Every Object An Orchestra

A Software Preservation & Emulation Showcase

Source Code Preservation at Vienna Museum of Science and Technology

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Nika Maltar, Software Heritage Ambassador, April 28th 2026

An event hosted by the Digital Preservation Coalition (DPC)

tmw

Software Archive & Collection (SAC):

Archiving Austrian Creative Legacy Source Code: VinylOS (2016-2017)

VinylOS, Jonas Bohatsch and Josef Wieser: https://www.tumblr.com/vinylos?redirect_to=%2Fvinylos&source=blog_view_login_wall

- New collection and research department for Austrian Software Hermitage (Since 2021)
- Opportunity to build a collection and infrastructure for digital preservation, aligned with global standards and frameworks (Software Heritage)
- Participatory approach: collecting together with the grass root community & capturing technical knowledge and context from developers, maintainers, and users
- Software Collection: Unique creative software (e.g., curated indie video games, demoscene works, game art, alternative controller projects etc.)
- Software Archive: Everyday commercial software (e.g., stored on obsolete media or embedded in devices, includes system extraction and emulation toolkit development (softwareLAB for digital archaeology)
- Both areas are essential to ensure long-term access, understanding, research, and impact
- Standardized workflow: Software Heritage Acquisition Process (SWHAP) for archiving legacy source code
- Case study: VinylOS as a complex "**orchestra**" of hardware and software dependencies

Software Heritage tmw

VinylOS Dependencies Map

How hardware, software, and source code interact in VinylOS

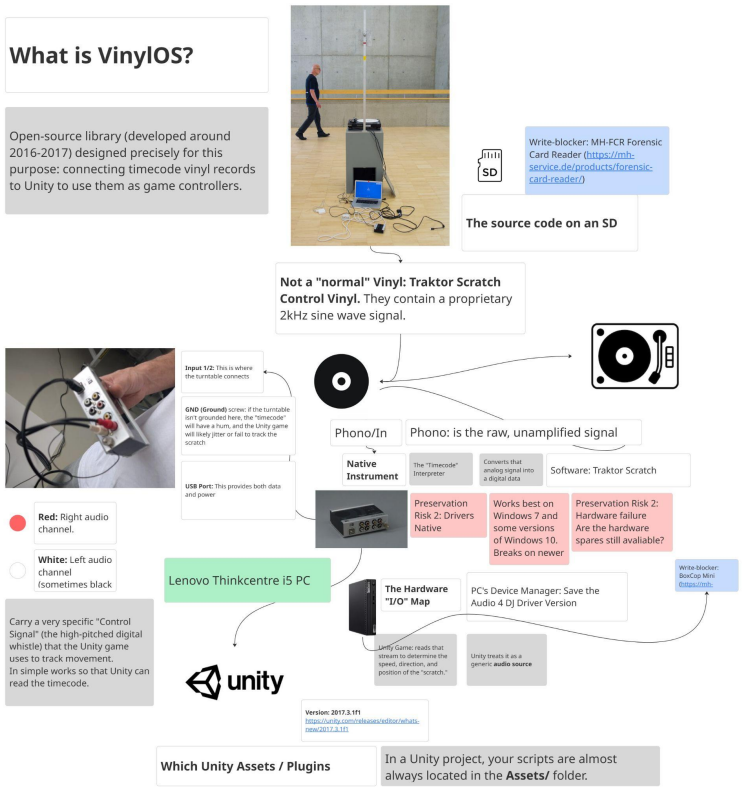
```
    }

    public void StartAudioIn() {
//      https://support.unity3d.com/hc/en-us/articles/206485253-How-do-I-get-Unity-to-playback-a-Microphone-input-in-real-time-
        audioIn = GetComponents<AudioSource>()[3];
        Debug.Log ("Playing audio from external input: " + audioDeviceName);
        audioIn.clip = Microphone.Start(audioDeviceName, true, 1, 44100);
        audioIn.loop = true;
        while ((Microphone.GetPosition(null) > 0)) { }
        Debug.Log("start playing... position is " + Microphone.GetPosition(null));
        audioIn.Play();
    }

    public void StopAudioIn() {
        Microphone.End (audioDeviceName);
        audioIn.Stop();
    }

    void Start() {

        foreach (string device in Microphone.devices)
        {
            // Debug.Log("Name: " + device);
            if (device.Contains ("Audio 4 DJ"))
                audioDeviceName = device;
        }
        // pitchFilter = new OneEuroFilter (filterFrequency, filterMinCutoff, filterBeta, filterDcutoff);

        m_audioPool = new AudioSource[numberOfSources];

        for (int i = 0; i < numberOfSources; i++) {
            m_audioPool[i] = gameObject.AddComponent<AudioSource>();
            if (musicFile [i] != null) {
                m_audioPool [i].clip = musicFile [i];
                m_audioPool [i].loop = true;
                m_audioPool [i].playOnAwake = false;
                m_audioPool [i].bypassEffects = true;
                m_audioPool [i].bypassListenerEffects = true;
                m_audioPool [i].bypassReverbZones = true;
                m_audioPool [i].ignoreListenerVolume = true;
            }
            else
                m_audioPool [i].clip = null;
        }
    }
```

