



# Unlocking Embedded Vision Innovation: The Kamaros API for cross-platform camera interoperability



Image Sensors Europe  
London, March 2025

# Yours Truly



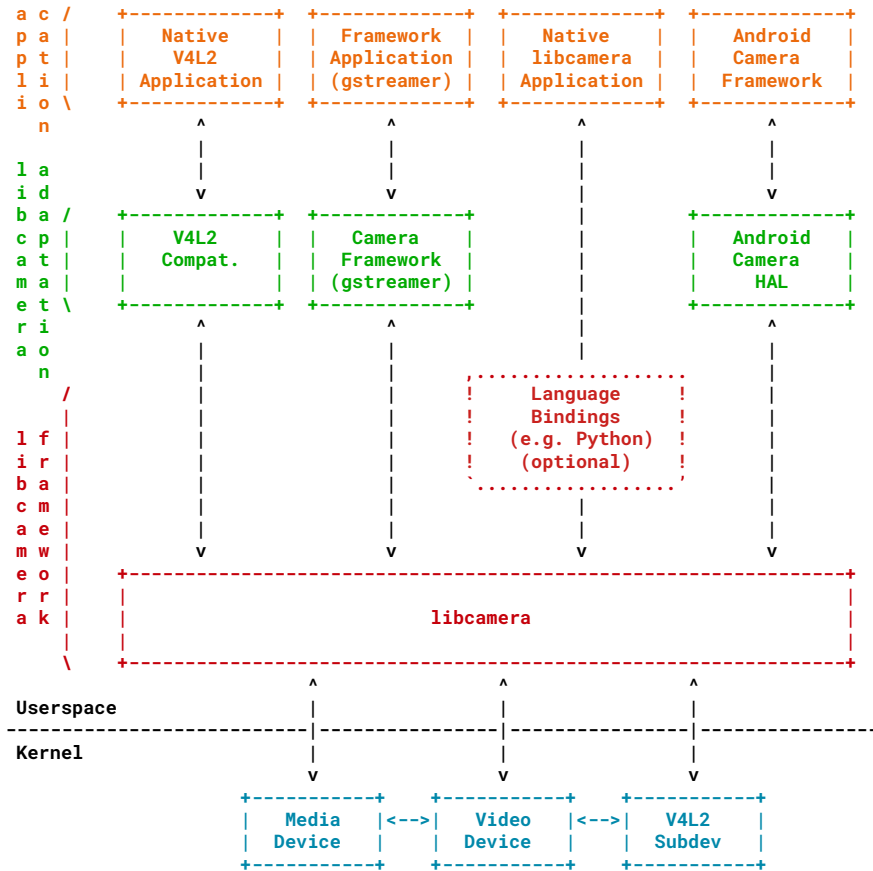
Laurent Pinchart is the founder and CEO of Ideas on Board, a software consulting company specialized in delivering imaging solutions for Linux across all markets, from image capture to display.



With 20 years of experience as a Linux kernel developer and maintainer, Laurent has driven the design of the Linux kernel camera API and has participated in multiple industry working groups to standardize camera protocols. Most recently, he has started the libcamera® project to give Linux a full camera stack in collaboration with silicon vendors and OEMs.



libcamera is an open source camera stack and framework for Linux, Android, and ChromeOS.



# Khronos Connects Software to Silicon



**KHRONOS**  
GROUP

Non-profit Standards Consortium creating  
open, royalty-free standards

Focused on runtime APIs and file formats  
for 3D, XR, AI, vision, and  
parallel compute acceleration

