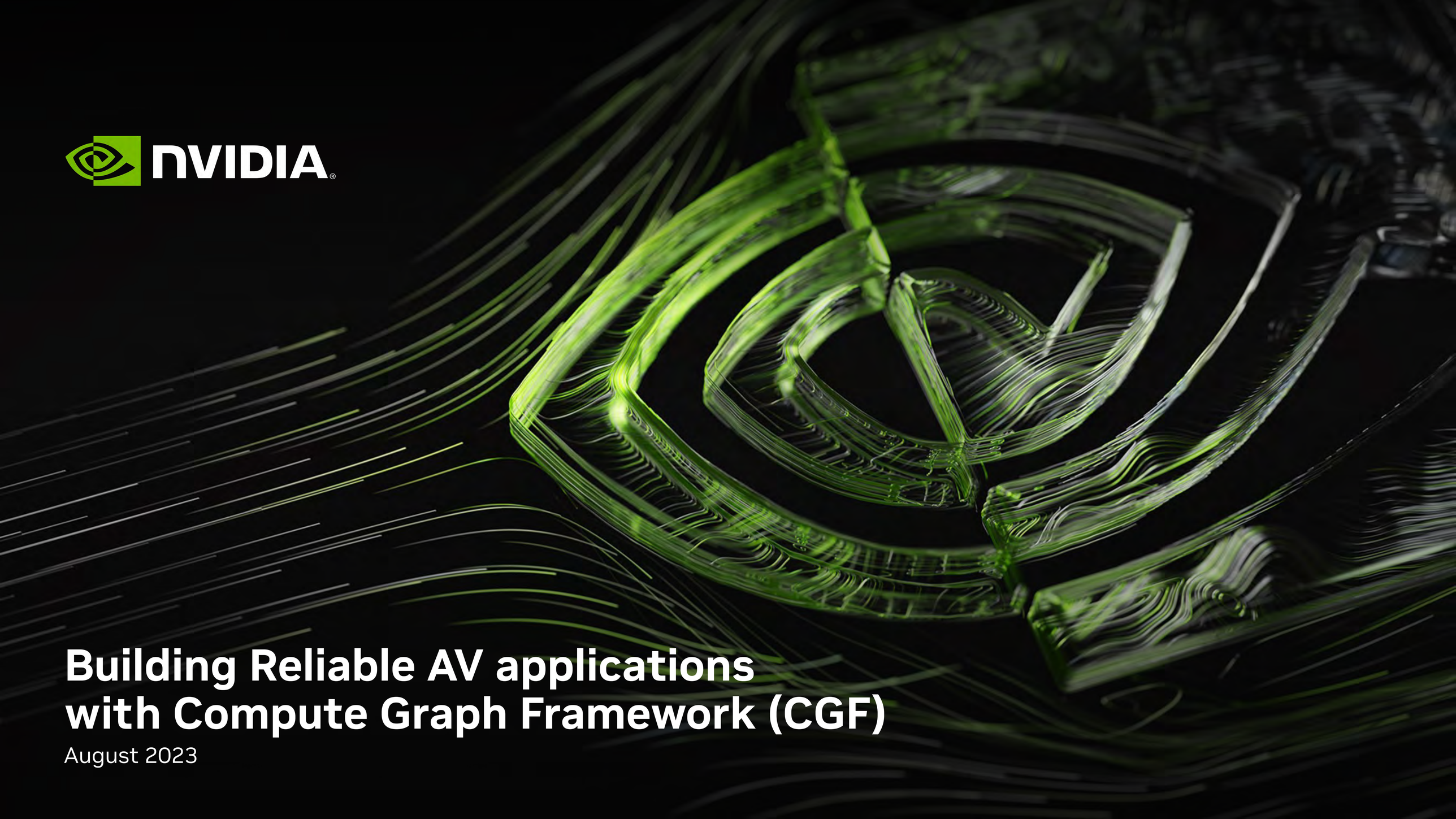