- Alternative controller game (alt.ctrl): uses a record player instead of a mouse or gamepad
- VinylOS: a specialized open-source library (2016–2017) that converts the signal from a Traktor Scratch control vinyl into an interactive controller for Unity (2017.3.1f1)
- **Source code as blueprint**: the code maps external hardware and software dependencies

Software Heritage tmw

VinylOS Dependencies Map

How hardware, software, and source code interact in VinylOS (drawing a boundary around an objects)



VinylOS shortened Transcript.txt

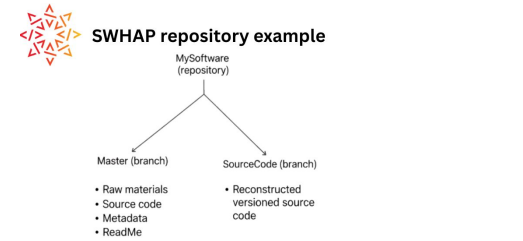
```
Das ist ein Case, den kann man so kaufen.
Den haben wir extra gekauft.
Aber drinnen ist so custom made.
Das ist quasi das Kit, das wir verschickt haben.
Und da ist alles dabei, außer der Plattenspieler, der ist extra.
Genau, den Plattenspieler konnten die Leute ausborgen, in Berlin zum Beispiel.
Wobei, wir sind schon oft mit dem Plattenspieler gereist.
Aber das Grundprinzip ist ja Plattenspieler und Projektion von oben.
Und deshalb braucht man so einen, wie nennt man das, einen Ständer, ein Gestell.
Und das ist das Originalgestell, aber das kann man schlecht verschicken.
Deshalb gibt es hier so ein anderes, das ist halt nicht so schön, das ist auch ein bisschen gebraucht schon.
(18-33)
Genau, das ist so die Platte, auf der der Beamer dann montiert ist.
(35-49)
Aber für diesen Koffer, so damit man das gesamte Paket hat, war es eigentlich ziemlich perfekt.
Ja, das ist eigentlich geiler.
Weil hier passt auch der Plattenspieler drauf und so.
(53-55)
Kann ich mich gar nicht erinnern, wie wir das gebaut haben.
Das haben wir auch ganz schön getüftelt.
Das ist halt nur Holz.
(59-61)
Weil ich jetzt gar nicht weiß, waren das echte Schrauben von den Düngern?
Stimmt, passt schon.
Zwei nur.
Ja, es waren nur zwei, genau.
Das müssen wir umdrehen, das muss oben sein.
Genau.
Macht aber eh [Sinn] mit den Biemann-Löchern.
(69-70)
So, jetzt brauchen wir einen Schraubenzieher.
Welchen?
So Kreuz.
Ich meine, die Schrauben, eine ist kürzer als die andere.
Schauen wir mal, ob du die reinfindest.
```

