SiMa.ai Palette Neat

Tech Brief

The New Software Standard for Physical AI

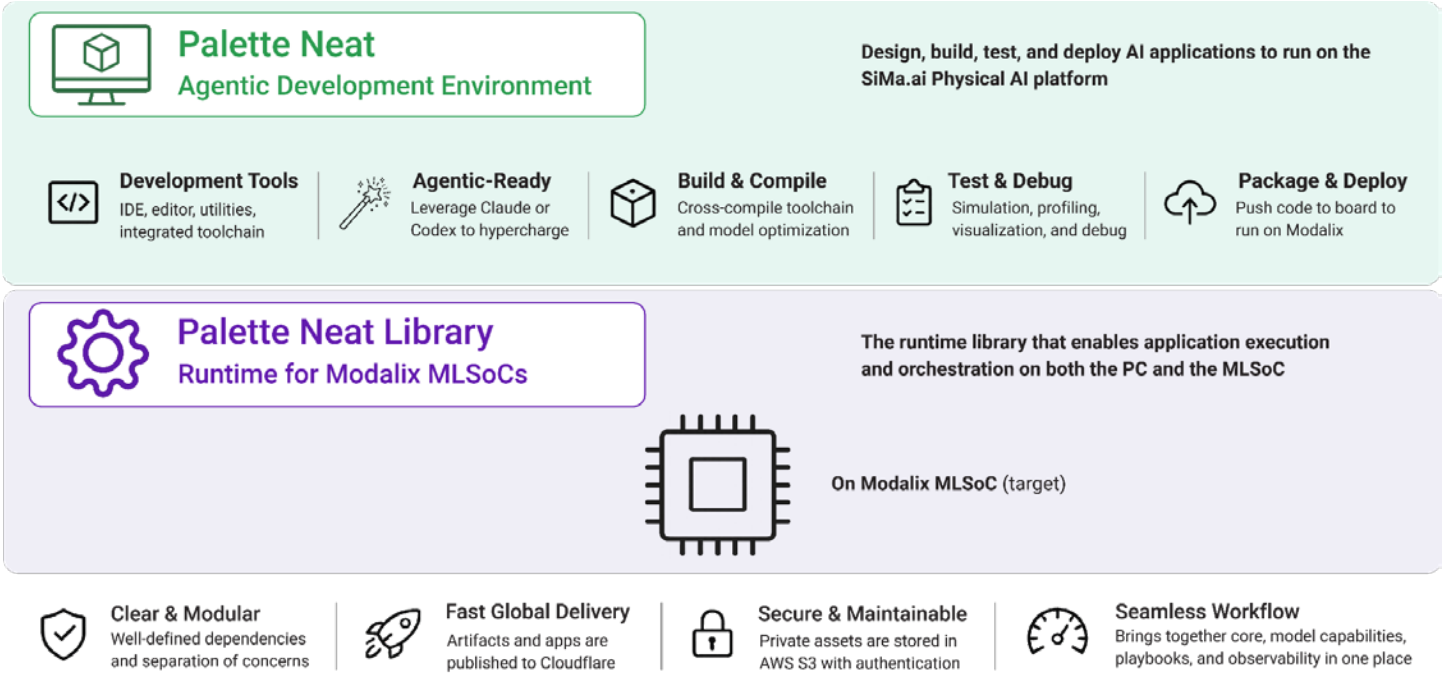
Accelerating development and deployment from months to days.

Manuel Roldan, Software Product Manager, SiMa.ai®

Building real-time Physical AI applications—like high-performance, multimodal object tracking on a drone in a GPS denied environment within a small power envelope—is notoriously difficult. It involves coordinating specialized hardware, managing complex inputs and outputs, and optimizing every microsecond with the highest precision.

