Digital Arts & New Media 220: Introduction to Programming in the Arts
University of California, Santa Cruz

instructor: Warren Sack <wsack@ucsc.edu>
time: winter term 2015, Wednesdays from 9:00am-12:00pm
place: Digital Arts Research Center, Room 206
office hours: Mondays & Fridays, 10:00am-11:00am (Global Village Café at McHenry Library; fifteen minute appointments reservable on Doodle; here is the link: http://bit.ly/1yvr4rI)

Learning objectives

This course is designed to develop necessary digital art and new media knowledge for students to understand the state of the field. We will be concentrating on technical and aesthetic knowledge. Students of this course are also expected to develop the ability to define, plan, and execute individual and collaborative digital art and new media projects. The main goals of the course are (1) to give students a broad enough knowledge of the various approaches to programming so that they will be able to select the right approach for a specific project; and, (2) gain a deep understanding of one programming language sufficient to execute a relatively large project. Students will be assessed according to how well they

- work individually and collaboratively to define, plan, debug, and implement computer programs;
- read and understand code and pseudo code;
- critique, speak and write about software;
- understand how to learn a new programming language;
- demonstrate a deep knowledge of one programming language;
- explain how code and software have influenced contemporary art (and vice versa).

Approach

"Conceptual Art, … a significant precursor to New Media art, focused more on ideas than on objects. New Media art is often conceptual in nature. … Much as Lawrence Weiner's 'Indefinite Material Descriptions' (e.g., One Quart of Exterior Industrial Enamel Thrown on a Brick Wall*, 1964) don't need to be realized to exist as art works, [a New Media art work] doesn't need to be seen (or completed) to be understood."*
Mark Tribe and Reena Jana, *New Media Art*, 2006

"I will refer to the kind of art in which I am involved as conceptual art. In conceptual art the idea or concept is the most important aspect of the work. … The idea becomes a machine that makes the art."
Sol Lewitt, *Paragraphs on Conceptual Art*, 1967

This course takes the approach that digital art is at the intersection of critical studies and computer science. In order to compose and analyze what we see and hear – what we experience – in digital art, we need the languages of art and critical theory. In order to talk about inner workings of digital art we need the languages of computer science or, more specifically, the language of code. The outstanding questions of digital art are how can we translate the language of art into the language of code and vice versa? We will pay attention to how the work of artwork is described and how the machines of code are articulated. Consequently, our discussions will recurrently address what I call "work languages" and "machine languages."

Most new media artworks are constructed collaboratively using techniques of "montage," "remix," "mashup," or, what the curator Nicolas Bourriaud has called "postproduction." In other words, when a new artwork is made, it is frequently made from pieces of other existing artworks, or, more generally, found things. Marcel Duchamp called this a "readymade" when he made art out of things he bought from local department store, like a bottle rack. In the twentieth century it was so-called "conceptual artists" who developed these techniques, this approach of "postproduction." Consequently, in this course, we explore the connections between earlier conceptual art and contemporary digital media works.

This course is based on the assumption that computer programming is a new medium of expression. This position has been eloquently stated on a number of occasions by Turing Award winning computer scientist Alan Kay. New media theorist Lev Manovich reiterates Kay's position in his book *Software Takes Command* (2013).

Because software is now so prevalent, programming literacy has been recognized as an essential part of contemporary education. In his essay, "Procedural Literacy: Educating the New Media Practitioner," Michael Mateas describes a course he taught at Georgia Tech that had very similar aims to this course for a program very much like DANM. He argues that *"…procedural literacy, of which programming is a part, is critically important for new media scholars and practitioners...*" Over the past fifty years, prominent computer scientists and programming language designers – including Alan Perlis, Alan Kay, Seymour Papert – have argued that programming should be integral to everyone's education. But, artists in this language of new media, need to strive not just for literacy but also for fluency.

Ever since the "readymades" of artist Marcel Duchamp (or some, especially philosopher and art critic Arthur Danto, would say since the "Brillo Boxes" of Andy Warhol) it has been difficult to tell the difference between contemporary art and anything else. Duchamp said that art is no longer "retinal"; i.e., you cannot tell if something is art, or not, just by looking at it. For instance, one might assume that there is an obvious difference between "programming for the arts" and "programming for the sciences," but this has never been a clear distinction and was further blurred when Linus Torvalds was awarded the Golden Nica at Ars Electronica in 1999, not for an artwork exhibited in galleries and

museums, but for his work on the Linux operating system (http://archive.aec.at/#34560).