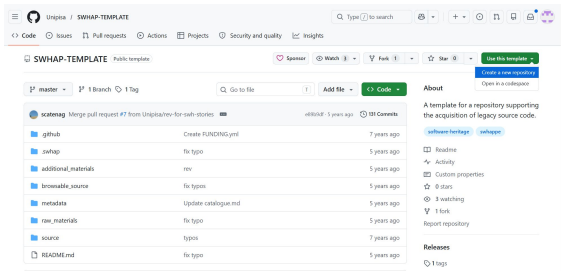
- ***Technical interview with the developers that was recorded and transcribed using Whisper (OpenAI Whisper)***

Software Heritage tmw

Software Heritage Acquisition Process (SWHAP)

Structured workflow from rescuing legacy code to making it identifiable, reusable, and preservable





Phase 1: Acquisition (Transfer from obsolete media)

- *Urgency due to fragility: "software that was created using outdated or now-obsolete tools, often stored in older formats, and may not be available through modern software repositories like GitHub or GitLab." (https://www.softwareheritage.org/swhap/)*

Phase 2: Curation (Curating technical & descriptive metadata)

- **Raw materials:** collecting source code in machine-readable form & ancillary material
- **Metadata:** research source, type, owner, licence
- **Documenting context:** talk to the early inventors, oral history & identify rights holders, authors, contributors
- **Read Me:** purpose, configuration, operation, known bugs

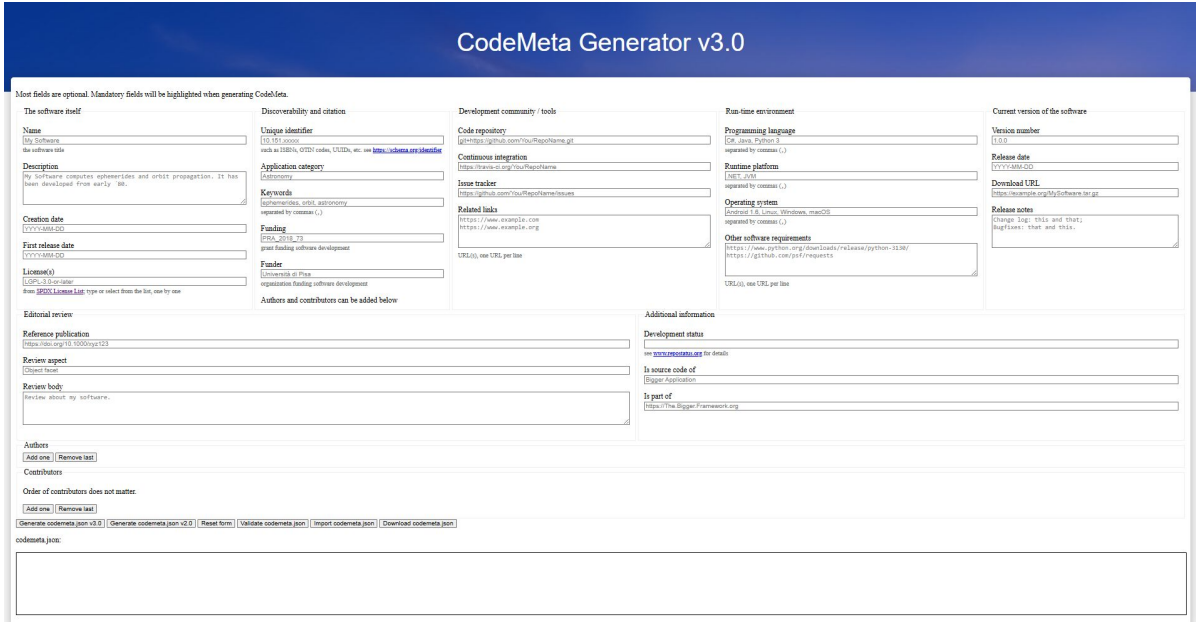
Stage 3: Preparing the local workbench (SWHAP template)

