PHIZURA : LIVE DJ RECORDING AND PLAYBACK SYSTEM USING COYPU IN PHARO

Personal Details

- **Name**-Nitya
- **Email:** nityabudhraja555@gmail.com
- **Github-link**
- **LinkedIn:profile**
- **University:** Graphic Era Deemed to be University
- **Program:** B.Tech in Computer Science & Engineering (2022–2026)
- **Expected Graduation Date:** June 2026
- **Organization:** Pharo Consortium
- **Project Title:** Phizura:Live DJ and Playback system using copyu in pharo
- **Mentor:** Sebastian Jordan (sebastian.jordan@inria.fr),
- Domenico Cipriani(mspgate@googlemail.com),

- **First-time contributor to GSoC:** Yes
- **Location :** Dehradun,Uttarakhand,India
- **Time Zone:** India (IST, UTC+5:30)

BRIEF BIO

I'm Nitya a Computer Science undergraduate with a strong foundation in systems programming, machine learning, and web technologies. I enjoy working with live coding tools and learning new technologies.I'm passionate about open source, live coding, and leveraging code for artistic expression.This project excites me because it combines music, real-time programming, and software development, making it an ideal opportunity to grow while contributing meaningfully to open source.

Project Description

Abstract

Phizura is a project that aims to build a live music recording and playback system for DJs using the Coypu package in Pharo. Coypu is a domain-specific language for music programming that uses an intuitive syntax inspired by TidalCycles. It is compatible with OSC-enabled apps and MIDI devices. Phizura will capture the live performance code and timing during a DJ session and automatically convert it into structured, replayable scripts. This project blends creative music programming with code documentation and playback, making live performances reproducible, remixable, and shareable.

Background

Live coding for music performance has emerged as a powerful form of creative expression, blending programming with real-time improvisation. While environments like Tidal Cycles and Sonic Pi have popularized this approach, the small talk-based Pharo ecosystem—with its live object-oriented environment—offers unique opportunities for introspection, interactivity, and educational engagement. Coypu, a domain-specific language built in Pharo, brings the expressive power of live coding into this environment, allowing users to compose and control music using intuitive code.

However, despite Coypu's strengths, there is currently no system within Pharo that allows DJs and performers to record their sessions, preserve timing information, and replay them later in a faithful and structured way. This lack of tooling limits Coypu's use in live performances, reproducibility, and educational demonstrations.

Phizura aims to fill this gap by introducing a reliable session recording and playback system tailored for live music coding in Coypu. By capturing commands, timestamps, and musical intent during a session, Phizura will allow performers to revisit, remix, and share their creations. The project draws on the power of Pharo's reflective capabilities, event handling, and live development features to build an extensible tool for creative technologists.

This project is not just about building a utility—it's about amplifying artistic expression through code and extending the capabilities of a growing live coding ecosystem within Pharo.

Project Proposal

Overview

Phizura is a live music recording and playback system designed for DJs and creative coders using the Coypu package in Pharo. It captures real-time code and timing from live sessions, turning performances into reusable, editable scripts. By leveraging Pharo's reflective environment and Coypu's expressive syntax, Phizura bridges live coding with documentation and reproducibility, making algorithmic music more accessible, educational, and shareable.

Technologies to Be Used

Technology	**Purpose**
Pharo	Main development language and environment
Coypu	Live Coding music dsl
OSC/MIDI	Communication protocol
Git & GitHub	Source control, hosting, and collaboration
CI/CD (GitHub Actions)	Automated testing and deployment

Plan of Action (Week-by-Week Timeline)

Timeline	**Tasks / Deliverables**
Community Bonding (May 20 - June 16)	Get familiar with the community, finalize design and interface specs, connect with mentor. Complete tutorials on advanced Pharo concepts. Publish roadmap on GitHub.
Week 1-2 (June 17 - June 30)	Setup and explore Coypu in Pharo, experiment with basic sound generation and OSC/MIDI.
Week 3-4 (July 1 - July 14)	Implement a command logger that records commands and timestamps during a session.
Midterm Evaluation (July 15)	Submit completed core linear structures, tests, and documentation.

