STATE UNIVERSITY

Chair/Director (Signature): 

Arizona State University Criteria Checklist for

MATHEMATICAL STUDIES [CS]

Rationale and Objectives

The **Mathematical Studies** requirement is intended to ensure that students have skill in basic mathematics, can use mathematical analysis in their chosen fields, and can understand how computers can make mathematical analysis more powerful and efficient. The **Mathematical Studies** requirement is completed by satisfying both the **Mathematics [MA]** requirement and the **Computer/Statistics/Quantitative Applications [CS]** requirement explained below.

The **Mathematics [MA]** requirement, which ensures the acquisition of essential skill in basic mathematics, requires the student to complete a course in College Mathematics, College Algebra, or Pre-calculus; or demonstrate a higher level of skill by completing a mathematics course for which a course in the above three categories is a prerequisite.

The **Computer/Statistics/Quantitative Applications [CS]** requirement, which ensures skill in real world problem solving and analysis, requires the student to complete a course that uses some combination of computers, statistics, and/or mathematics.* Computer usage is encouraged but not required in statistics and quantitative applications courses. At a minimum, such courses should include multiple demonstrations of how computers can be used to perform the analyses more efficiently.

*CS does *not* stand for computer science in this context; the "S" stands for statistics. Courses in computer science must meet the criteria stated for CS courses.

Revised April 2014

Proposer: Please complete the following section and attach appropriate documentation.

ASU--[CS] CRITERIA			
A COMPUTER/STATISTICS/QUANTITATIVE APPLICATIONS [CS] COURSE MUST SATISFY ONE OF THE FOLLOWING CRITERIA: 1, 2, OR 3			
YES	**NO**		**Identify Documentation Submitted**
		1. Computer applications*: courses must satisfy both **a** and **b**:	
☒	☐	**a.** Course involves the use of computer programming languages or software programs for quantitative analysis, algorithmic design, modeling, simulation, animation, or statistics.	syllabus
		b. Course requires students to analyze and implement procedures that are applicable to at least one of the following problem domains **(check those applicable)**:	
☐	☒	i. Spreadsheet analysis, systems analysis and design, and decision support systems.	
☒	☐	ii. Graphic/artistic design using computers.	syllabus
☐	☒	iii. Music design using computer software.	
☐	☒	iv. Modeling, making extensive use of computer simulation.	
☐	☒	v. Statistics studies stressing the use of computer software.	
☒	☐	vi. Algorithmic design and computational thinking.	syllabus

*The **computer applications** requirement **cannot** be satisfied by a course, the content of which is restricted primarily to word processing or report preparation skills, the study of the social impact of computers, or methodologies to select software packages for specific applications. Courses that emphasize the use of a computer software package are acceptable only if students are required to understand, at an appropriate level, the theoretical principles embodied in the operation of the software and are required to construct, test, and implement procedures that use the software to accomplish tasks in the applicable problem domains. Courses that involve the learning of a computer programming language are acceptable only if they also include a substantial introduction to applications to one of the listed problem domains.

YES	NO		Identify Documentation Submitted
		2. Statistical applications: courses must satisfy **a**, **b**, and **c**.	
☐	☒	**a.** Course has a minimum mathematical prerequisite of College Mathematics, College Algebra, or Pre-calculus, or a course already approved as satisfying the MA requirement.	
		b. The course must be focused principally on developing knowledge in statistical inference and include coverage of all of the following:	
☐	☒	i. Design of a statistical study.	
☐	☒	ii. Summarization and interpretation of data.	
☐	☒	iii. Methods of sampling.	
☐	☒	iv. Standard probability models.	
☐	☒	v. Statistical estimation	
☐	☒	vi. Hypothesis testing.	
☐	☒	vii. Regression or correlation analysis.	
☐	☒	**c.** The course must include multiple demonstrations of how computers can be used to perform statistical analysis more efficiently, if use of computers to carry out the analysis is not required.	