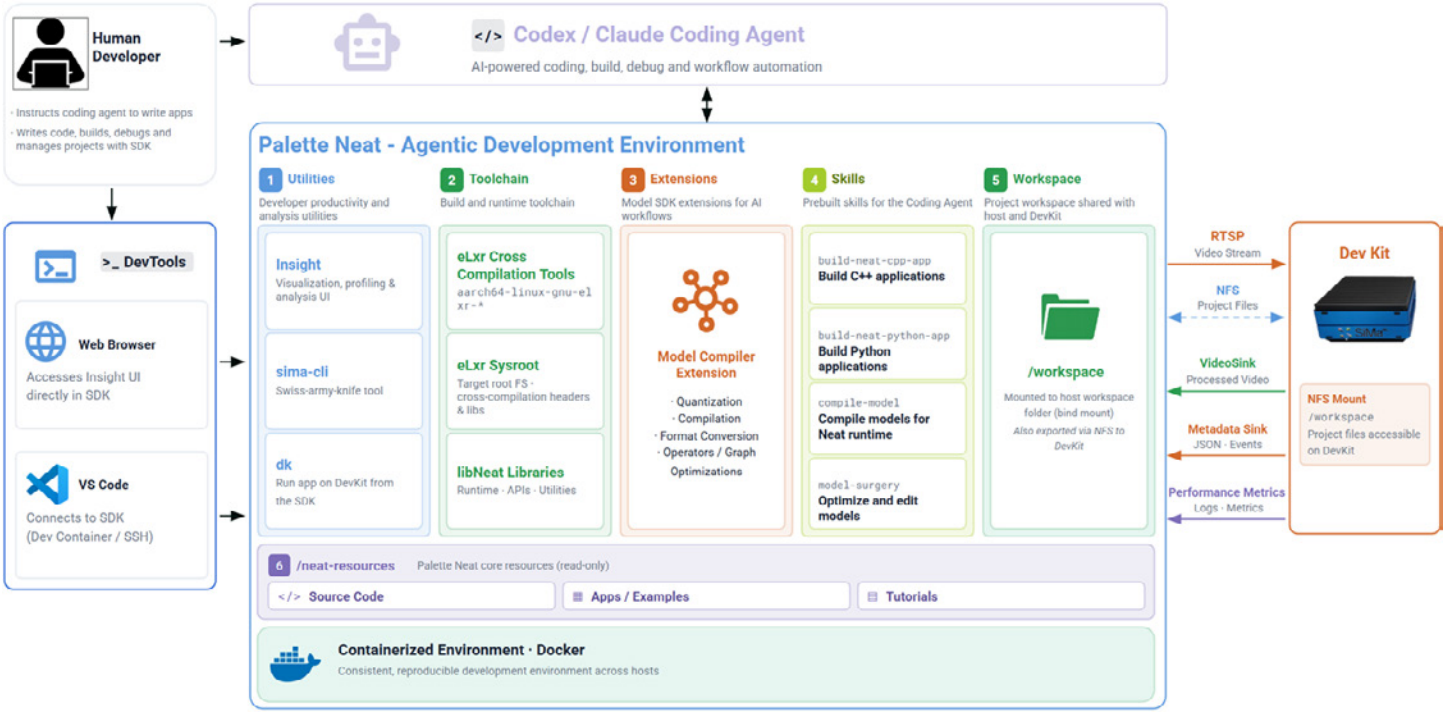
Palette Neat™ is the software agentic AI environment that makes this not only possible on the SiMa.ai Modalix™ chip but dramatically faster and easier to develop and deploy. By abstracting low-level heterogeneous compute complexity into a structured, high-level library, Palette Neat enables developers to move from model assets to working Physical AI applications in days instead of months.

When developers first encounter the **Palette Neat,** it feels almost magical because you can literally describe an application to the AI agent, and it just materializes. The secret is in the Palette Neat Library underneath, which consists of a set of libraries and nodes to solve different types of operations that let developers quickly build applications that use the Modalix MLSoC. You can think of it as the 'PyTorch for heterogeneous compute,' providing abstractions that simplify development across the Modalix MLSoC architecture. Without the Palette Neat Library under the hood, developers would need to learn the chip's low-level details (i.e. how to move data into the right memory, how to format tensors, how to talk to the accelerator). With Palette Neat, you merely call high-level functions like "load this model" and "run this video through it" and the framework handles the rest.



