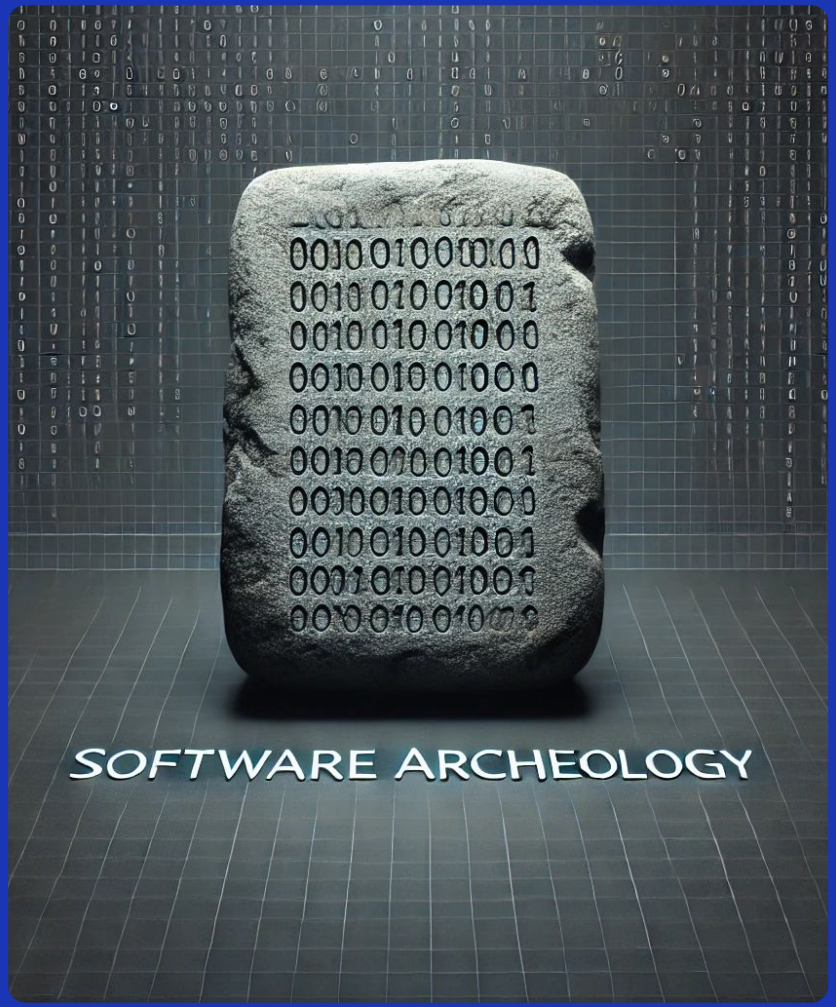
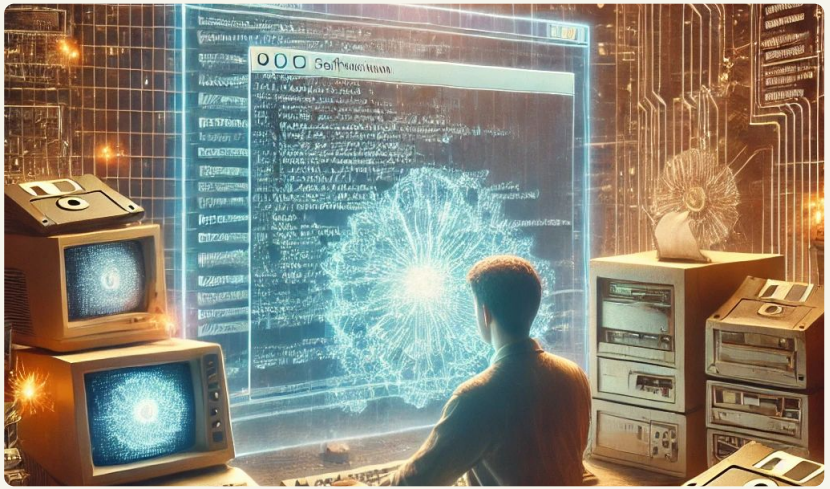
Software Archaeology

Joel A. Jaffe
2024-12-03



What Is Software Archeology?

- The study of insufficiently documented legacy source code
- A nascent field, given the brief history of software engineering
- Both art and science- a craft that has standard procedures yet requires intuition and clever innovation

