What’s Happening Inside the ANARI Working Group and How YOU Can Get Involved!

August 2020
Welcome!

This slide deck will be posted at [www.khronos.org](http://www.khronos.org) today.

The webinar will be recorded and made freely available.

Please take a moment to answer the polls.

Ask any questions via the Q&A Panel at any time.

We will answer questions at the end of the webinar.
Introduction to Khronos
Neil Trevett
Khronos President

ANARI Working Group Overview
Peter Messmer
ANARI Working Group Chair
Sr. Mgr HPC Visualization, NVIDIA

A Glimpse at the Evolving ANARI API
Jefferson Amstutz
Software Engineer, Intel

How to Use the ANARI API
John Stone
Senior Research Programmer
University of Illinois
Khronos Connects Software to Silicon

Open interoperability standards to enable software to effectively harness the power of 3D and multiprocessor acceleration

<table>
<thead>
<tr>
<th>Khronos Group</th>
<th>AMD</th>
<th>NVIDIA</th>
<th>Apple</th>
<th>epic</th>
<th>arm</th>
<th>Qualcomm</th>
<th>Samsung</th>
<th>Sony</th>
<th>Valve</th>
<th>IKEA Communications AB</th>
</tr>
</thead>
</table>

Over 150 members worldwide
Any company is welcome to join

3D graphics, XR, parallel programming, vision acceleration and machine learning

Non-profit, member-driven standards-defining industry consortium

Open to any interested company

All Khronos standards are royalty-free

Well-defined IP Framework protects participant’s intellectual property

Founded in 2000

>150 Members ~ 40% US, 30% Europe, 30% Asia
Guidelines for creating APIs to streamline system safety certification
Khronos for Global Industry Collaboration

Khronos membership is open to any company
Influence the design and direction of key open standards that will drive your business
Accelerate time-to-market with early access to specification drafts
Provide industry thought leadership and gain insights into industry trends and directions
Benefit from Adopter discounts
www.khronos.org/members/
ANARI Working Group Overview

Peter Messmer
ANARI Working Group Chair
Senior Manager HPC Visualization, NVIDIA
Visualization ≠ Rendering
Industry Need and Opportunity

- New rendering technology - including ray tracing - is impacting scientific visualization
  - Accurate generation of imagery
  - Sophisticated visual cues provide intuitive understanding of complex data

- But building efficient, portable renderer using low-level APIs - such as Vulkan - is often outside an ISV’s mission
  - Rendering is just a necessary technique to be utilized
  - True for scientific visualization and emerging data analytics space

- Define a high level API to simplify scientific visualization applications
  - Leveraging full potential of modern rendering techniques
  - Platform independent
  - Portable code
ANARI (Analytic Rendering Interface) Goals

Analytic Rendering Portability
Common API to describe objects in a scene
API to build the description of a scene rather than specifying the details of the rendering process
Rendering details left to the implementation of the API
Subset of more general scene graph APIs
Strong ANARI Industry Support

All Layers in the Scientific Visualization Stack are Represented
GPU Vendors, Rendering Engines, Visualization Libraries and Tools, Applications
ANARI Timeline

Analytic Rendering Exploratory Group Formed
November 2019

ANARI Working Group Announced
3rd March 2020

TARGET ANARI Provisional Specification
Late 2020

TARGET First ANARI Specification
Mid 2021
Progress So Far

Defined and prioritized use cases

Investigated existing APIs
Selected starting-point API

Exploratory implementations
Different types of backends
Multiple frontend apps

Identifying friction points
Exploring issues requiring clarification
Discussing API changes

Starting to write specs
Not too late to get involved to influence the API design!
A Glimpse at the Evolving ANARI API

Jefferson Amstutz
Software Engineer
Intel
ANARI API Overview

- Viewport rendering API, not a scene graph
  - Avoid complexity of application-specific structure, traversals, and metadata APIs
  - Very hard to unify
- Unidirectional: data flows from the app to ANARI
  - Freedom for implementers to deliver more diverse back ends (e.g. distributed rendering)
  - Maximizes freedom to define internal non-rendering APIs with itself (above ANARI API)
- Diverse execution topologies (examples):
API Design

- C99 based API
  - optional C++ bindings for added convenience, type safety
- Based around opaque object handles
  - Named types (e.g. “triangle” and “sphere” geometry types)
  - Named parameters (e.g. “intensity” on lights)
- Everything revolves around a single API call...

```c
// Render a frame (non-blocking), return a future to the task executed by ANARI ANARIFuture anariRenderFrame(ANARIDevice, ANARIFrameBuffer, ANARIRenderer, ANARICamera, ANARIWorld);
```
Object Parameters

- Name-value pairs for parameters
  - Unused parameters by an implementation can be ignored (e.g. parameter extensions)
  - C++ bindings recover convenience/robustness from type information without losing C API flexibility

```c
// Set a parameter, where 'mem' points the address of the type specified
void anariSetParam(ANARIDevice,
                   ANARIOBJECT, obj,
                   const char *id,
                   ANARIDataType type,
                   const void *mem);

glm::vec3 cam_pos(0, 0, 1);
// ...
anariSetParam(device, camera, "position", ANARI_VEC3F, &cam_pos);
```
Object Types