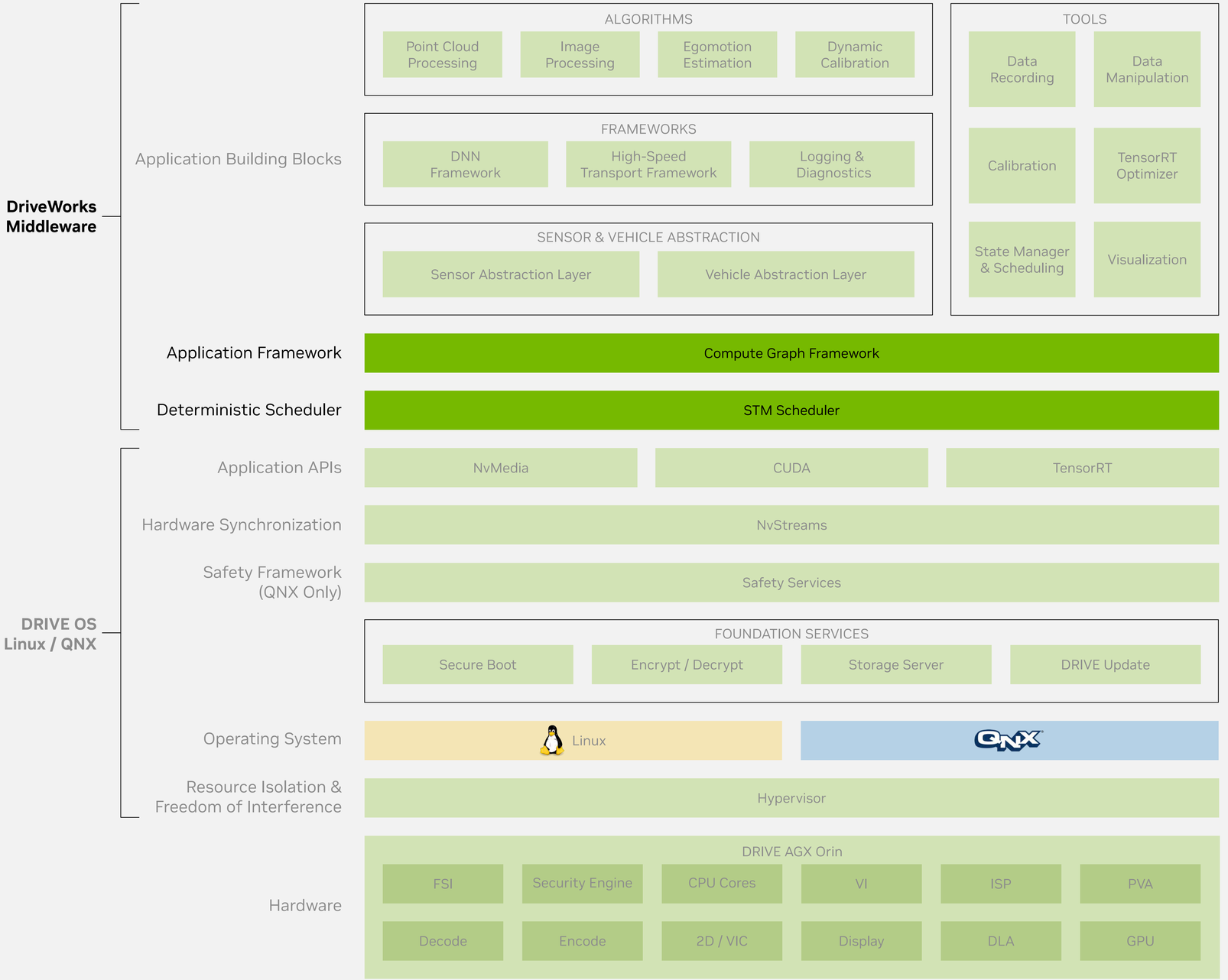


