Software Instruments

Introduction

Point - a lot of design spaces are made navigable by software instruments

Creative Software is Personal

Claim - useful software supports human creativity

The Computational Meta-Medium

Point - different software instruments act on different design spaces

Data Science and Generative Design

Claim - software instruments are useful to make things

Game Making and Explorable Explanation

Claim - software instruments are useful to discover things

Three Desiderata of Software Medium Design

Claim - software instruments are used to equip artifact spaces with a navigable graph

Build Intuition

Point - discovering links between things

Find Artifacts

Point - discovering new things

Perform With

Point - going on walks between things

Performing the Meta-Medium

Point - software instruments encode intensive skill

Real-Time Feedback

Claim - concepts are invisible without instruments based on direct manipulation

Tutorials Can't Be Read

Claim - software instruments must be performed with to receive their benefit

Canvases Aren't Tutorials Yet

Claim - software instruments can point toward monuments, when they embody domain knowledge

Software Ecology

Example - interactive data visualizations comprise a family of software instruments

A Library is an Instrument

Claim - almost all software instruments are composed using software instruments

Composing an Instrument

Example - synthesizers are instruments that make instruments

Compiling an Instrument

Point - performances of programming using software instruments typically become artifacts

Data Visualization in HTML5

Case Study - using software instruments to develop an interactive example of library usage

Rich Output

Goal - solve the circle problem of apollonius in real time with direct manipulation

Renderer

Need - to see the circles

Data Binding

Need - to model the circles one-to-one with data

Callbacks

Need - direct manipulation of circle position and radius

Library Binding

Need - bidirectional communication with the Jupyter kernel

Heterogeneous Input

Claim - visualizations are engaging when you are in a tight interaction loop with the underlying data

Spatialization

Claim - videogames are a kind of heavily semantically loaded data visualization

Reactivity

Claim - interactive pipelines break often, you want them to break locally

Reproducibility

Claim - libraries are languages, so their portability is a serious priority

Live Intuition Pumps

Point - software instruments are highly expressive

Metaphors are Frozen Intuition Pumps

Claim - conceptual thought is organized as metaphor

Math is Several Metaphors in a Trenchcoat

Claim - media artifacts refine the dependency chains underlying conceptual thought

Culture is Materially Contingent

Claim - conceptual blends producing media artifacts exist in a sociotechnical ecology

Computational Media is Being Cultivated

Case Study - models of psychological theories

Game Design and Combat Epistemology

Claim - assemblages of concepts want to be expressed generatively

You Are the Monster in Universal Paperclips

Example - the paperclip maximizer plays an idle game

The Parrigues Tarot and its Co-Authors

Example - aspects selectively inform all readings

Design Space of Ontologies

Claim - interactive data visualizations are a means of querying ontologies, as in 'deep games'

Scores and Recipes

Point - software instruments are both learned and modified

Degrees of Implementation

Claim - software instruments with interaction loops of varying scope exist in the wild

Generators

Example - this person does not exist

Explorables

Example - online character creators

Editors

Example - digital clay

Generative Pipelines

Example - facial expressions and personality in Sims 4

Live Service

Example - character overhaul mods in Skyrim

Case Study of a Software Medium

Claim - Sims 3 is a software instrument for portraiture

Portraiture is Virtual

Claim - face space has arbitrarily high dimensionality

Instruments are Actual

Claim - Sims 3 CAS is used in a community practice of navigating face space

<u>Three Methods of Traversing Design Space</u>

Reprise - software instruments are used to navigate

Exploration

Goal - build intuition

Landmarks

Claim - you can tell if you're getting close in artifact space to a thing you've seen

Mimetic Learning

Claim - making the things you've seen tells you a lot about artifact space

Explanation

Goal - find artifacts

Narrative Tutorials

Claim - you can follow directions to move around in artifact space

Technical Language

Claim - precise directions motivate specialized terminology

Improvisation and Grammar

Goal - perform with

Glitch

Claim - small edit-distance errors in artifact space become deeply familiar

Grain of the Medium

Claim - you can sense how far the edit distance is to the artifact you want when you are familiar with a medium

<u>Conclusion</u>

Point - to get open-ended evolution, first facilitate play

<u>Co-Creative Meaning-Making</u>

Claim - we modify the instrument to support our performance

<u>The Research Debt Arms Race</u>

Claim - prototyping widgets are software documentation