Timeline	Tasks / Deliverables
Week 5-6 (July 16 - July 28)	Build a system that converts the logged data into structured, readable Pharo/Coypu code.
Week 7-8 (July 29 - August 11)	Add playback functionality to replay the recorded script.
Week 9-10 (August 12 - August 25)	Testing and improving user experience; optional UI or visualization component.
Final Week	Submit final work, gather community feedback, deploy to Pharo Catalog, publish blog post and documentation.

Plan of Action

Phase 1: Community Bonding and Exploration (Community Bonding Period)

- Engage with mentors and the Coypu/Pharo community.
- Finalize the feature scope and clarify any technical constraints.
- Explore Coypu's syntax and playback mechanisms with hands-on examples.
- Set up the development environment and version control system.

Phase 2: Logger Development (Weeks 1–4)

- Build the core module to capture live-coded commands and their timestamps.
- Ensure compatibility with Coypu's DSL and Pharo's event system.
- Design data structures for storing session data.
- Implement basic tests to verify command capture accuracy.

Phase 3: Script Generation System (Weeks 5–7)

- Develop logic to convert raw session logs into readable, modular Coypu/Pharo scripts.
- Ensure syntax validation and proper formatting.
- Allow customization of output (e.g., grouping by instrument or pattern).
- Include unit tests to verify output consistency.

Phase 4: Playback Engine (Weeks 8–10)

- Implement a player to replay generated scripts with correct timing.
- Integrate OSC/MIDI communication to send playback instructions.
- Add control options like play, pause, and stop.
- Optimize timing precision and responsiveness.

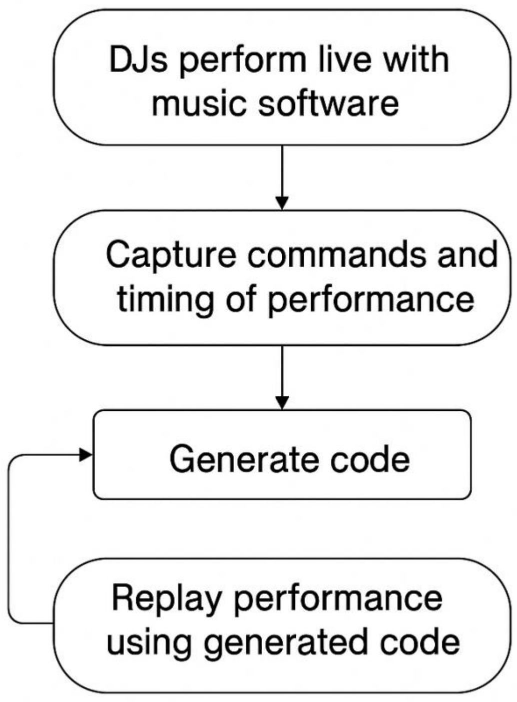
Phase 5: UI and Visualization (Optional) (Weeks 11–12)

- Build a minimal Morphic-based interface to interact with logs and playback.
- Visualize commands on a timeline or grid (optional stretch goal).
- Conduct user testing and polish UX.

Phase 6: Final Touches and Submission (Week 13)

- Comprehensive testing and bug fixing.
- Write developer and user documentation.
- Create a demo video or presentation showcasing the tool.
- Final submission for evaluation.

Architecture Diagram


DJs perform live with music software
Capture commands and timing of performance
Generate code
Replay performance using generated code

Commitments

During weekdays (Monday through Friday), I will be able to dedicate 3–5 hours to project work, ensuring alignment with UK working hours for smooth communication and collaboration. To complement this, I plan to invest 7–8 hours per day on weekends (Saturday and Sunday), providing additional time to meet deadlines and maintain consistent progress throughout the program.

Github repo:
https://github.com/testgithubrit11189/phizura/blob/main/README.md

Post-GSoC Plans

After GSoC, I plan to continue maintaining and enhancing Phizura by:

Improving playback accuracy, refining the user interface, and adding support for new features like track layering, real-time visualization, and session exporting.

Continuing to contribute to Coypu and Pharo, exploring how Phizura can integrate with other creative coding environments and grow into a full-featured tool for live performance and teaching.