Member-driven, open to any company

Founded in 2000

~ 160 Members | ~ 40% US, 30% Europe, 30% Asia

ISO/IEC JTC 1 PAS Submitter

# Increasing Needs for an Embedded Camera API Standard

## Increasing Sensor Diversity

Including camera arrays and depth sensors such as Lidar



## Multiple Sensors Per System

Synchronization and coordination become essential



Cost and time to integrate and utilize cameras is major industry friction point

Lack of robust camera software ecosystem limits use of camera/sensor innovations



## AI

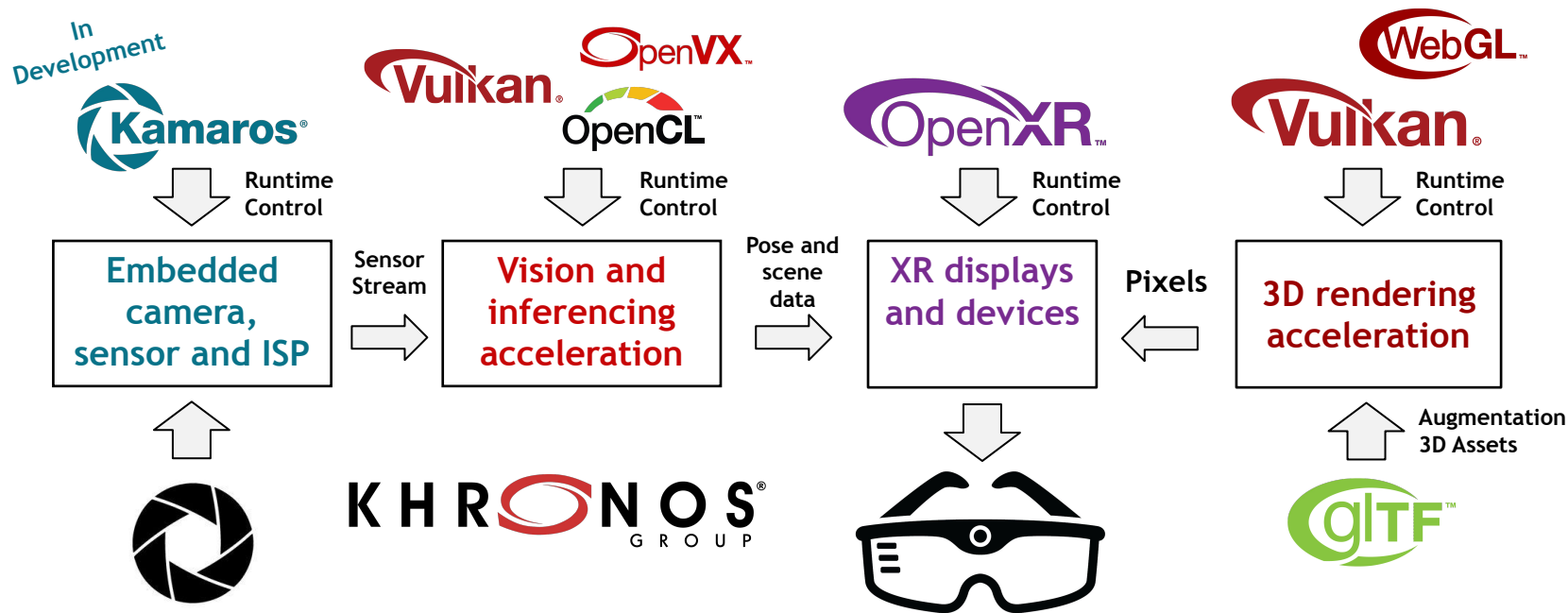
Sensor flexibility needed to feed inferencing AND traditional acceleration pipelines

## Efficiency and Low Latency

For real-time interactivity on power-constrained systems

# Spatial Computing APIs

Cameras are now the *only* part of the spatial computing pipeline not yet well-served by Khronos open API standards





## Embedded Camera System API - In Development

Open, cross-vendor, royalty-free open standard for camera, sensor and ISP control  
Intended for embedded, mobile, industrial, XR, automotive, and scientific markets

