ANARI 1.0 Launch
The Industry’s First Open Standard, Cross-platform 3D Rendering Engine API
Press Pre-Briefing
August 2023
Visualization, Rendering and ANARI

Many new 3D rendering technologies are available to scientific visualization applications. Techniques such as path tracing provide significant visualization enhancements.

BUT can be complex and time-consuming for domain experts to use low-level rendering APIs. Rendering engines can hide that complexity - and a rich diversity of vendor and open-source rendering engines are now available - BUT every rendering engine uses a different API.

Cross-Platform 3D Rendering Engine API
Simplified application development
Application portability to any engine supporting ANARI
ANARI 1.0 Launch

Simplified Application Development
High-level API to describe WHAT is to be rendered not HOW

Application Portability
Common API for ANY rendering-engine independent of vendor, platform or ecosystem

Cross-Platform 3D Rendering Engine API

ANARI 1.0 Finalized
Multiple implementations shipping and open-source SDK available

Scientific Visualization Beachhead
Many types of application will benefit from ANARI
Scientific Visualization Before and After ANARI

Before ANARI

<table>
<thead>
<tr>
<th>VTK</th>
<th>ParaView</th>
<th>VMD</th>
<th>VisIt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel OSPRay</td>
<td>AMD Radeon ProRender</td>
<td>NVIDIA VisRTX</td>
<td>Others</td>
</tr>
</tbody>
</table>

After ANARI

<table>
<thead>
<tr>
<th>VTK</th>
<th>ParaView</th>
<th>VMD</th>
<th>VisIt</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANARI</td>
<td>Intel OSPRay</td>
<td>AMD Radeon ProRender</td>
<td>NVIDIA VisRTX</td>
</tr>
</tbody>
</table>

ANARI applications are portable to any engine supporting the ANARI API
Independently of vendor, platform or ecosystem
ANARI Development Stack

Processing to construct a scene description with application-specific structures, traversals, and metadata

ANARI API used to build in-memory scene representation
NO rendering details prescribed
C99 frontend API dispatch library with C++ type-safe wrappers
Extensible API design with installable development layers
Asynchronous scene updates for low-latency interactivity

Engines use in-memory scene representation to drive rendering operations

Explicit control over hardware resources and operations

Scene Graphs

Apps and Engines

ANARI

Rendering Engines
Intel OSPRay, Radeon ProRender, NVIDIA VisRTX etc.

Acceleration APIs
Vulkan, OpenGL, DX12, Metal
Embree, OptiX, Radeon Rays, CUDA, OpenCL, etc.

Hardware
CPUs, GPUs etc.

VMD Rendering using ANARI
Satellite Tobacco Mosaic Virus
1M atoms, U. Illinois
ANARI and Scientific Visualization Apps

**VisIt**
Visit Renderings with ANARI

**VMD**
VMD Rendering with ANARI using NVIDIA ANARI-USD and Omniverse

**ParaView**
ParaView visualization rendered with ANARI OSPRay backend

**VTK**
VTK-m for Real Time Filtering + Rendering with ANARI

ANARI integration with key open-source visualization applications
ANARI Scene Representation

Hierarchical object tree that expresses the complete scene for a single frame

Sections of the tree can be re-used to optimize resource utilization

Scene representation can be used to drive any rendering backend - rasterization techniques are NOT prescribed
ANARI Timeline

Open-source SDK includes Conformance Test code

- SDK v0.1.0 March 2022
- SDK v0.2.0 July 2022
- SDK v0.3.0 Feb 2023
- SDK v0.7.0 August 2023

Working Group Launched March 2020

V1.0 Provisional Specification November 2021

Industry Input and Feedback
- glTF-compatible Physically-based materials
- Improved object interface and error handling
- Directly mapped array parameters
- Revamped runtime feature queries
- Improved volume shading

V1.0 Final Specification August 2023

Adopters Program v1.0

- Adopters Agreement
- Testing Process
- Conformance Tests

3Q23

ANARI 1.0 Launch

Industry Input and Feedback

All specification, SDK and Conformance Test development work in publicly accessible GitHub
ANARI SDK

- Rendering engine backend layers
  - Adopters can fully implement the API or use convenience layers that implement common functionality such as handling parameters or object lifetime

- Loadable debug and trace layers
  - Debugging layer for application API stream validation
  - Trace layer for API call tracing + replay

- Conformance Test Suite based on Python
  - Used in ANARI Adopters Program

- ‘Helide’ sample implementation
  - Demonstrate possible API implementations choices
  - Shows how adopters can integrate with the SDK

- Example applications demonstrating ANARI concepts
  - Including simple interactive viewer

- ...and much more to come!

https://github.com/KhronosGroup/ANARI-SDK
Implementations Shipping Today

- Khronos ‘Helide’ open-source sample implementation
  - Ships with ANARI SDK
- AMD Radeon ProRender
  - https://github.com/GPUOpen-LibrariesAndSDKs/RadeonProRenderANARI
- Intel OSPRay
  - https://github.com/ospray/anari-ospray
- NVIDIA VisRTX + VisGL
  - https://github.com/NVIDIA/VisRTX
- NVIDIA Omniverse
  - https://github.com/NVIDIA-Omniverse/AnariUsdDevice
- All implementations expected to be officially conformant
  - When ANARI 1.0 Adopters Program released
ANARI Beyond Scientific Visualization

Proof-of-concept Blender Add-On
Amazon Lumberyard Bistro
NVIDIA Open Research Content Archive (ORCA) 2017

ANARI-USD Brings ANARI applications to USD/Omniverse
NVIDIA OmniGraph geometry processing
ANARI Beyond Scientific Visualization

ANARI with NVIDIA VisRTX Backend. San Miguel Scene © Guillermo M. Leal Llaguno
Get Involved!

Use ANARI in YOUR application
Multiple implementations and SDK shipping today

Send us your feedback and requirements on GitHub
What rendering features important to your application domain?
In what new application domains and use cases would you use ANARI

Join Khronos and the ANARI Working Group
Have a voice and a vote in the design of the ANARI specification
Fast track ANARI for your renderer or hardware

https://www.khronos.org/anari
https://github.com/KhronosGroup/ANARI-Docs
https://github.com/KhronosGroup/ANARI-SDK