Hyperbolically, this aesthetic can be described like this: "*…computer scientists who invented these technologies – J.C. Licklider, Douglas Engelbart, Ivan Sutherland, Ted Nelson, Seymour Papert, Tim Berners-Lee, and others – are the important artists of our time – maybe the only artists who are truly important and who will be remembered from this historical period*." Lev Manovich, "Introduction: New Media from Borges to HTML," in *The New Media Reader*," edited by Noah Wardrip-Fruin and Nick Montfort (MIT Press, 2003): 13-25. Yikes! Manovich's pronouncement is grandiose yet it does push one to think that perhaps there is an art practice within the productions of computer science.

One artist who personifies this possibility in an inspiring and quietly self-reflective manner is Casey Reas, Professor of Design | Media Arts at UCLA and co-designer (with Ben Frye) of the programming language for artists and designers, Processing. We will closely study the work of Casey Reas and a few other artist/programmers who illustrate different ways in which art and code are eloquently entangled.

Requirements

Meetings

We will meet every Wednesday for eleven weeks from 9:00am-12:00pm in DARC 206. Please come to class meetings and come on time. If you are unable to come to a meeting or need to be late, please contact me date of your absence. Three unexcused absences will result in a failing grade for the course.

Assignments

There will be weekly coding assignments.

Final project: You will be asked to devote the final few weeks of the term to developing a final project. I will ask you for a proposal before the end of the term. At the last meeting of the course, during finals week, you will be asked to present your final project. I will also be encouraging you to submit your final project to a competition or festival.

Grades

1. 10%: Class attendance and discussion: You will be expected to attend all class meetings and actively participate in discussions and critiques.
2. 20%: Final project proposal and final project: During the last few weeks of the term, you will propose, develop, and document a final project that will be reviewed at the end of the term.
3. 70%: Small projects: There will be seven small projects assigned to you over the course of the term. Each will be due the week following the assignment.

Resources

I will add links to texts, code, and videos on the course website over the term. Some of these can already be found in the schedule.

SCHEDULE

January 7

Agenda
1: Introduction to the course
2: Hour of Code: Angry Birds; see http://studio.code.org/hoc/1
3: Hour of Code: Scratch; see http://scratch.mit.edu/hoc2014/

Assignment 1 (due January 14)
Find two or more projects on the Scratch website; combine them into a game of your own design.

January 14

Read
Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai, "Scratch: Programming for All," Communications of the ACM, Vol. 52, No. 11, November 2009
http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf

The LEAD Project, Chapters 1 & 2, *Super Scratch Programming Adventure! (Covers Version 2), 2nd Edition* (No Starch Press, 2013)
http://scratched.media.mit.edu/sites/default/files/scratch2_samplechapters.pdf

View
Mitchel Resnick on Scratch
http://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code.html

Lifelong Kindergarten page: http://llk.media.mit.edu

Scratch tutorials (optional)
http://scratch.mit.edu/help/videos/

Do
1: Hour of Code: Scratch; see http://scratch.mit.edu/hoc2014/

2: Familiarize yourself with the Scratch language and website; explore the archive of projects and find the “Tips” menu
3: Experiment with the code in the *Super Scratch* book

Agenda
1: Discuss the design of the Scratch language.
2: Review of the games produced.
3: Hour of Code: Processing; see http://hello.processing.org/
4: Browse Christiane Paul’s CODeDOC commissions.

Assignment 2 (due January 21)
Connect Three Points:
http://artport.whitney.org/commissions/codedoc/index.shtml

January 21

Read
Reas, Casey and Ben Fry, *Getting Started With Processing* (O’Reilly, 2010)

Sol LeWitt, “Paragraphs on Conceptual Art,” 1967
http://www.ddooss.org/articulos/idiomas/Sol_Lewitt.htm

Lawrence Weiner, “Declaration of Intent,” 1968; and, “Indefinite Material Descriptions,” 1968
http://iaaa.nl/cursusAA&AI/concept/weiner.html

View
Processing Tutorials
http://processing.org/tutorials/

Videos of Casey Reas’ work
(e.g., http://www.youtube.com/watch?v=_8DMEHxOLQE)

{Software} Structures by Casey Reas with Robert Hodgin, William Ngan, Jared Tarbell
http://artport.whitney.org/commissions/softwarestructures/

Agenda
1: Discuss the Processing language; compare it with Scratch.
2: Review responses to assignment 2, CODeDOC.
3: What is conceptual art?
4: Translating from Lewitt’s Structures to Processing

Assignment 3 (due January 28)
Pick one of Lewitt’s Wall Drawings and write it in Processing.