### Benefits for Integrators and Developers

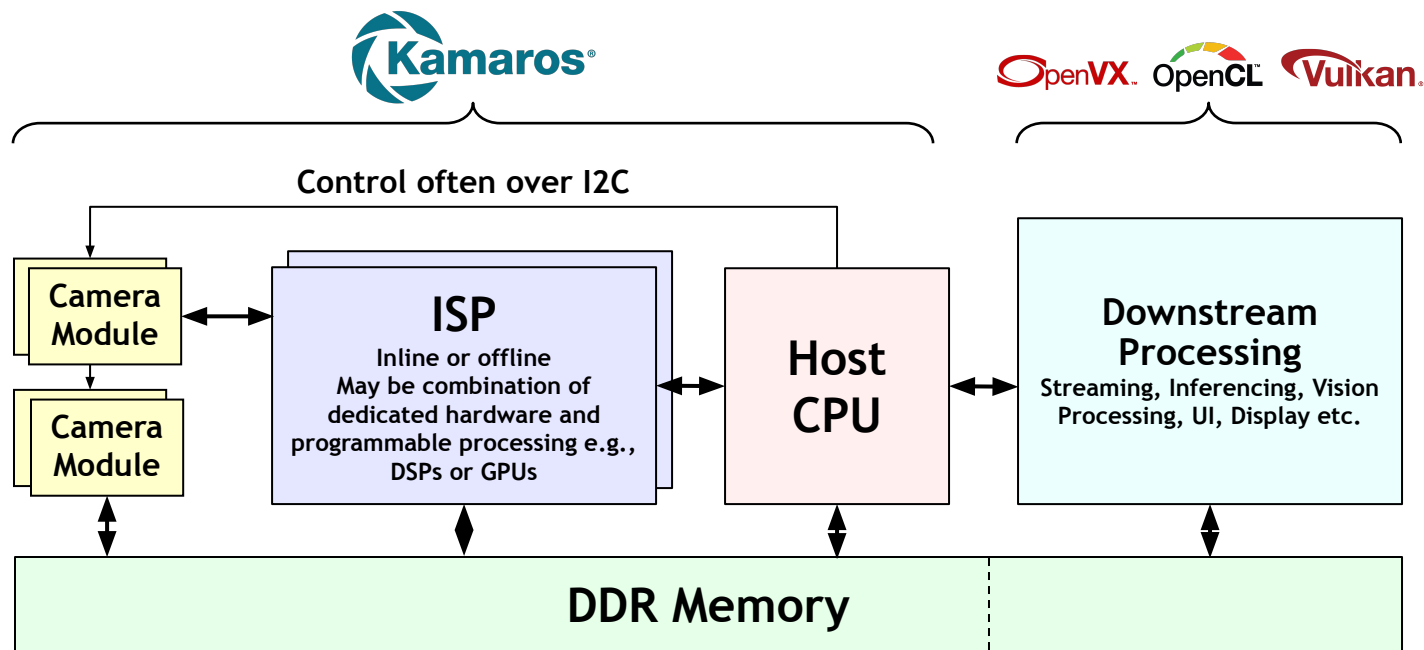
Portability of camera/sensor code for easier system integration of new sensors  
Preservation of application code across multiple generations of cameras and sensors  
Sophisticated sensor stream control for effective downstream inferencing and processing

### Benefits for Camera and Sensor Vendors

Enables vibrant software ecosystem to accelerate widespread use of sensor innovations  
Expose hardware innovations without disclosing proprietary implementation details