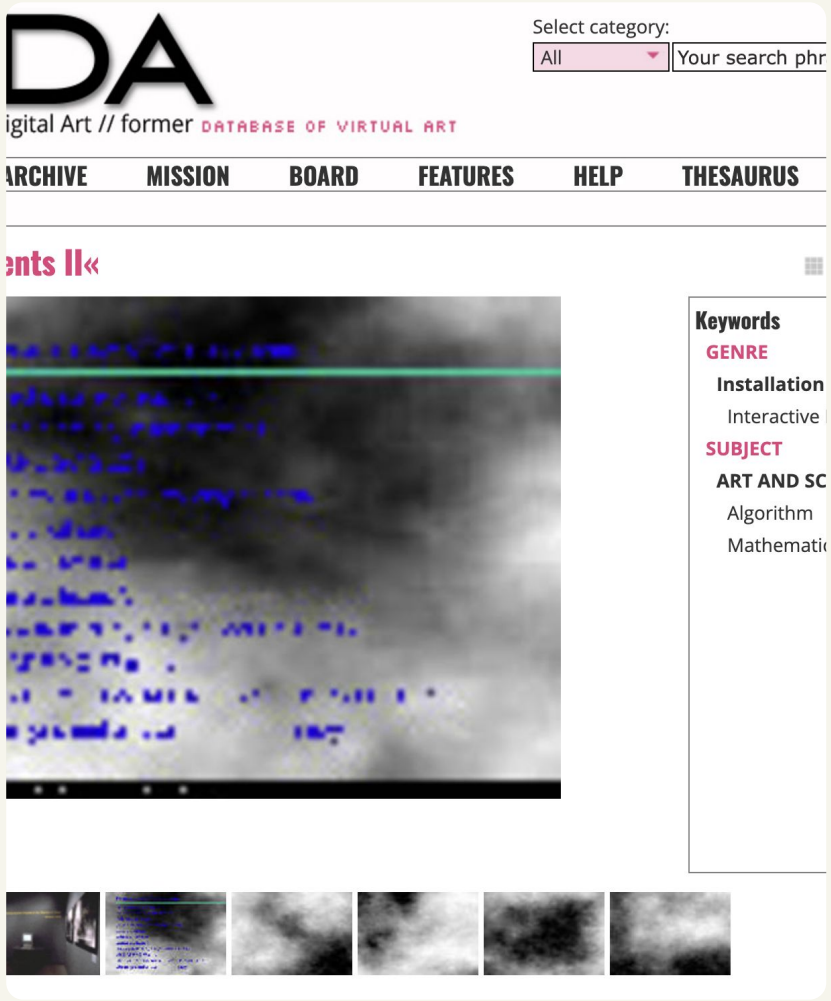


"[software archeology is] the recovery of essential details about an existing system sufficient to reason about, fix, adapt, modify, harvest, and use that system itself or its parts."

-G. Booch, Software Archaeology, ACM OOPSLA (2008)

Learn by Doing: A Day in the Life of a Software Archeologist

To further my understanding, I spent a few weeks working on my own software archeology project with George.



The Task:

Revive a Project from 90s Source Code (C)

Challenges:

- **Language Barrier:** I typically work with C++, not C
- **Documentation:** The original source code has no README, only inline comments
- **Dependencies:** The codebase had platform-specific dependencies from legacy operating systems
- **Practices:** The codebase was not version tracked with Git (it hadn’t been invented yet)

Steps:

Study the legacy code

Grep its structure, function, organization, etc.

Learn how it works at both high and low levels

Add version control

Organize codebase with modern version control practices: init a Git repository, host on GitHub, add documentation (README)

Organize dependencies

Rewrite the project to have a single, stable dependency.

We chose AlloLib for its cross-platform utility and intuitive API

Iterate on a draft

Start implementing pieces of the project in its new format.

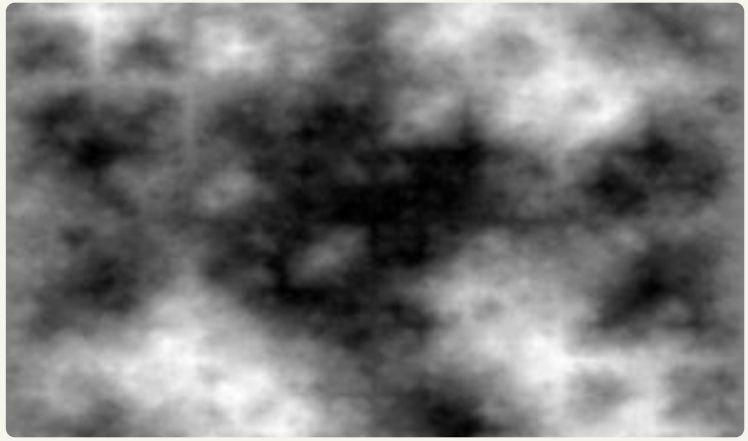
Each step should be tested individually.

Legacy Code

Equivalents II is a digital artwork developed by George Legrady in the 1990s.

It generates images procedurally using a modified version of the “diamond-square” algorithm.

The modification is that prompted user input (at the start of each generation) modifies how the image will form.



Diamond-Square Algorithm Wiki

Version Control

Version control is ubiquitous in modern software development. It makes the creation, storage and management of projects much easier for all involved.

Git is the primarily version control tool used today. It's ubiquity is due to the fact that it is free & open source.

It was invented by Linus Torvalds (Linux's namesake) to replace costly, proprietary source control software.

Dependencies

Dependencies describe pre-existing software that a project relies on to function. Their careful management is essential to a project's long-term stability (LTS).

Equivalents II had platform-specific dependencies for its graphics, so it could not run on computers other than the ones it was developed on.

We're rewriting *Equivalents* in AlloLib, which is cross platform and has LTS guarantees.

Our Draft:

https://github.com/joeljaffesd/Clouds2024

(Private for now...)

Next Steps...

Now that the project is version controlled and web hosted, multiple people can work simultaneously on the project, or pieces can be accomplished in series like baton passing.

Tangible outcomes are:

- Variable order diamond-square algorithm
- Bindings from the imGui interface to the legacy user-input processing code (C)
- Mapping the image generation to 3D for display in the Allosphere

If interested in collaboration, please contact George and I.

Thank you!