SM3809: SOFTWARE ART STUDIO

Effective Term
Semester A 2025/26

Part I Course Overview

Course Title
Software Art Studio

Subject Code
SM - School of Creative Media
Course Number
3809

Academic Unit
School of Creative Media (SM)

College/School
School of Creative Media (SM)

Course Duration
One Semester

Credit Units
6

Level
B1, B2, B3, B4 - Bachelor's Degree

Medium of Instruction
English

Medium of Assessment
English

Prerequisites
SM2715 Creative Coding

Precursors
Nil

Equivalent Courses
Nil

Exclusive Courses
Nil

Part II Course Details

Abstract
A project-oriented course for creative coders wishing to take their computational art practice to the next level. Lectures will address aesthetic, conceptual, and technical concerns arising at the intersection of creative coding, interactive media,

and software art. This six-credit workshop will focus on key techniques in programming visual, textual and aural media including recursion, compression, data-mapping, rule-based systems, stochastic processes and machine learning. Students will complete assignments and create mini-projects, building toward a publication-worthy final project of their choice.

The course will be run as a studio (project-driven, production-oriented, critique-based) augmented with regular lectures. Early lectures will introduce the core ideas of software art, theoretically, historically, and technically. As the course progresses, discussions will focus on differing conceptions of software art and the core debates that have driven the field. The final section of the course will focus on large-scale, student-initiated projects. Here students will develop, document, and present works-in-progress for critique, as they learn to develop original artworks and to enhance their practical awareness of software as a medium. In class presentations and critiques, they will be asked to contextualize their work, and the work of their peers, in the larger theoretical and historical frameworks of software art.

Course Intended Learning Outcomes (CILOs)

	CILOs	Weighting (if app.)	DEC-A1	DEC-A2	DEC-A3
1	Describe and reflect on different theories and historical examples of software art.		x		
2	Describe and implement different concepts and algorithms that have achieved prominence in software art.			x	x
3	Create software artworks via programming		x	x	x
4	Document their own works in a manner that clearly and accurately expresses their key processes and ideas.		x	x	x

A1: Attitude
Develop an attitude of discovery/innovation/creativity, as demonstrated by students possessing a strong sense of curiosity, asking questions actively, challenging assumptions or engaging in inquiry together with teachers.

A2: Ability
Develop the ability/skill needed to discover/innovate/create, as demonstrated by students possessing critical thinking skills to assess ideas, acquiring research skills, synthesizing knowledge across disciplines or applying academic knowledge to real-life problems.

A3: Accomplishments
Demonstrate accomplishment of discovery/innovation/creativity through producing /constructing creative works/new artefacts, effective solutions to real-life problems or new processes.

Learning and Teaching Activities (LTAs)

	LTAs	Brief Description	CILO No.	Hours/week (if applicable)
1	Lecture	Lectures about the history, theory and implementation of software art.	1, 2	
2	Workshop	Programming workshops to introduce students to the key methods and concepts in software art through visual, linguistic, and aural media	2, 3	

3	Critique	Student presentations to describe existing work and work in progress, share documentation, and participate in discussions of the work of other students.	1, 2, 4	

Assessment Tasks / Activities (ATs)

	ATs	CILO No.	Weighting (%)	Remarks ("-" for nil entry)	Allow Use of GenAI?
1	Complete weekly assignments (including 3 mini-projects) that demonstrate coding proficiency and an understanding of the genre of software art.	1, 2	55	-	Yes
2	Participate actively in class workshops and critiques.	3, 4	10	-	Yes
3	Complete a final project by means of code, together with a web site that documents the work and process, and includes a theoretical reflection on its relation to and place within the tradition of software art.	1, 4	35	-	Yes

Continuous Assessment (%)

100

Examination (%)

0

Assessment Rubrics (AR)

Assessment Task

1. Complete weekly assignments (including 3 mini-projects) that demonstrate coding proficiency and an understanding of the genre of software art.

Criterion

Critical understanding of software art, ability to practice it creatively, and critical understanding of the relevant theoretical and historical background.

Excellent (A+, A, A-)

Very innovative implementations in the form of polished works and highly original reflections on the history and theory of software art that shows an in-depth and extensive knowledge of the subject matter.

Good (B+, B, B-)

Reasonably innovative implementations in the form of polished works and original reflections on the history and theory of software art that demonstrate strong knowledge of the subject matter.

Fair (C+, C, C-)

Average implementations of the assignments and accurate discussion of the history and theory of software art, but limited to a superficial understanding of a small set of examples.

Marginal (D)

Marginal understanding of the concept of software art and its theory and history, without much effort in the realization of the assignments.

Failure (F)

The student has not finished any work, or has made a work that shows little to no understanding of software art.

Assessment Task

2. Participate actively in class workshops and critiques

Criterion

Ability to articulate ideas about software art, to present work clearly and accurately, and to give feedback to peers.

Excellent (A+, A, A-)

Excellent ability to describe the student and professional work, to explain its implications, to take questions from other students, and to ask questions about and deliver valuable insights into the work of peers.