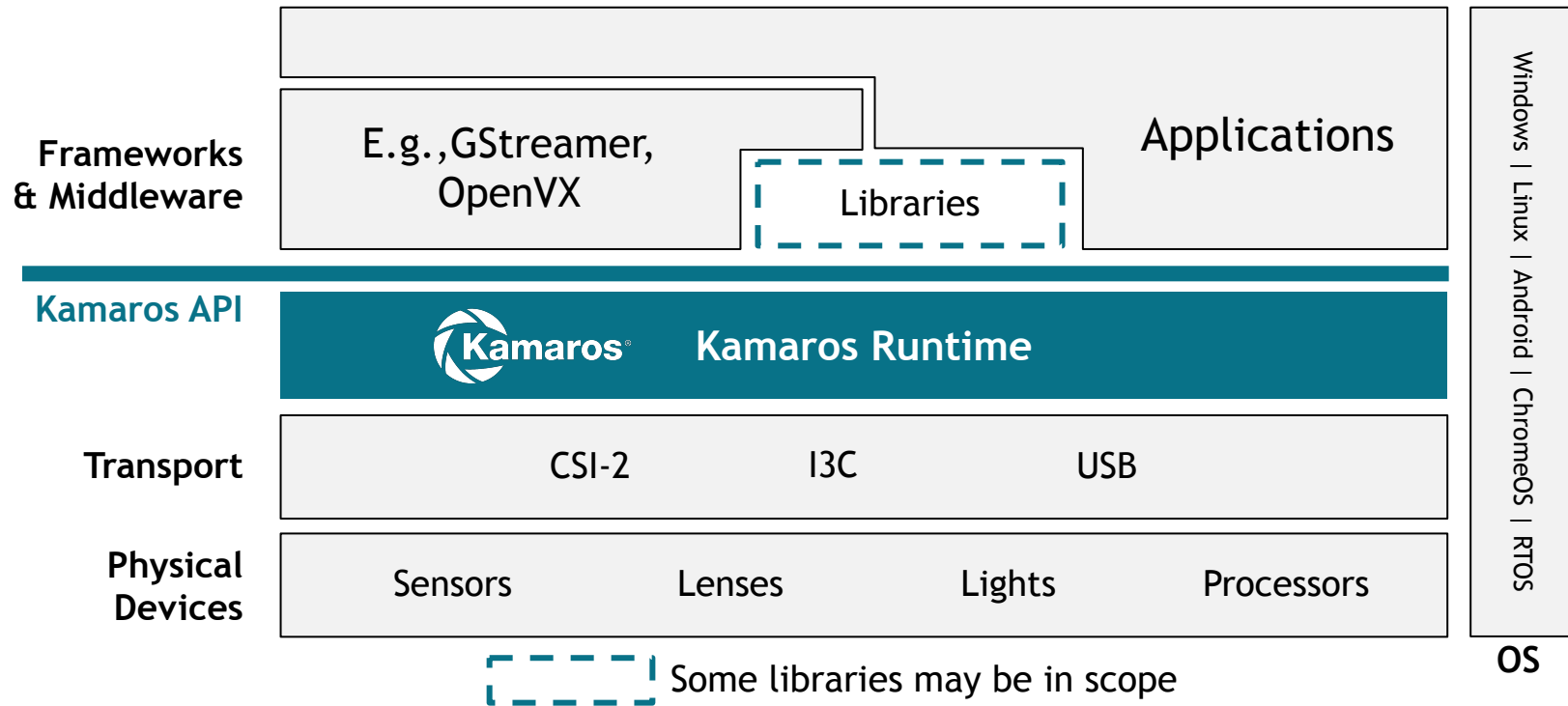
# Kamaros Scope

Kamaros API provides application facing controls for Camera Modules and close-to-sensor Image Signal Processing (ISP) hardware