- *Use GitHub / local repository at TMW*
- *Use predefined folder structure (standardisation): https://github.com/Unipisa/SWHAP-TEMPLATE*

Software Heritage ✦ tmw

Software Heritage Acquisition Process (SWHAP)

Adding accurate museum labels by collecting proper metadata



Stage 4: Standardising metadata (CodeMeta)

- Structuring metadata from stage 2 into a standardised, machine-readable format (JSON)
- Stored as **dedicated metadata files**
- Enables: standardisation and interoperability

CodeMeta Template accessible on the GitHub Repository:

https://codemeta.github.io/codemeta-generator/

Software Heritage tmw

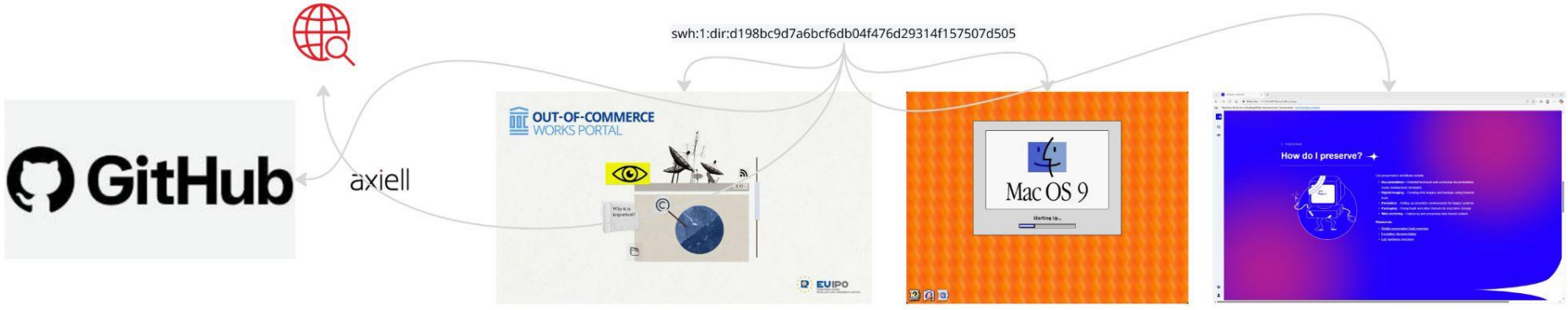
Software Heritage Acquisition Process (SWHID)

Assign Persistent Identity & Enable Traceability (SWHID)

Stage 5: Make Software Identifiable and citable

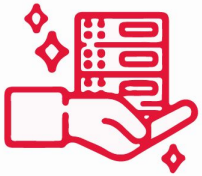
- Once the software is curated and structured, we assign a Software Heritage Identifier
- This is what makes the software uniquely identifiable and traceable across systems, enabling long-term preservation and reuse.
- SWHID ensures: integrity, non-ambiguity, persistence (provenience)
- Derived from the software itself
- Remains valid even if software is moved or renamed
- Generated independently (no third-party dependency), works across platforms
- Standard: ISO/IEC 18670 (April 23, 2025) https://www.swhid.org/

Decentralised approach: referencing the same Unique ID Across different platforms



Software Heritage Acquisition Process

Deposit or publish to Software Heritage Archive and establish relationships



- **Optional: Contribute the source code to a global, open archive (Software Heritage) or only work locally: https://archive.softwareheritage.org/**
- **Role of SWHAP: Provides a standardized way to document and structure source code**
- **Collective effort: Software preservation relies on shared standards and community participation**
- **Scope limitation: Does not solve hardware or dependency issues**
- **Core value: Makes code understandable, preservable, and linkable within the Software Heritage ecosystem**
- **Outcome: Ensures work is connected, not isolated, enabling long-term reuse and traceability**