Building Reliable AV applications with Compute Graph Framework (CGF)

August 2023

DRIVE SDK Supercharges AV Development

One architecture
DRIVE OS & DriveWorks
DRIVE AGX Orin

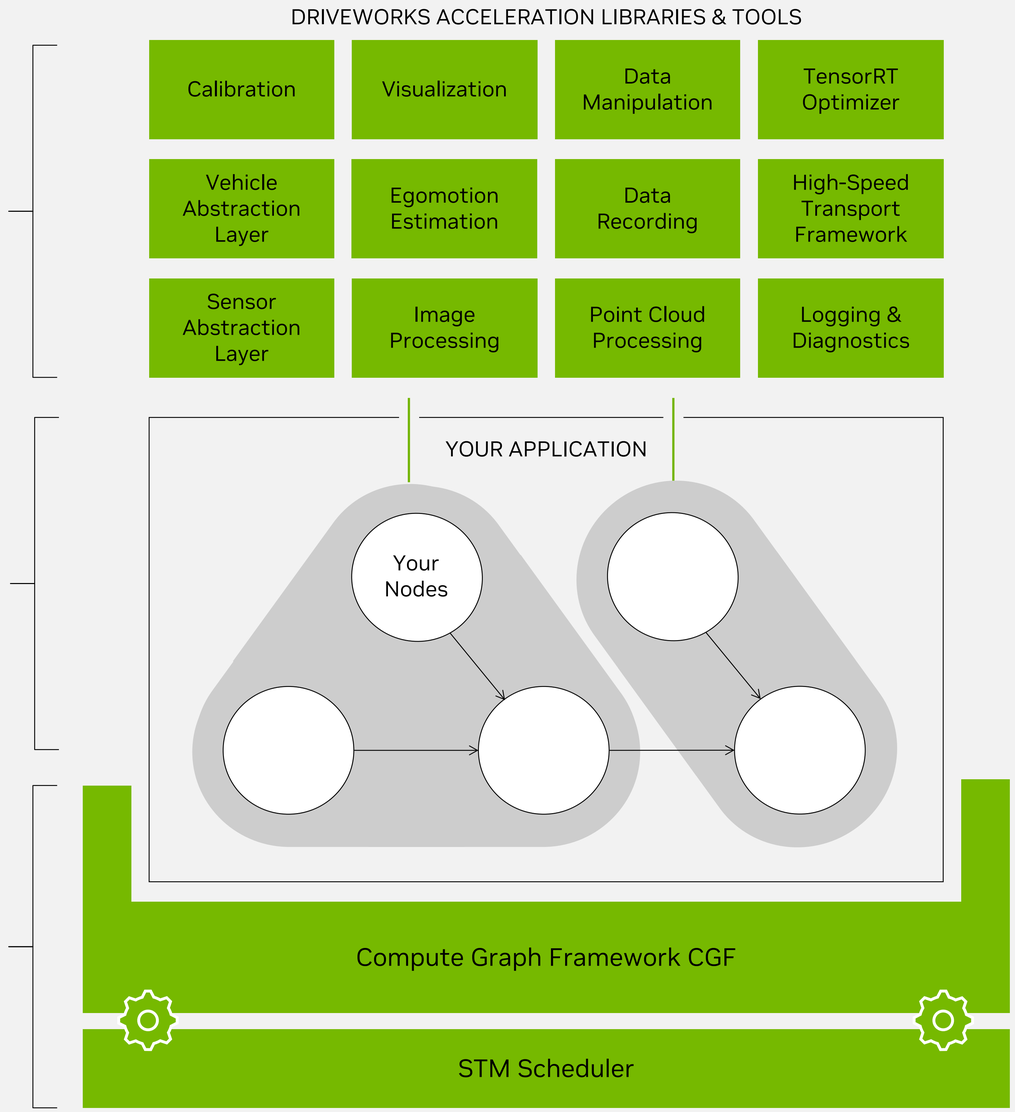


DriveWorks — Comprehensive Middleware Solution