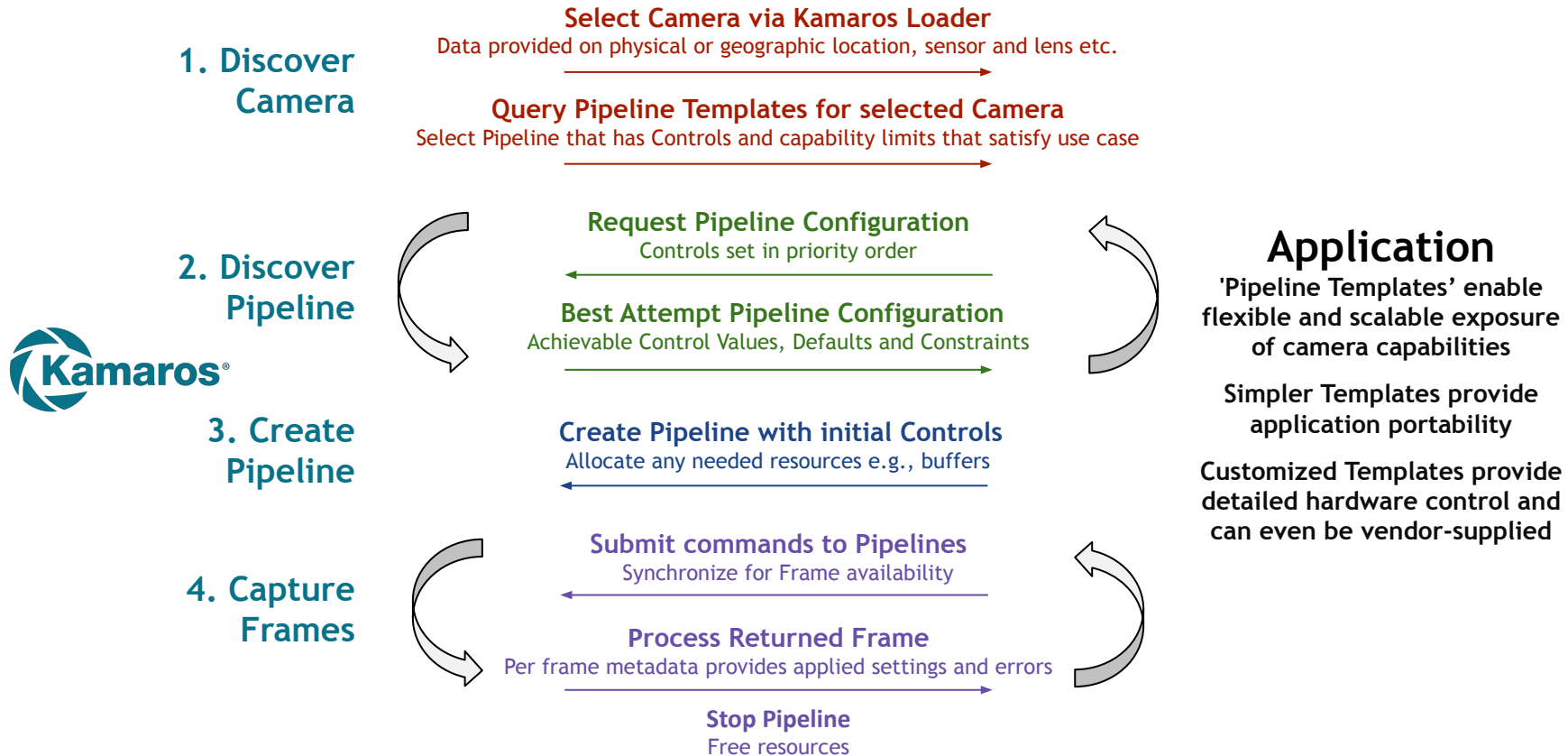


# Typical Kamaros Software Stack



*Names of transport layers, framework and operating systems are illustrative examples*

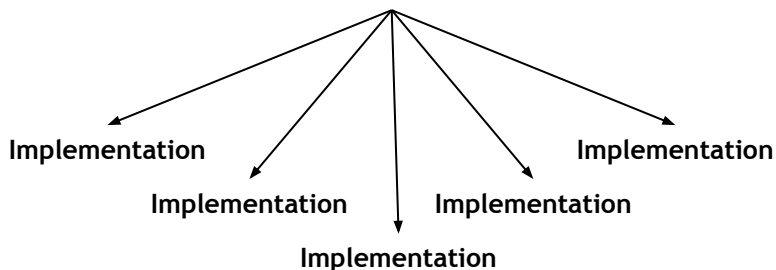
# Kamaros Portable Application Structure