YES	NO		Identify Documentation Submitted
		3. Quantitative applications: courses must satisfy **a, b, and c:**.	
☐	☒	**a.** Course has a minimum mathematical prerequisite of College Mathematics, College Algebra, or Pre-calculus, or a course already approved as satisfying the MA requirement.	
		b. The course must be focused principally on the use of mathematical models in quantitative analysis and decision making. Examples of such models are:	
☐	☒	i. Linear programming.	
☐	☒	ii. Goal programming.	
☐	☒	iii. Integer programming.	
☐	☒	iv. Inventory models.	
☐	☒	v. Decision theory.	
☐	☒	vi. Simulation and Monte Carlo methods.	
☐	☒	vii. Other (explanation must be attached).	
☐	☒	**c.** The course must include multiple demonstrations of how computers can be used to perform the above applications more efficiently, if use of computers is not required by students.	

Course Prefix	Number	Title	General Studies Designation
AME	230	Programming for Media Arts	

Explain in detail which student activities correspond to the specific designation criteria.
Please use the following organizer to explain how the criteria are being met.

Criteria (from checksheet)	How course meets spirit (contextualize specific examples in next column)	Please provide detailed evidence of how course meets criteria (i.e., where in syllabus)
1A	Teaches the programming lanuage Processing for animation, algorithmic design and interactive media works	See highlighted sections in syllabus under "overview" and "course objectives"
1Bii	Uses programming to create media art via interactitivy, animation and basic computational drawing techniques	see highlighted section in "course objectives" and the course outline
1Bvi	Uses Processing to design algorithmically and to create foundations of computational thinking	see course outline

course catalog & class search

Term: Spring 2016

Search: Course catalc

Subject: AME Num: 230

Level:

Gen Studies:

Keywords:

Search

Clear

Advanced Search

AME 230 - Programming for the Media Arts

Spring 2016

Course description: Introductory-level course in practical aspects of programming systems for media arts. Explores techniques in programming in the context of images, sounds, interaction, networking and data visualization. Uses the processing programming language, an accessible yet powerful environment for learning how to program. Students create images, animations and interactive programs, and learn how to use images, movies, sound files, microphones and cameras in their programs.

Enrollment requirements: Credit is allowed for only AME 230 or AME 294 (Programming for Media Arts)

Units: 3

Repeatable for credit: No

General Studies: No

Offered by: Herberger Institute for Design and the Arts

Class meeting details

Class #:	Days:	Start:	End:	Location:	Instructor:	Seats open:	
28193	M W	10:30 AM	11:45 AM	Tempe - STAUFB125	Olson	0 of 100	Add

Additional class details

Component: Integrated Lecture/Lab

Session: Session C

Dates: 1/11/2016 - 4/29/2016

Instruction Mode: In-Person

Fees: None

Books:

View books for this class

Back

Feedback

AME 230 - Programming for Media Arts

School of Arts, Media and Engineering
Herberger Institute for Design and the Arts
Spring 2016. Monday, Wednesday 10:30 am - 11:45 am, Stauffer B125

Instructor: Loren Olson, loren.olson@asu.edu
Office: Stauffer B260
Office Hours: MW 3-4pm, TTh 1-2pm, or by appointment

Teaching Assistant: Courtney Brown, courtney.d.brown@asu.edu
Office: DC Lab
Office Hours: MW 12-1p

Teaching Assistant: Prashanth Seshasayee, spseshas@asu.edu
Office: Stauffer B204
Office Hours: MW 9-10am

This syllabus is subject to change.

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 Xbox console game, others want to make an app for iPhone. What should we choose for learning how to program?

We will use Processing. Processing is an open source project, that is freely available for Windows, Mac, and Linux. It is very accessible, so it is easy to get started with, and has relatively few distractions and hurdles for new programmers. Yet Processing is still a powerful programming environment that can create complex, novel and interesting programs.

The goal will be to teach programming using Processing. Not to teach you "just Processing." Our goal is that you leave the class understanding how to program. Once you have learned how to program (using Processing), 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), Xbox (C# or C++), iPhone or iPad (Swift, Objective-C) or any other 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 Processing programming language.
- Create a dynamic, interactive Processing sketch.
- Use various media types in an interactive Processing sketch including images, movies, and sound.
- Use object oriented programming techniques to develop flexible, extensible interactive Processing sketches.

