Squawk Farm: A Constraint-Guided System for Interactive Audio-Visual Music Composition

Kayli Requenez
Massachusetts Institute of Technology
kayli195@mit.edu

Abstract

Squawk Farm is an interactive music system that transforms user-recorded vocal sounds into animated musical creatures that perform together within a shared rhythmic and harmonic structure. The system is designed to support playful, exploratory music creation while enforcing constraints that preserve musical stability. By combining streamed audio recording, symbolic sequencing, context-aware rhythm generation, and audio-driven visual synthesis, Squawk Farm enables both novice and experienced users to construct coherent compositions without requiring formal musical training. This document presents the system design, architectural decisions, and extensibility considerations underlying Squawk Farm.

1 Introduction

Interactive music systems often struggle to balance expressive freedom with musical coherence, particularly for non-musicians. Squawk Farm addresses this challenge through a constraint-guided design philosophy: users are encouraged to experiment freely, but all interactions are structured to prevent rhythmically or harmonically invalid states.

Users create musical "animals" by recording short audio samples or selecting preset sounds. Each animal contributes a looping musical voice to a shared "garden," where all sounds remain synchronized to a global tempo, meter, and harmonic context. Rather than exposing unconstrained audio editing, the system progressively reveals control, allowing users to build complexity while maintaining stability.

—

2 System Overview

Squawk Farm is structured around four primary user-facing screens that together guide users from sound capture to full musical playback. These screens serve as the organizational backbone of the system, with each exposing a different level of musical control while sharing a unified underlying timing and harmonic framework.

2.1 User-Facing Screens

- **Recording Screen**: Users record vocal audio or select preset sounds, visualize waveforms, and extract rhythmically quantized loops.
- **Loop Sequencer Screen**: Users edit an individual animal's rhythmic and pitch behavior using a grid-based symbolic sequencer, similar to a piano roll editor.
- **Chord Progression Screen**: Users select high-level harmonic progressions that govern global pitch transformations.
- **Garden Screen**: Users experience the full composition as a cohesive audiovisual performance environment.

All screens operate over shared global timing, harmonic, and transport state. Audio and visual components are fully decoupled and communicate only via stable identifiers, enabling modular development and future interface changes without affecting musical logic.

—

3 Audio Recording and Loop Extraction

User audio is captured through a streamed recording pipeline with live waveform visualization. Audio is written into a rolling buffer, enabling immediate feedback during capture. Recordings are capped at two measures (eight beats), equal to the length of one full composition loop, to ensure musical compatibility.

After recording, users select a loop region using draggable margins over the waveform. Available loop lengths are presented as quantized categories

(tiny, short, medium, large), each corresponding to a fixed number of beats. The selected region is always aligned to the global beat grid. Amplitude normalization, pitch correction, and fade-in/out processing are applied to avoid artifacts such as clicks.

Preset audio samples are supported as first-class inputs and processed through the same pipeline. This ensures shared logic between recorded and preset sounds and lowers the barrier to entry for users uncomfortable with recording.

—

4 Global Timing and Grid Abstraction

All musical time in Squawk Farm is defined by a shared grid abstraction specifying tempo, meter, number of measures, and pulses per beat. Time is represented internally in discrete slots, allowing consistent scheduling across audio playback, sequencing, and visualization.

A countdown metronome is used before recording to help users with timing. During playback, a shared transport drives all loops and visual feedback. Pause and resume operations preserve the current transport position across editing and playback contexts.

—

5 AudioLoop and Symbolic Sequencing

Each animal owns an `AudioLoop`, which separates audio data from symbolic musical structure. Audio is stored as a buffer, while edits are represented as symbolic events mapping quantized time slots to pitch values within a single octave.

Symbolic edits include adding, deleting, and dragging notes in time or pitch, as well as octave shifting. Notes are prevented from overlapping, and all events remain aligned to the global grid. During playback, the system transposes and triggers the shared audio buffer according to the symbolic schedule, combining the expressiveness of sampled audio with the editability of MIDI-style sequencing.

5.1 Loop Sequencer Interface

The loop sequencer screen provides a grid-based interface for editing an individual animal's musical behavior. The horizontal axis represents quantized time slots aligned to the global grid, while the vertical axis represents pitch within a constrained octave range.