The Palette Neat workflow was deliberately designed to be exceptionally friendly to code generation and - even more importantly - easy for developers to use and debug.

This allows teams to move from installation to working applications significantly faster, while reducing the amount of low-level integration work traditionally required for Physical AI development.



The Core Principle: Clarity and Debuggability

Palette Neat abstracts the complexity and pain of working with low-level media and hardware orchestration (such as GStreamer), replacing it with a stable, typed, and predictable development experience.

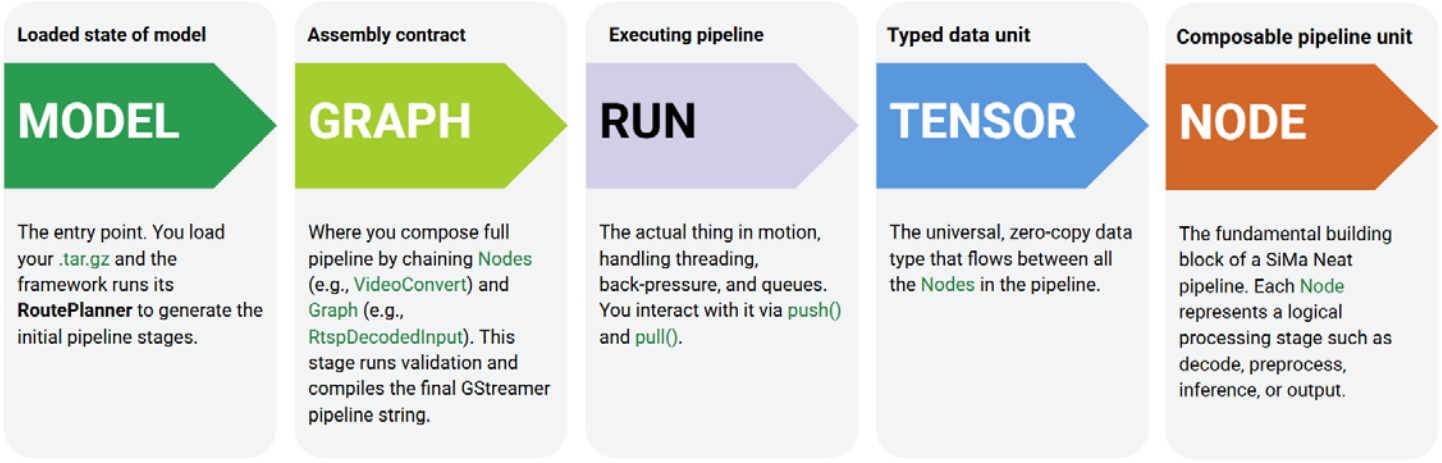
- Structured Errors, Not Strings: When issues occur, Palette Neat provides machine-readable error codes and human-readable notes in a GraphReport. Debugging tasks that traditionally involve cryptic GStreamer negotiation failures are faster, more determivnistic, and easier to resolve instead of becoming a forensic exercise.
- No Silent Fallbacks: Predictability beats convenience. The library deliberately fails fast rather than auto-correcting.
- For example, if you pass BGR pixels to a model trained on RGB, the build fails loudly, allowing you to fix the real problem immediately instead of chasing accuracy regressions later.
- Strong, Small Abstractions: The library is built around five primary concepts - Model, Graph, Run, Tensor, and Nodes, which bounds the impact of any individual component change.
- Composability by Design: Pipelines are built from isolated Nodes and Graph. When a pipeline doesn't work, the failure localizes to one component, allowing you to iterate on that single piece without impacting the entire application.
- Python and C++ Parity: The same conceptual model is maintained across Python and C++ - minimizing context-switching or the need to re-learn the API.

- One Workspace, One Loop: The development environment uses a single workspace and a shared filesystem mount, simplifying the compile-and-deploy story. You build on a fast x86 or Arm64 host and deploy to the DevKit with one command.
- Open by Design: Palette Neat is open source, providing transparency, extensibility, and the flexibility to integrate with existing tools and workflows.