Rich Library of Algorithms and Tools
to accelerate your applications

Compute Graph Framework
to simplify complex computational pipelines

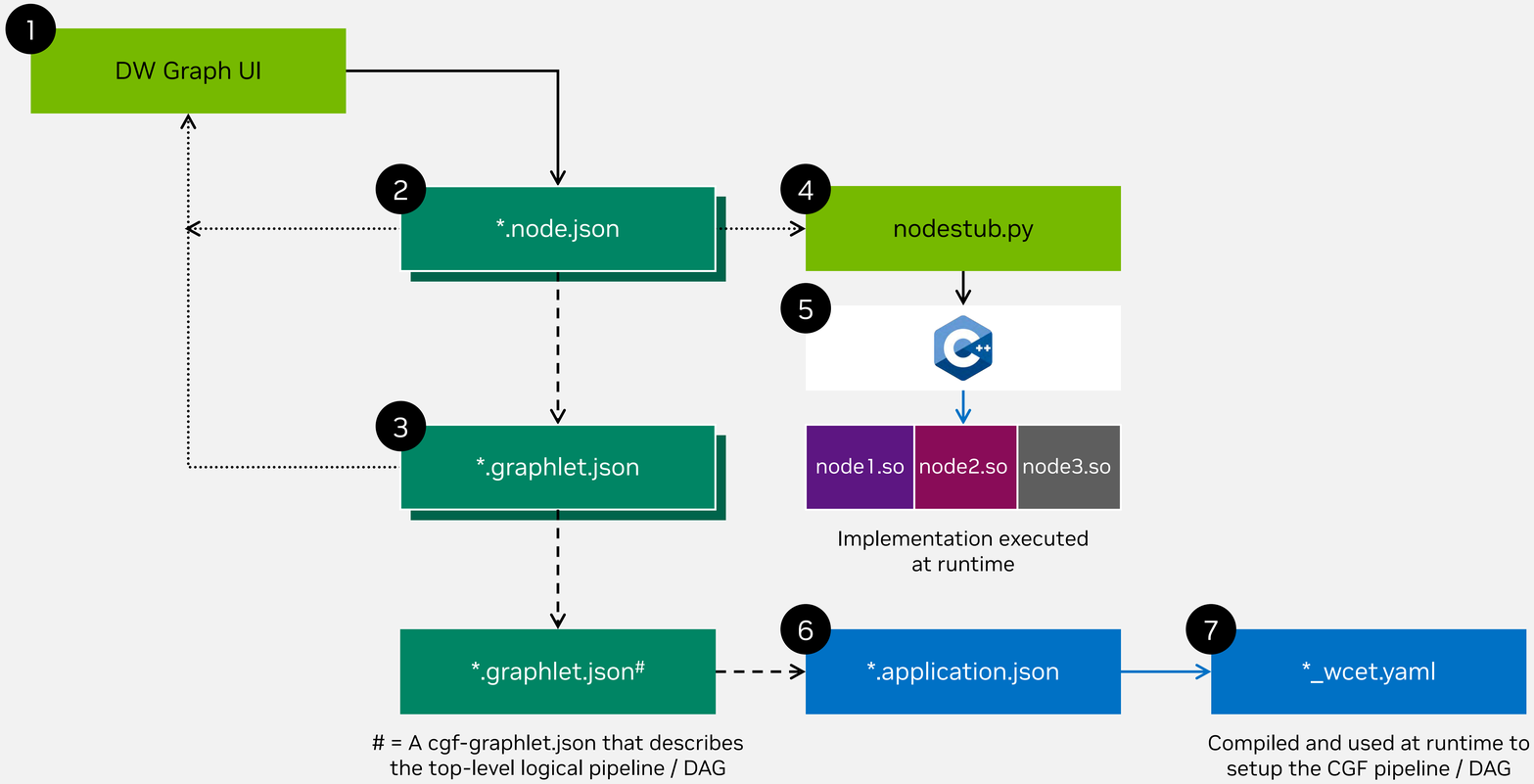
System Task Manager
to deterministically schedule application tasks