# Openness Makes Technology Pervasive

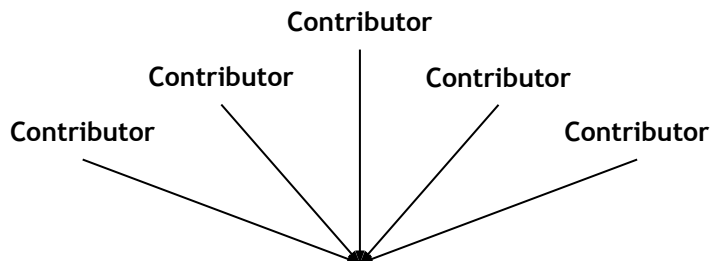
INTEROPERABILITY standards  
define precise COMMUNICATION  
E.g., software to hardware, client to server

**Open Standard =  
Shared Specification**



Rigorous conformance testing for consistency  
across **multiple implementations** to meet diverse  
market needs, price points, and use cases

Open Source projects enable sharing of  
implementation effort and expertise  
from multiple organizations



**Open Source =  
Shared Implementation**

Open implementations enable **rapid and consistent  
deployment** of functionality that doesn't benefit  
from multiple implementations

**Open Standards and Open Source are Complementary**

E.g., open standards often use open source for sample implementations,  
example applications, conformance tests, validators etc...

# Open Standards Drive Market Growth e.g., GPUs

- **Cross-vendor software access to hardware leads to strong technology adoption**
  - GPUs are now as significant as CPUs, or more
- **Well-designed API standards enable and encourage innovation**
  - Level of abstraction defines operation NOT implementation
  - Healthy market competition on features/performance/power/cost *without fragmentation*
- **Extensible APIs enable full speed, vendor-lead, use case exploration and expansion**
  - No permission needed to ship new extended functionality
  - 3D graphics -> ray tracing -> video/streaming -> neural graphics
- **Open API standards enable and encourage open-source software stacks**
  - Accelerates and eases implementation of powerful software capabilities
  - E.g., the open source Mesa 3D Graphics Library is available on multiple vendor GPUs



***A strong software ecosystem sells more hardware!***

# Sensor Market Opportunity and Challenges

- **Kamaros has a carefully chosen API abstraction: defines operation, not implementation**
  - No need for implementers to expose proprietary technology
  - FASTER software ecosystem access to and utilization of sensor innovations
- **Kamaros will be extensible using Khronos's proven extension process**
  - To rapidly track and evolve to handle sensor innovations
  - Vendors can experiment with custom extensions before collaborative standardization
- **Kamaros Pipeline Templates enable definition of tested, well-performing pipelines**
  - Reduce risk of sensor mis-configuration
- **Challenge and request for feedback**
  - Kamaros drivers will be often implemented and shipped by SOC vendors or system integrators
  - How should Kamaros support sensor integration?
  - What sensor controls are needed beyond exposure time, gain, cropping and scaling?



# MIPI & Khronos Standards

- MIPI and Khronos share a common industry goal
  - Streamline integration of cameras/sensors with ISPs across the industry
- Efforts of the two groups are complementary
  - MIPI - hardware interfaces (CSI-2) and low-level software control of cameras (CCS/CCI)
  - Khronos - higher level API for programming sensors, cameras and ISPs
- Potential collaborative liaison topics in discussion
  - Standardized Camera to ISP interoperability to reduce testing and calibration effort
  - Jointly agreed implementation guidelines, normative references

## Hardware Connectivity Standards

 **mipi**alliance  
CSI-2/CCS/CCI



Sensor Vendors  
Camera Vendors  
ISP Vendors




## Software API Standards

 **K H R O N O S**  
G R O U P  
 **Kamaros**

# Working Group Process

- Cooperative Design Wiki to capture design decision trail with associated reasoning
  - Hosted in Working Group GitLab
- Collaborative prototyping
  - Informs API design and early testing of API effectiveness
  - Foundation for open-source sample implementation in parallel with specification release

## Home



### How to use the wiki

Good starting points for navigation are collected to the side bar at right side of the browser window.

For instructions on how to edit content, see [GitLab documentation](#).

This wiki is backed by a Git repository, so it is also possible to create a local clone, do edits there and push back to Khronos Gitlab; this may be easier for large edits. Instructions for cloning can be found from the top line of right side bar.