Spring 2016 Important Dates

January 11	First Day of Class
January 17	Drop/Add Deadline
January 18	Martin Luther King, Jr. Holiday Observed - University Closed
January 24	Tuition & Fees 100% Refund Deadline
March 6-13	Spring Break
April 3	Course Withdrawal Deadline
April 29	Complete Session Withdrawal Deadline
April 29	Last Day of Classes
May 2, (Monday) 9:50-11:40am	Final Exam

Evaluation

60% Assignments
Assignments will involve writing a short program using Processing, assignments are turned in using Critviz. Always check Critviz for updated Assignment information and due dates.

20% Midterm and Final Exam
There will be a midterm exam on Wednesday March 2nd. The final exam will be on Monday May 2nd, at 9:50am.

10% Tests
There will be some in class tests, quizzes or activities.

10% Class Participation, Attendance
There will be a class roster to sign for attendance, it is your responsibility to sign the roster every class. If you don't sign the roster, you don't get attendance credit.

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.

Grading Scale

A+	97-100%
A	93-96%
A-	90-92%
B+	87-89%
B	83-86%
B-	80-82%
C+	77-79%
C	70-78%
D	60-69%
E	0-59%

A curve may be applied to raw scores at the instructor's discretion.

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.

Academic Integrity. All necessary and appropriate sanctions will be issued to all parties involved with plagiarizing any and all course work. Plagiarism and any other form of academic dishonesty that is in violation with the Student Code of Conduct will not be tolerated. For more information, please see the ASU Student Academic Integrity Policy: Student Academic Integrity Policy

Regarding plagiarism, code sharing and code reuse.
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 processing.org, 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 on a related note, I should add that you should never add code to a project that you don't understand. Its 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.

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.

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. Late assignments cannot be turned in via Critviz, you will need to email them to the Teaching Assistants.

Outline

1. Regarding programming.
    1.1. What is programming?
    1.2. Where do you find programs? Who programs?
    1.3. Viewing Assignment: SXSW talk, Program or Be Programmed (optional book link if you want to know more), Douglas Rushkoff (talk at SXSW 2010)
    1.4. Reading Assignment: Getting Started, Casey Reas and Ben Fry. Processing.org.
2. Simple drawing with Processing.
    2.1. Processing coordinate system.
    2.2. Making a window.
    2.3. Basic shapes.
    2.4. Color for drawing shapes.
    2.5. Reading Assignment: Coordinate System and Shapes, Daniel Shiffman. Processing.org.
    2.6. Reading Assignment: Color, Daniel Shiffman. Processing.org.
    2.7. API: size, background, rect, ellipse, line, point, ellipseMode, rectMode, stroke, fill, noStroke, noFill.
3. Introducing the Processing environment.
    3.1. The open source project.
    3.2. The processing app.
    3.3. Processing sketch on disk.
    3.4. Documentation.
4. Interactive Processing sketches.
    4.1. The flow of a sketch.
    4.2. Setup and Draw.
    4.3. Mouse location.
    4.4. Framerate.
    4.5. Mouse clicks and Key presses.
    4.6. Reading Assignment: Setup and Draw.
5. Variables.
    5.1. Variable declaration, assignment.
    5.2. Types.
    5.3. Names.
    5.4. System variables.
    5.5. Simple expressions and math.
    5.6. Reading Assignment: Variables.
    5.7. Reading Assignment: Integers and Floats.
6. Conditional statements.
    6.1. Boolean expressions.
    6.2. Relational operators.
    6.3. If, else, else if.
    6.4. Logical operators.

- 6.5. Boolean variables.
- 6.6. Reading Assignment: True and False.
- 6.7. Reading Assignment: Conditionals 1.
- 6.8. Reading Assignment: Conditionals 2.
- 6.9. Reading Assignment: Logical Operators.

