intelligrit

Labs Case Study

Omnivox: Cross-Platform Speech Infrastructure

A usable Rust speech server for Emacspeak, evolving SwiftMac's macOS-native lessons into a broader cross-platform architecture.

Back to Case Studies | Full PDF | GitHub

Robert Melton / Intelligrit Labs | Usable now, still evolving

Rust	3
WORKSPACE	PLATFORM TARGETS
170+	MIT
TESTS	LICENSE

Usable Today | In Flight | Cross-Platform

The Short Version

Omnivox is a Rust speech server for Emacspeak. It is usable today, and it is also still in active evolution. The point is to take the real lessons from SwiftMac and rebuild the system around a broader architecture: platform backends, a shared audio pipeline, concurrent streams, and fallback engines.

SwiftMac proved the macOS-native path. Omnivox asks the next question: what should this look like when the goal is durable, cross-platform accessibility infrastructure rather than one platform's best implementation?

> ***SwiftMac proved the local solution. Omnivox generalizes the architecture.***

Why Rebuild

The first working answer was valuable. The broader problem needed a broader system.

SwiftMac solved a real accessibility problem on macOS. It made Emacspeak responsive and useful for Robert's daily development work. But a tool shaped tightly around one operating system has a natural ceiling.

Omnivox is the next architecture. It keeps the Emacspeak protocol boundary, but splits the implementation into focused Rust crates for command parsing, TTS engines, audio processing,

and CLI/server behavior. That makes the system easier to test, easier to port, and easier to evolve without losing the user-facing behavior that made SwiftMac useful.

SwiftMac taught	Omnivox generalizes
Native speech can be fast enough for daily work	TTS backends behind a shared engine trait
Audio icons and tones matter	Common buffer format for speech, tones, and audio files
Users need independent volume and routing	Effects pipeline with volume, silence trimming, and channel routing
Simple protocol boundaries are valuable	Platform-agnostic parser, queue, and state crates
Local tools should not depend on cloud services	Native and fallback engines running locally

The Architecture

Protocol in, typed state and queues inside, standard audio buffers out.

The Omnivox workspace separates the system into clear parts: omnivox-core for commands, queues, and state; omnivox-tts for platform backends; omnivox-audio for buffers, effects,

tones, file loading, and output; and omnivox-cli for the actual server binary.

All generated audio converges on a common stereo floating-point buffer format. From there, effects can trim silence, scale volume, route channels, and send speech, tones, and audio icons through separate streams. That keeps the audio model consistent even when the source is different.

Protocol Core

Command parsing, state, and queues live in platform-agnostic Rust.

TTS Backends

macOS native speech, Windows speech, Linux/espeak paths, and fallback behavior sit behind one interface.

Audio Pipeline

Speech, tones, and audio icons share a buffer and effects pipeline before playback.

What This Proves

The important signal is engineering judgment, not language preference.

Omnivox is a good Labs case study because it shows when to keep using a working tool and when to rebuild the shape underneath it. The user need stayed the same. The architecture changed because the next version needed portability, testability, and a clearer boundary between protocol, synthesis, and audio output.

That is a pattern Intelligrit uses in larger systems too: preserve the boundary that works, replace the part that has become the constraint, and make the new design easier to operate and verify.

The caveat is intentional: Omnivox is usable now, but still moving. That makes it a live engineering artifact, not a polished buyer proof claim.

See SwiftMac
Our Approach

SERVICES

Overview
AI Integration
Legacy Modernization
Custom Development
Our Approach

CASE STUDIES

Overview
Integrity
GraphWizard
SwiftMac
Omnivox

DelveGlass

TWI Map

MORE

Products

Open Source

Blog

COMPANY

Let's Talk

Careers

Contact

Capabilities

LinkedIn

GitHub

Startup DNA. Enterprise-grade. Source available.