- **Device (OSPRay, NVGL, ANARX, RadeonProRender, etc.)**
  - Encapsulates everything provided by a cohesive implementation
  - A “software device”, underlying hardware usage is implementation specific
- **Buffers + Arrays (buffers for memory ownership, arrays to apply type info)**
- **Frame Buffer (color, accumulation, depth, variance, etc.)**
- **Renderer**
  - Represents shading/lighting models offered by the device
- **Camera**
- **Image Operation (e.g. tone mapping, denoising, SSAO)**
- **Material (fixed material set and/or a front-end interface to OSL/MaterialX/MDL/etc.)**
- **World [...]**
  - All the objects to be rendered!
World Hierarchy

- World
  - Instance
    - Geometric Model
      - Geometry
    - Light
    - Volumetric Model
      - Volume
World Hierarchy

- World
- Instance
  - Geometric Model
    - Geometry +surface
  - Light +scalar field
  - Volumetric Model +Volume
World Hierarchy

- **World**
  - **Instance**
    - **Geometric Model**
      - **Geometry**
        - 1:1 +appearance
    - **Light**
      - +appearance
    - **Volumetric Model**
      - **Volume**
        - 1:1
World Hierarchy

- World
  - Instance
    - Geometric Model
      - Geometry
    - Light
      - 1:N [ ] +illumination
    - Volumetric Model
      - Volume
World Hierarchy

- World
- Instance
  - Geometric Model
    - Geometry
  - Light
  - Volumetric Model
    - Volume

[Diagram showing the hierarchy with '1:N' relationship between World and Instance]
World Hierarchy

- **World**
  - [ ] \(1:N\)
  - +world transform

- **Instance**
  - \(1:N\)
  - +illumination

- **Geometric Model**
  - \(1:N\)

- **Light**

- **Volumetric Model**

- **Geometry**
  - \(1:1\)
  - +appearance
  - +surface

- **Volume**
  - \(1:1\)
  - +appearance
  - +scalar field
Implementation Status

- Current approach is “implement first, then write the spec”
  - Sharing a common ‘libanari’ front-end library
  - Multiple ray tracing implementations + OpenGL
  - More on the way!
How to Use the ANARI API

John Stone
Senior Research Programmer
University of Illinois
ANARI Benefits for Applications and Middleware

- Designed with scientific and technical visualization needs foremost
- Easy developer adoption on-ramp
  - One API, many conformant implementations
  - Abstracts + encapsulates rendering details
  - API tracing, debugging, replay tools
- Use state-of-the-art renderers and rendering techniques, low application development cost
  - Choose renderers and rendering techniques best suited to application requirements
  - Exploit hardware-optimized renderers
- Middleware opportunities
  - Domain-specific features, data handling and translation, interoperability with other APIs

Swine Flu A/H1N1 neuraminidase bound to Tamiflu (VMD)
Scientific and Technical Visualization Needs

• High-fidelity rendering of analytical scene content
  - Geometric and volumetric objects, time series, ...
  - Complex materials and texturing, transfer functions
  - Text labels and legends
  - High quality lighting, shadows, AO, path tracing, ...

• Interactive, in-situ, and offline rendering of huge visualizations, massive memory-resident data

• Hardware platforms
  - CPUs, GPU accelerators, heterogeneous HPC nodes
  - Run on laptops up to desktop workstations
  - Distributed memory parallel computing on HPC clusters and supercomputers

On skin microrelief and emergence of expression micro-wrinkles.
Georges Limbert, U. Southampton (ParaView)
Runtime Enumeration of ANARI Renderers, Features

- ANARI plans include runtime query of renderers and features
  - Enable “matchmaking” of app requirements to renderers
  - Feature profiles to simplify app-renderer matching
  - Release of new renderers can augment existing shrink wrapped ANARI apps
  - Enable dynamic population of app graphical and scripting interfaces
  - Query availability of renderer-specific extensions
Exploit State-of-the-art Renderers + Techniques

- ANARI: Applications and middleware can make use of sophisticated rendering techniques simply by “turning them on”
- Advanced rendering techniques improve perception of complex 3-D geometry and spatial relationships within a visualization

Classic simple lighting...

Shadows, ambient occlusion, and directional lights

Satellite Tobacco Mosaic Virus Capsid (VMD)
Molecular Visualization Examples (VMD)
by Niklas Röber DKRZ

Cloud resolving climate models

ParaView 5.1
Publication Quality Journal Figures, Technical Communication, Movies, and Public Outreach (VMD)

Faraday Discussions
Volume: 169
Molecular Simulations and Visualization

KLAUS SCHULTEN MEMORIAL ISSUE
Rasterization (NCSA Virtual Director, Left)
Ray Tracing (VMD, Right)
Interactive, In-situ, and Offline Rendering of Huge Visualizations

- Interactivity crucial for exploratory visualization and for many technical and scientific modeling tasks
  - Direct coupling of application with hardware optimized renderers
  - Low overhead APIs
  - Async APIs overlap compute+viz
  - Exploit parallel computing techniques
- In-situ, visualization of massive memory-resident data sets
  - Render live in-memory app data
  - Use data in-place when possible

~1 billion atom protocell membrane w/ ~1400 proteins (VMD)
by Niklas Röber at https://www.dkrz.de/ of the Agulhas Current
Get Involved in Developing ANARI!

Join Khronos and the ANARI Working Group
Have a voice and a vote in the design of the ANARI specification
Gain early access to alpha release APIs, tools, renderers
Fast track ANARI for your own renderer or hardware

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

https://www.khronos.org/anari
anari-chair@lists.khronos.org