Users may add notes by clicking empty grid slots, reposition notes by dragging them, and remove notes by dragging them to a trash target. Notes cannot overlap in time, ensuring structural validity. Pitch changes remain musically constrained, with octave shifts exposed as discrete controls.

The sequencer operates locally per animal but remains synchronized with the global transport. During playback, only the active loop is visually animated, similar to track-focused editing in traditional digital audio workstations.

In addition to manual editing, users may toggle between automatically generated beat templates derived from the surrounding composition.

—

6 Beat Template Generation and Scoring

To maintain rhythmic coherence while supporting expressive variety, Squawk Farm introduces a beat template system. When an animal is created or edited, the system generates multiple candidate rhythmic patterns based on the animal's role, loop duration, and the global meter.

Candidate patterns are scored according to overlap with existing animals, rhythmic density, emphasis on strong beats, and balance between sparsity and fullness. The top four scoring templates are presented to the user.

—

7 Harmonic Framework and Animal Roles

Each animal is assigned a musical role (bass, harmony, or melody) at creation time. Role assignment is determined by the detected pitch of the sample relative to the global root, the duration of the sample, and the current distribution of roles within the composition. Borderline cases are biased toward underrepresented roles to preserve harmonic balance.

The global root is derived from the first animal and remains fixed for the session. Pitch generation is constrained to a pentatonic scale to avoid dissonance. After a beat template is selected, pitches for each note are assigned using a constrained random walk within the pentatonic pitch space, anchored around the animal's base pitch and clamped to a single-octave window. This preserves melodic variation while maintaining harmonic stability. Chord progressions advance at fixed structural intervals

and are applied by transposing notes at playback time rather than rewriting loop data.

7.1 Chord Progression Interface

The chord progression screen exposes high-level harmonic control without requiring users to edit individual notes. Users select from predefined progressions appropriate to the current key mode. Chord changes apply globally at fixed boundaries through real-time transposition.

—

8 Playback and Scheduling

Playback is driven by an audio scheduler that resolves pitch, harmonic context, and timing at trigger time. Audio and visual systems remain decoupled, with visuals subscribing to playback events without owning audio state.

—

9 Creature Generation and Visual Mapping

Animals are generated procedurally from extracted audio features. Pitch influences size and proportions, while randomized palettes and shape parameters ensure visual diversity. When animals sing, their mouths animate in response to playback.

—

10 Garden Environment

The garden functions as a non-editing playback environment where all animals perform together. Spatial placement is aesthetic only. A sun element pulses on the beat, reinforcing the global tempo visually. A hard cap of twenty animals is enforced for performance and clarity. Animals may be repositioned freely or removed via a trash interaction.

—

10.1 Scalability and Architecturally Prepared Extensions

Although the system initializes with fixed defaults (two measures, fixed tempo, 4/4 time signature, major key), the loop engine was designed from the outset to support broader flexibility. Beat generation supports alternative meters, and recording length granularity is extensible via the pulse-based grid.

Infrastructure for persistent saving and loading was architecturally prepared but not fully exposed due to time constraints. Additional prepared extensions include expanded audio effects, support for additional musical scales, role-aware chord tone assignment, and a “graveyard” system for deleted animals.

—

11 Conclusion

Squawk Farm demonstrates how constraint-aware system design can enable playful music creation without sacrificing musical integrity. By organizing interaction around progressive screens and shared musical structure, the system supports both exploration and compositional depth.

Acknowledgments

Squawk Farm was developed as a collaborative course project with Maxine Perroni-Scharf and Raymond Brookman. Maxine contributed the procedural animal generation system, pitch assignment logic following rhythm generation, the chord progression selection interface and content, and the overall visual aesthetic of the application. Raymond contributed early-stage UI structure, the recording countdown metronome, and the preset audio selection interface. The author designed and implemented the core system architecture and integrated all components into a unified pipeline.

Author Contributions

The author designed and implemented the core system architecture, including the global timing grid, symbolic loop engine, beat template generation and scoring system, harmonic framework, role assignment logic, and audio scheduling pipeline. The author also implemented the audio recording pipeline, loop extraction and quantization logic, symbolic sequencer model, and system extensibility infrastructure.