#### Kamaros wiki home

##### API Design

- Kamaros Use Cases
- Application life cycle
  - Camera discovery
    - Pipeline Template API
    - Pipeline Controls
  - Sessions, pipelines & capture loop
    - Metadata
    - Error handling
  - Standard Pipeline Templates
    - Control Parameters
- Outdated pages (for reference)
  - Session
  - Capture loop

##### System Integration

- System Integration Overview
- Loader
- ABI & API definition
- Debug etc. layers
- Camera Module Integration
- Kamaros & Vulkan

##### Requirements

##### Glossary

##### General Issues

##### Planning

- Planning & status
- Workshops
  - Cambridge Sep 2024
  - Helsinki Jun 25-27, 2024
  - Brussels Feb 2024
- Prototyping

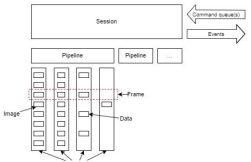
##### References

View all pages

## Camera Session and Pipelines

### Overview

Camera session is the top level container and interface for accessing a camera and controlling camera system in Kamaros. The following diagram shows a high level overview of a camera session.



Session contains one or more camera pipelines that process and produce frames. Pipeline has one or more inputs (image sensors and/or memory data streams) and zero or more outputs. Each data item received from input (or set of data items in case of multiple inputs) creates a frame. Each output may produce a data item from each frame. Data items produced from a single pipeline output are called **streams**.

Pipeline exposes a set of control points and associated controls that are used to control pipeline operations.

Processing and producing a frame is triggered by pipeline inputs and cannot (usually) be controlled by application; a typical imaging sensor captures images at constant (or variable) rate until stopped, and pipeline will produce a frame for each input image regardless of application actions. Pipeline sends events to application to inform it about frame capture and processing.

Pipeline is part of a session and its lifetime is equal to session's lifetime. Pipeline is created based on a pipeline template and pipeline configuration. At creation time session gets exclusive access to image sensors and/or other resources that are needed by its pipelines.

Multiple pipelines in a single session may have the same imaging sensor as an input. However, only one for these pipelines can be active at a time.

The application controls pipelines with **commands** it sends to the containing session via **command queues**. Commands can for example start or stop a pipeline or modify pipeline control values. Commands from a command queue are executed in the same order they are submitted to it. Each command is always executed in the context of a specific frame.

Unlike *Discovery* and *API*, Kamaros *does not* require application to submit a frame request for each frame. After a pipeline is started, it produces new frames until stopped. Kamaros allows application to explicitly control how individual frames in a frame sequence are produced (see *commands*) but if it does nothing, the session will continue producing new frames using the latest control values set by application.

#### Kamaros wiki home

##### API Design

- Kamaros Use Cases
- Application life cycle
  - Camera discovery
    - Pipeline Template API
    - Pipeline Controls
  - Sessions, pipelines & capture loop
    - Metadata
    - Error handling
  - Standard Pipeline Templates
    - Control Parameters
- Outdated pages (for reference)
  - Session
  - Capture loop

##### System Integration

- System Integration Overview
- Loader
- ABI & API definition
- Debug etc. layers
- Camera Module Integration
- Kamaros & Vulkan

##### Requirements

##### Glossary

##### General Issues

##### Planning

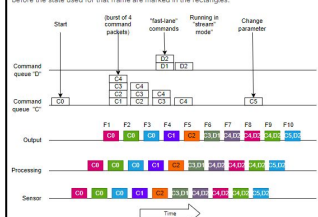
- Planning & status
- Workshops
  - Cambridge Sep 2024
  - Helsinki Jun 25-27, 2024
  - Brussels Feb 2024
- Prototyping

##### References

View all pages

## Command example

This example demonstrates the different cases of command submission and execution. The picture below shows a timeline of Kamaros application pipeline while a camera captures 10 frames. Application actions (command submissions) are shown in the top line; content of the two command queues on the 2nd and 3rd line. The bottommost lines show hypothetical Kamaros pipeline stages processing a frame. Activity related to each frame is shown in different color and the latest command executed before the state used for that frame are marked in the rectangles.