January 28

Read
Reas, Casey and Ben Fry, “Image 5: Image Processing” and “Extension 3: Vision by Golan Levin,” *Processing: A Programming Handbook for Visual Designers and Artists* (MIT Press, 207), 355-364 & 547-561.

Flusser, Vilém “The Technical Image,” *Towards a Philosophy of Photography*, 1983

Agenda
1: Review responses to assignment 3, translating Lewitt into Processing.
2: What is a technical image?
3: Discuss machine vision.

Assignment 4 (due February 4)
Play with machine vision

February 4

Read
Haverbeke, Marijn. *Eloquent JavaScript*, Second Edition
http://eloquentjavascript.net/Eloquent_JavaScript.pdf

or, if you are an expert programmer, read

Crockford, Douglas. *Javascript the Good Parts* (Sebastopol, CA: O'Reilly, 2008). [available digitally through the UCSC subscription to the Safari (O’Reilly) Databade; you can search Melvyl to get a direct link to the ebook]

Drucker, Johanna. “Experimental Typography as a Modern Art Practice,” *The Visible Word: Experimental Typography and Modern Art, 1909-1923* (University of Chicago Press, 1994), 91-140.

Cayley, John. “Zero-Count Stitching”
http://programmatology.shadoof.net/index.php?p=contents/zeroCounting.html

View
Young-Hae Chang Heavy Industries, “Rain on the Sea”
http://www.yhchang.com/RAIN_ON_THE_SEA.html

Do
Hour of Code: JavaScript
https://www.khanacademy.org/computing/hour-of-code/hour-of-code-tutorial/v/welcome-hour-of-code

1: Review responses to assignment 4, machine vision
2: Discuss the design of the JavaScript language and compare it to Processing
3: What is the visible word?

Assignment 5 (due February 11)
Write a piece of dynamic, concrete poetry.

February 11

Read
Weizenbaum, Joseph "ELIZA – A Computer Program For the Study of Natural Language Communication Between Man and Machine," *Communications of the ACM*, Volume 9, Issue 1 (January 1966): p 36-45.

Wardrip-Fruin, Noah. "Chapter 1: Introduction" and "Chapter 2: The Eliza Effect," *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*. Cambridge, Mass: MIT Press, 2009.

Mozilla Developer Network, "Regular Expressions in JavaScript"
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Landsteiner, Norbert. "ELIZA in JavaScript"
http://www.masswerk.at/elizabot/

Agenda
1: Review responses to assignment 5, concrete poetry
2: Discuss ELIZA
3: Discuss the "ELIZA Effect"

Assignment 6 (due February 18)
Write a new script for ELIZA

February 18

Read
Meehan, James. "Chapter 9: Tale-Spin" and "Chapter 10: Micro Tale-Spin," in Roger Schank and Christopher Riesbeck eds., *Inside Computer Understanding: Five Programs Plus Miniatures* (Lawrence Erlbaum, 1981).

Wardrip-Fruin, Noah. "Chapter 5: The Tale-Spin Effect," *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*. Cambridge, Mass: MIT Press, 2009.

Sack, Warren. "HTN Planner in JavaScript"
http://danm.ucsc.edu/~wsack/SoftwareArts/Code/Plan/

Sack, Warren. “Spinner Story Generator in JavaScript”
http://danm.ucsc.edu/~wsack/SoftwareArts/Code/Narrative/

Agenda
1: Review responses to assignment 6, a new script for ELIZA
2: Discuss the Tale-Spin Effect
3: Review the design and implementation of LLPL, SHOP, Spinner, and Mumbler

Assignment 7 (due February 25)
In the file initial.js, there are six “stories” defined with combinations of initial facts and initial tasks. Define a new story for Spinner. Consult the exercises at the end of Meehan’s “Chapter 10: Micro Tale-Spin” for some suggestions about what you might do.

February 25

Agenda
1: Review responses to assignment 7, a new story for Spinner
2: Discuss final project proposals and final projects

Final Project Proposal and Final Project (due Friday, March 20)
Create a final project proposal and implement some portion of it for the final project.

March 4

Agenda
Work on final project proposals and final project

March 11

Agenda
Work on final project proposals and final project

March 20 (note that this is a Friday)

Agenda
Final presentations