Agenda

- CGF Workflow at Design and Development Time
- CGF Execution Flow at Runtime
- What is a node
- Typical pipeline design and development workflow
- DW Graph UI — Introduction to Nodes and Graphlets
- Nodestub — Autogenerating Code for Custom Nodes
- Adding Custom Logic Within Autogenerated Code
- Introduction to an Application (CGF Demo)
- stmvizschedule — Static Visualization
- Profiling — NSight Systems + NVTX
- Overall Safety Concept

CGF Workflow at Design / Development Time



X ——→ Y = X Generates Y

X ·····→ Y = X is Used by Y

X - - - - - → Y = X is Referenced in Y

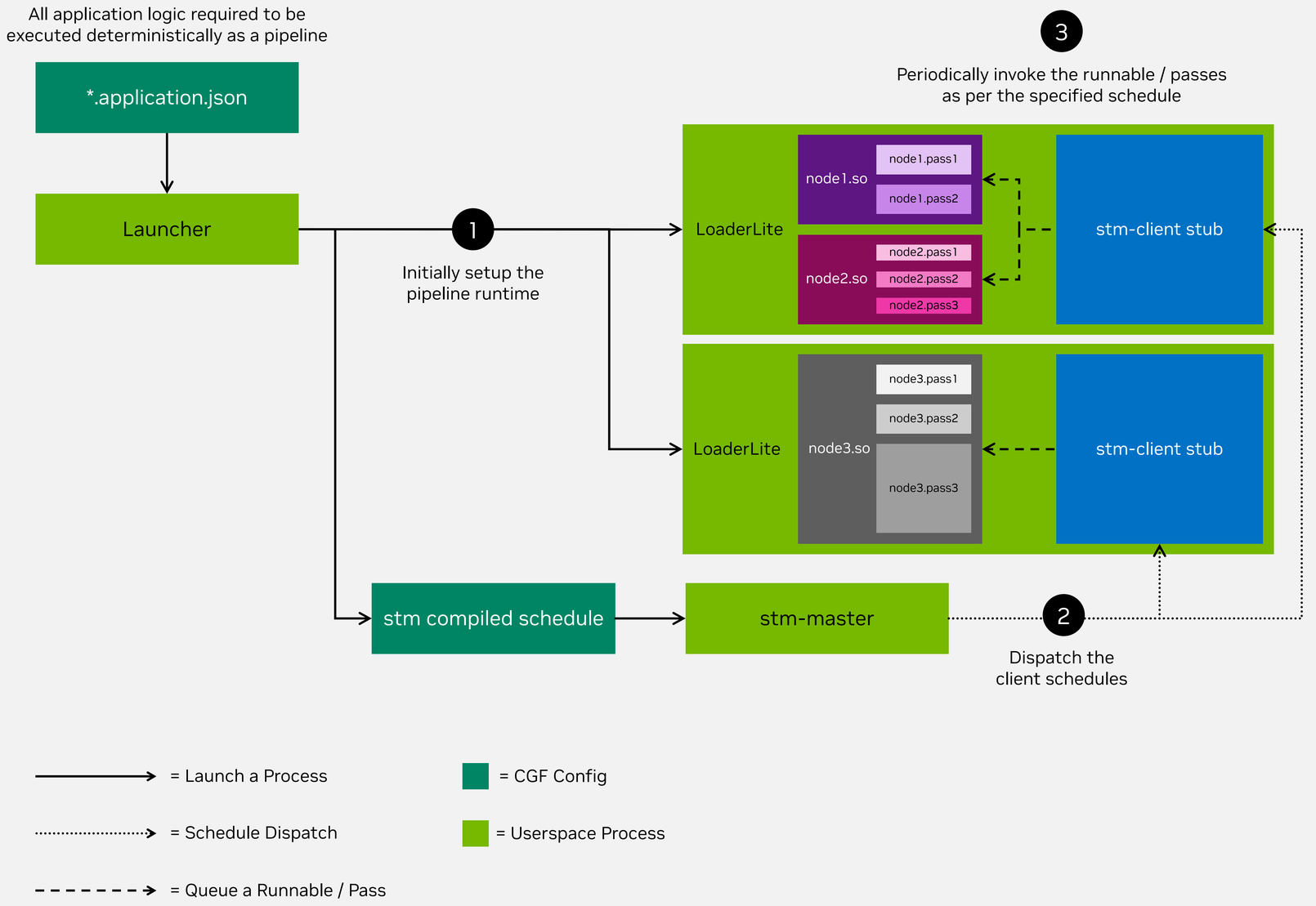
X ——→ Y = Output Y used at runtime

= CGF Tool

= CGF Config (tool-assisted)

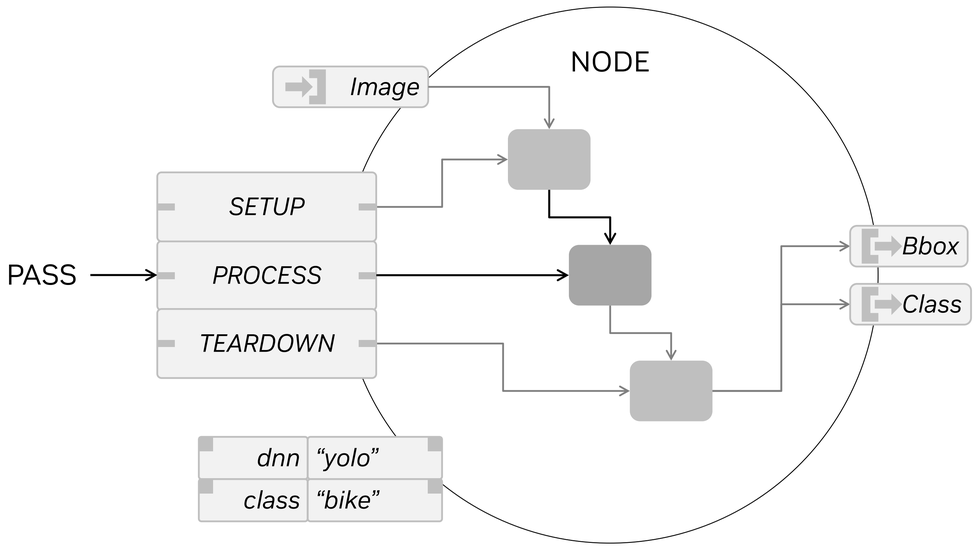
= CGF Config (manual)

CGF Execution Flow at Runtime



What is a CGF NODE

Key components and implementation



Node
Inheriting from base Node

Constructor receives parameters

Create function registers node

Ports
Inputs/Output ports specifying data type and unique port name

Passes
Passes to be executed in order, on specified compute engine

Parameters
Parameters for the constructor

Ports
Initialize ports, prepare output data containers

Passes
Register methods used for passes defined in public interface

Implementation of the passes

Public Node Interface `MyNode.hpp`

```
class MyNode : public dw::framework::ExceptionSafeProcessNode
{
   MyNode(const MyNodeParams& params, ...);

   dw::framework::create<Node>(ParameterProvider& provider);

   static constexpr auto describeInputPorts(){...};
   static constexpr auto describeOutputPorts(){...};

   static constexpr auto describePasses(){
      return dw::framework::describePassCollection(
         dw::framework::describePass("SETUP", DW_PROCESSOR_TYPE_CPU),...)};

   static constexpr auto describeParameters(){...};

}
```