1. First the application submits `startPipeline` command to a queue and Kamaros starts to produce frames.
2. After a while, application wants to capture a burst of 4 frames using different settings for each (say, a bracketed exposure for some computational photography use case). It submits a sequence of 4 command packets to the command queue in a single API call, each packet containing parameters for one frame.
3. While the burst capture is still ongoing, application needs to change some other pipeline controls urgently (e.g. the subject has moved and it needs to change focus position). If it would submit a command to the same queue it used previously, the command would be executed after the whole burst has been captured. By submitting commands to another queue application can ensure that the commands are executed immediately.
4. After frame F7 all command queues are drained and the pipeline continues in "free running" mode using the control values of that frame.

#### Kamaros wiki home

##### API Design

- Kamaros Use Cases
- Application life cycle
  - Camera discovery
    - Pipeline Template API
    - Pipeline Controls
  - Sessions, pipelines & capture loop
    - Metadata
    - Error handling
  - Standard Pipeline Templates
    - Control Parameters
- Outdated pages (for reference)
  - Session
  - Capture loop

##### System Integration

- System Integration Overview
- Loader
- ABI & API definition
- Debug etc. layers
- Camera Module Integration
- Kamaros & Vulkan

##### Requirements

##### Glossary

##### General Issues

##### Planning

- Planning & status
- Workshops
  - Cambridge Sep 2024
  - Helsinki Jun 25-27, 2024
  - Brussels Feb 2024
- Prototyping

##### References

View all pages

# Working Group Deliverables

- **Enabling the Kamaros Ecosystem**
  - Needs much more than just a specification
- **Widespread and consistent implementations**
  - Precise API specification for use by implementers and developers
  - A conformant open-source sample implementation of the API
  - Open-source conformance test suite, including a precise definition of conformance
  - Adopters Program to enable implementations to become officially conformant
  - Trademark and logo for promotion and use on conformant implementations
- **Developer education**
  - Open-source samples and documentation
  - Open-source SDK, tools and Libraries
- **Extensibility to enable rapid innovation**
  - Central extension namespace registry for Working Group and vendor extensions

**Kamaros API will be openly available to the  
industry - royalty-free**  
**Under the Khronos Intellectual Property Rights Framework**





# Timeline

2021

## Camera Exploratory Group

In response to industry requests EMVA and Khronos cooperate to explore industry interest and build consensus on use cases and requirements

Over 70 companies join and contribute to the discussions

**KHRONOS**  
GROUP



2022

## Kamaros Working Group Created

Working Group formed under the Khronos membership and IP framework

Work starts on the detailed specification of the API

**KHRONOS**  
GROUP

2023-2025

## Kamaros Spec Drafting

Including cooperative prototyping using Linux on Raspberry Pi

**KHRONOS**  
GROUP

2026

## Kamaros 1.0 Release

Including open-source sample implementation and Conformance Test Suite

**KHRONOS**  
GROUP

### Two ways to get involved:

1. Join Khronos to directly influence API design and evolution
2. Join Kamaros Advisory Panel to review pre-release spec drafts (\$0)

# Call for Input and Participation

- Please help ensure Kamaros creates new opportunities for the sensor vendor community
  - What are the industry pain points that an open standard API can help solve?
  - What extensibility is needed to accommodate sensor technology roadmaps?
  - What software components need to cooperate/interoperate to enable sensor innovation?
  - How can Khronos best engage & cooperate with this community to address industry needs?
- Any company is welcome to join Khronos to directly influence API design and evolution
  - [www.khronos.org/members/](http://www.khronos.org/members/)
- Talk to us about joining the Kamaros Advisory Panel to review pre-release specifications
  - Email [memberservices@khronosgroup.org](mailto:memberservices@khronosgroup.org)
- More information on Kamaros and all Khronos APIs
  - <https://www.khronos.org/>