Good (B+, B, B-)

Good ability to describe works, to explain their implications, to take questions from other students, and to ask questions about the work of other students, although not outstanding in all of those areas.

Fair (C+, C, C-)

Fair ability to describe works, to explain their implications, to take questions from other students, and to ask questions about the work of other students, but not particularly original or insightful.

Marginal (D)

Marginal ability to describe the works and to ask questions on the work of other students, without expressing insights.

Failure (F)

No ability to describe the works, to explain their implications, to take questions from other students, or to deliver insights into the work of other students. The student is not articulate about her own work and unable to participate in the discussion of the works of others.

Assessment Task

3. Complete a final project by means of code, together with a web site that documents the work and process, and includes a theoretical reflection on its relation to and place within the tradition of software art.

Criterion

Ability to complete a larger work using code, to document it effectively, and articulate (in writing) its relation to theoretical and historical precursors in the field.

Excellent (A+, A, A-)

The work is extremely original in content and method. Its execution is highly polished.
The student has mastered the programming skills necessary to carry out the work.
The documentation is clear and accurate.
The student has new and deep insights into the history and concept of software art.

Good (B+, B, B-)

The work is reasonably original in either content or method.
Its execution is polished.
The student can use programming as an artistic medium effectively, but without full mastery.
The documentation is clear and accurate, with some historical and theoretical awareness, but lacking in significant insights.

Fair (C+, C, C-)

The student has made a work that is not particularly original but nonetheless demonstrates a good understanding of software art.
The student has a fair ability to program, although without a high degree of confidence. Programming is not completely a medium that the student can use with confidence, but instead sometimes an obstacle.
The documentation is clear and accurate, but not insightful and with limited historical and theoretical awareness.

Marginal (D)

Poorly executed work, although the student did put a little effort into the execution.
The lack of programing skills constitutes a substantial obstacle for the student.
No theoretical or historical awareness.

Failure (F)

The student completed no work or the work shows no understanding of software art.
The student is not a competent programmer.
The student cannot describe what software art is or its main historical tendencies/examples.

Additional Information for AR

All A+/A/A- grade assignment should comply with the highest performance of Discovery-oriented learning.

Part III Other Information

Keyword Syllabus

Creative Coding, Algorithmic Art, Intermedia, Electronic Writing, Digital Sound, Machine Learning

Reading List

Compulsory Readings

	Title
1	Arns, Inke. "Read_Me, Run_Me, Execute_Me: Software and its Discontents, or: it’s The Performativity of Code, Stupid’." Read_Me: Software Art & Cultures-Edition 2004
2	Hartman, Charles O. Virtual muse: experiments in computer poetry. Wesleyan, 1996
3	Geoff Cox, Alex McLean, Adrian Ward, “The Aesthetics of Generative Code”, http://generative.net/papers/aesthetics/
4	Florian Cramer. Concepts, Notations, Software, Art. 2002
5	Brian Eno, “Generative Music”, http://www.inmotionmagazine.com/eno1.html
6	Tero Parviainen, “JavaScript Systems Music” https://teropa.info/blog/2016/07/28/javascript-systems-music.html

7	Przemyslaw Prusinkiewicz, “Score generation with L–systems” http://algorithmicbotany.org/papers/score.icmc86.pdf
8	Ham, Ethan. "Randomness, Chance, & Art." Handbook of Research on Computational Arts and Creative Informatics. IGI Global, 2009. 85-102.
9	Molnar, Vera. "Toward aesthetic guidelines for paintings with the aid of a computer." Leonardo (1975): 185-189.

Additional Readings

	Title
1	Charles Ames, “Automated Composition in Retrospect: 1956-1986”, Leonardo, Vol. 20, No. 2, Special Issue: Visual Art, Sound, Music and Technology (1987), pp. 169-185
2	Fred Lehrdal, “Cognitive Constraints on Compositional Systems”, Contemporary Music Review,1992, Vol. 6, Part 2, pp. 97-121.
3	Warren Motte, ed. Oulipo: A Primer of Potential Literature. (Univ. of Nebraska Press, 1986)
4	Daniel Shiffman, The Nature of Code: Simulating Natural Systems with Processing. http://natureofcode.com/
5	Harry Mathews, and Alastair Brotchie, eds., Oulipo Compendium. (London: Atlas, 1998)
6	Hegselmann, Rainer, and Andreas Flache. "Understanding complex social dynamics: A plea for cellular automata based modelling." Journal of Artificial Societies and Social Simulation 1.3 (1998): 1.
7	David Hosking, “Art Let Loose: Autonomous Procedures in Art-making”, Leonardo http://www.mitpressjournals.org/doi/pdf/10.1162/LEON_a_01476
8	Mel Bochner, "The Serial Attitude". Artforum 6, no. 4 (December, 1967): 28–33.
9	Sol LeWitt, “Paragraphs on Conceptual Art” Artforum (June, 1967)