Implementation `MyNode.cpp`

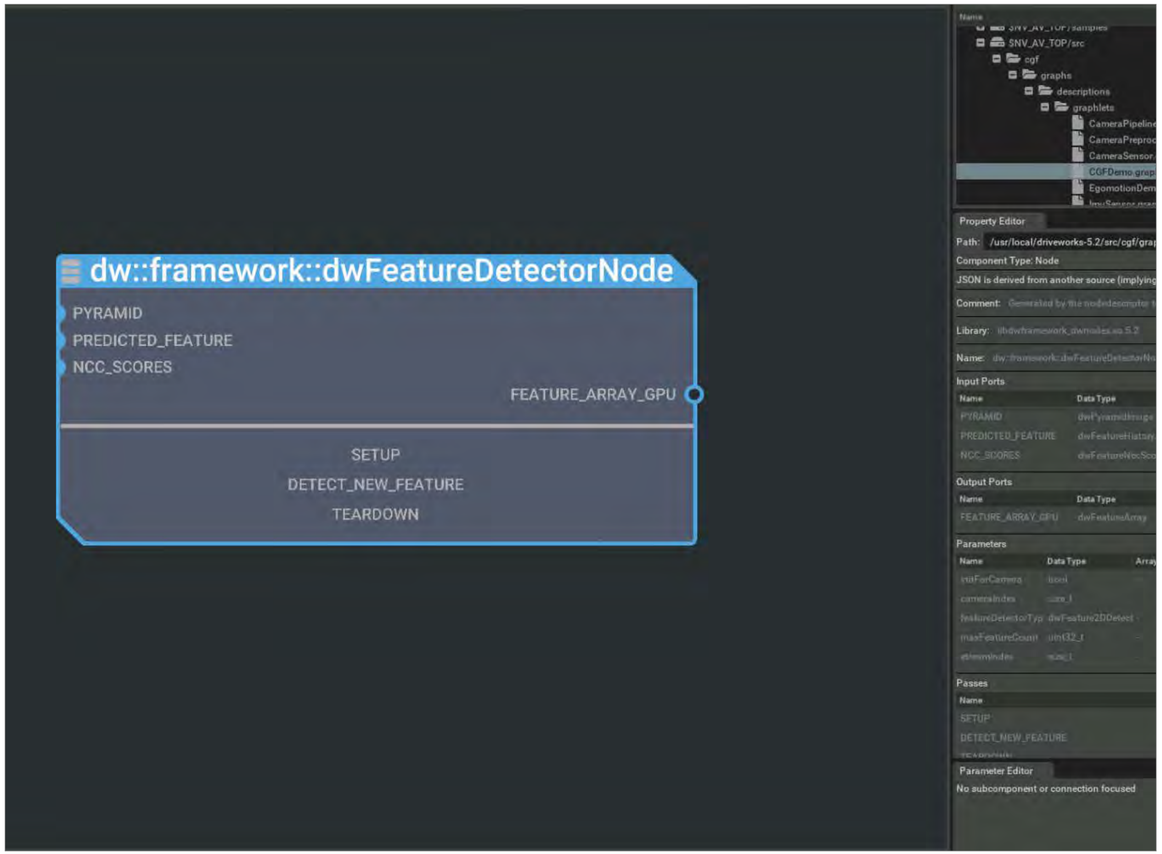
```
void MyNodeImpl:initPorts(){
   NODE_INIT_INPUT_PORT("IMAGE"_sv);
   ...
}

void MyNodeImpl::initPasses(){
   NODE_REGISTER_PASS("PROCESS"_sv, [this]() { return process(); });...}

dwStatus MyNodeImpl::process(){...}
```