The Data Flow Architecture: Speed, Zero-Copy, and Seamless Conversion

The performance of the Modalix MLSoC relies on optimized data formats (INT8 or BF16), unified memory architecture, and efficient pipeline execution. Palette Neat abstracts this complexity, allowing developers to focus on application behavior rather than low-level data orchestration.

- FP32 In, FP32 Out: For model input and output, the user-facing format is almost always the standard FP32. The framework's route planner automatically inserts necessary pre-processing and post-processing stages to translate your FP32 data into the MLA's required INT8 or BF16 format, and back again.
- Fused Kernels for Efficiency: Palette Neat has options when loading a model to upgrade the pre and post stages to fused kernels.
    - Generic Preproc performs resizing, color conversion, and normalization in a single pass before inference.
    - BoxDecode handles detection-specific post-processing including tensor parsing, confidence thresholding, and non-max suppression, returning clean bounding-box outputs without requiring custom implementation.
- Zero-Copy Processing: The Modalix MLSoC architecture uses partitioned memory - meaning all processors (A65 CPU, EV74 vector core, MLA accelerator) can read from and write to the same physical memory. The Tensor type manages data movement efficiently, ensuring the framework moves handles (pointers and metadata) instead of duplicating large data buffers, thereby minimizing overhead and maximizing pipeline performance.



The Application Lifecycle: From Recipe to Runtime

The Neat framework guides you through a clear application lifecycle using five primary concepts:

Timing Decisions: Sync vs. Async?

When you build your Session, you choose a timing mode for the resulting Run:

- Synchronous Mode (run()): For one-shot operations, like processing a single image or a batch of files. Your code blocks until inference is complete.Fused Kernels for Efficiency: Palette Neat has options when loading a model to upgrade the pre and post stages to fused kernels.
- Asynchronous Mode (build(RunMode::Async)): For continuous streaming inputs (like cameras). This mode spins up threads and queues, allowing the pipeline to absorb input jitter and maximize throughput by processing multiple frames simultaneously.

Coordination with Graphs

While many applications only require a single pipeline, when you need to coordinate multiple streams—such as frame-aligned multi-camera fusion or running a single input through parallel models—you use the Graph. This layer orchestrates multiple independent Run instances using an actor model, explicit data-flow boundaries, and bounded queues, allowing for correct and scalable concurrency without the risk of deadlocks from shared locks.

The Result: Fastest Development, Maximum Performance

Palette Neat doesn't just deliver superior performance on the Modalix chip; it completely re-architects the developer experience around clarity and speed. Tasks that traditionally required weeks of manual pipeline integration and optimization can now be deployed in days, and in many cases hours, using high-level Python and C++ workflows built on Palette Neat.

The classic pain points of Physical AI—cryptic, string-based GStreamer errors, manual memory management, and brittle, non-deterministic pipeline assembly—are now abstracted away. By providing strong, small abstractions like the Tensor (which ensures zero-copy data flow across the A65, EV74, and MLA) and fusing complex stages into single, high-efficiency kernels like Generic Preproc and BoxDecode, Palette Neat ensures that you get maximum hardware utilization without sacrificing development velocity.

Whether you are using the simplified Model::run() shortcut, composing custom Graph pipelines, or orchestrating frame-aligned multi-camera systems with the Graph, you are working with an API that is deterministic, debuggable, and built to scale. The result is a shift from fighting low-level complexities to focusing on high-level application logic, allowing you to move from prototype to production on the edge with unprecedented speed and confidence.

Visit our Developer Center to start building with Palette Neat.

SiMa.ai

About SiMa.ai

SiMa.ai is a leader in Physical AI, delivering a purpose-built, software-centric platform that brings best-in-class perforamance, ower efficiency, adn ease of use to Physical AI applications. Focused on scaling Physical AI across robotics, automotive, industrial automation, aerospace & defense, smart vision, and healthcare, SiMa.ai is led by seasoned technologists and backed by top-tier investors. Headquartered in San Jose, California.
Learn more at www.sima.ai.