7. Loops.
   - 7.1. Iteration.
   - 7.2. While loop.
   - 7.3. Exit conditions.
   - 7.4. For loop.
   - 7.5. Variable scope.
   - 7.6. Reading Assignment: Iteration.
   - 7.7. Reading Assignment: Variable scope.
8. Functions.
   - 8.1. Organizing code.
   - 8.2. Creating a function.
   - 8.3. Modularity.
   - 8.4. Function parameters.
     - 8.4.1. Pass by value
   - 8.5. Return values.
   - 8.6. Flow in a program with functions.
   - 8.7. Reading Assignment: Functions.
9. Objects.
   - 9.1. Intro to Object Oriented Programming.
   - 9.2. Data and instructions.
   - 9.3. Class as a template.
   - 9.4. Using objects in Processing.
   - 9.5. Creating a class.
     - 9.5.1. Instance variables.
     - 9.5.2. Constructor.
     - 9.5.3. Methods.
   - 9.6. Reading Assignment: Object Oriented Programming, Daniel Shiffman.
10. Arrays.
    - 10.1. Declaring an array.
    - 10.2. Initializing an array.
    - 10.3. Array operations.
    - 10.4. Reading Assignment: Array.
    - 10.5. Reading Assignment: Array objects.
11. Algorithms.
    - 11.1. Process of programming.
    - 11.2. Recipes.
    - 11.3. Divide and conquer.
    - 11.4. Reading Assignment: Anatomy of a Program, J David Eisenberg. Processing.org.
12. Text.
    - 12.1. String class.
    - 12.2. Rendering text.
      - 12.2.1. createFont, textFont, text
    - 12.3. Reading: Strings and Drawing Text, Daniel Shiffman. Processing.org.
13. Grab bag of technique.
    - 13.1. Modulus and a loop trick.

13.2. Probability.
13.3. Perlin noise.
13.4. Angles, radians and polar coordinates.
13.5. Recursion.
13.6. Two dimensional arrays.

14. Transformations.
14.1. Translate, Rotate, Scale.
14.2. Transformation Matrix and pushMatrix/popMatrix.
14.3. Reading Assignment: 2d Transformations, J David Eisenberg. Processing.org.

15. Images.
15.1. PImage class.
15.2. Animating an image.
15.3. Pixel data and image processing.
15.4. Reading Assignment: Images and Pixels, Daniel Shiffman. Processing.org.

16. More about text.
16.1. String methods indexOf, substring.
16.2. split, join.

17. Data visualization.
17.1. loadStrings.
17.2. Parsing a CSV file.
17.3. Creating a data viz graph.
17.4. saveStrings.
17.5. Reading Assignment: Data, Daniel Shiffman. Processing.org.

18. Sound.
18.1. Minim audio library
18.1.1. AudioSample, loadSample, trigger.
18.1.2. AudioPlayer, loadFile, play, pause.
18.1.3. AudioInput.
18.1.4. AudioRecorder, createRecorder, beginRecord, endRecord, save.

19. Where to go next.

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 Processing installed. There may be weekend hours this semester, check the lab to find current hours.

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.

Reference

This class does not have a required textbook. There will be required readings, usually via the web, at various times during the semester. If you would like to seek out more information about Programming for Media Arts, and about Processing, here are some ideas.

Books about Processing. There is a pretty good Processing community of users, and a bunch of books have been published. These are two that I have read, and thought were good.

Learning Processing, Daniel Shiffman.
Visualizing Data, Ben Fry.

Related Web Sites.

The official Processing web site. http://processing.org. This is where you can download processing, find the reference documentation and lots of learning resources.

http://www.openprocessing.org. A community site, and there are lots of example Processing sketches submitted by many authors. A great way to learn is to look at a project that you like.

http://benfry.com. Ben Fry is an artist and designer, and the co-inventor of Processing.

http://reas.com. Casey Reas is an artist and professor at UCLA, and the co-inventor of Processing.

http://www.shiffman.net. Daniel Shiffman is a professor at ITP, and author. He has some interesting Processing related work on his website.

http://www.flong.com. Golan Levin is an artist and professor at Carnegie Mellon University.

http://processingjs.org. Processing.js is a port of Processing to javascript, that runs natively in a web browser.