Typical Pipeline Design and Development Workflow

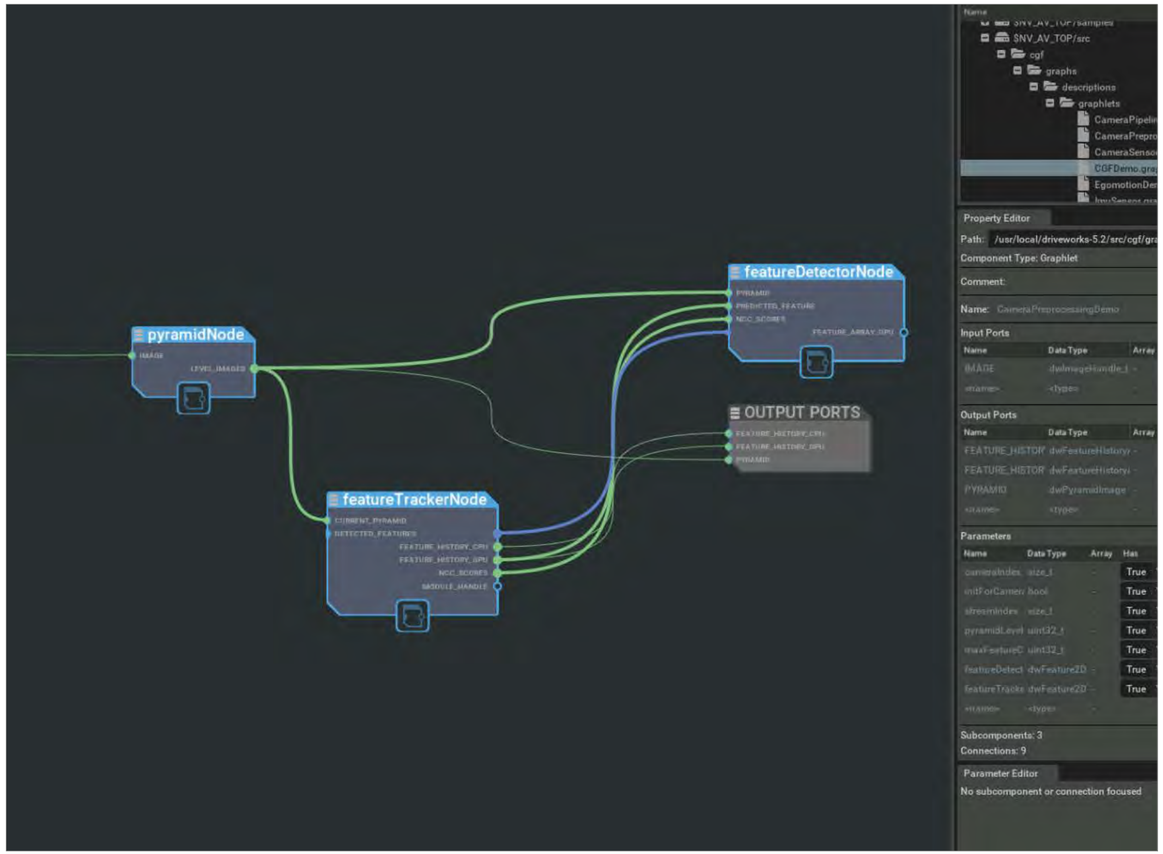
1. Create logical nodes
2. Create graphlets using the nodes
   - Specifying the data dependencies
3. **Generate stub / template code**
4. **Add the actual logic at appropriate places within the stub code**
5. **Compile and generate executable library**
6. Specify an application
   a. Process layout
   b. Distribution across execution engines (CPU, GPU, …)
   c. Timing Constraints (WCET)
7. Compile a schedule that satisfies the above specified constraints (data-dependency and timing)
8. Execute the application logic on the target
9. Profile. Tune. Repeat.

NOTE: Even without **writing/compiling any code**, one can specify nodes, graphlets, schedule constraints, WCET constraints and generate a schedule, visualize it, and draw conclusions about expected performance.

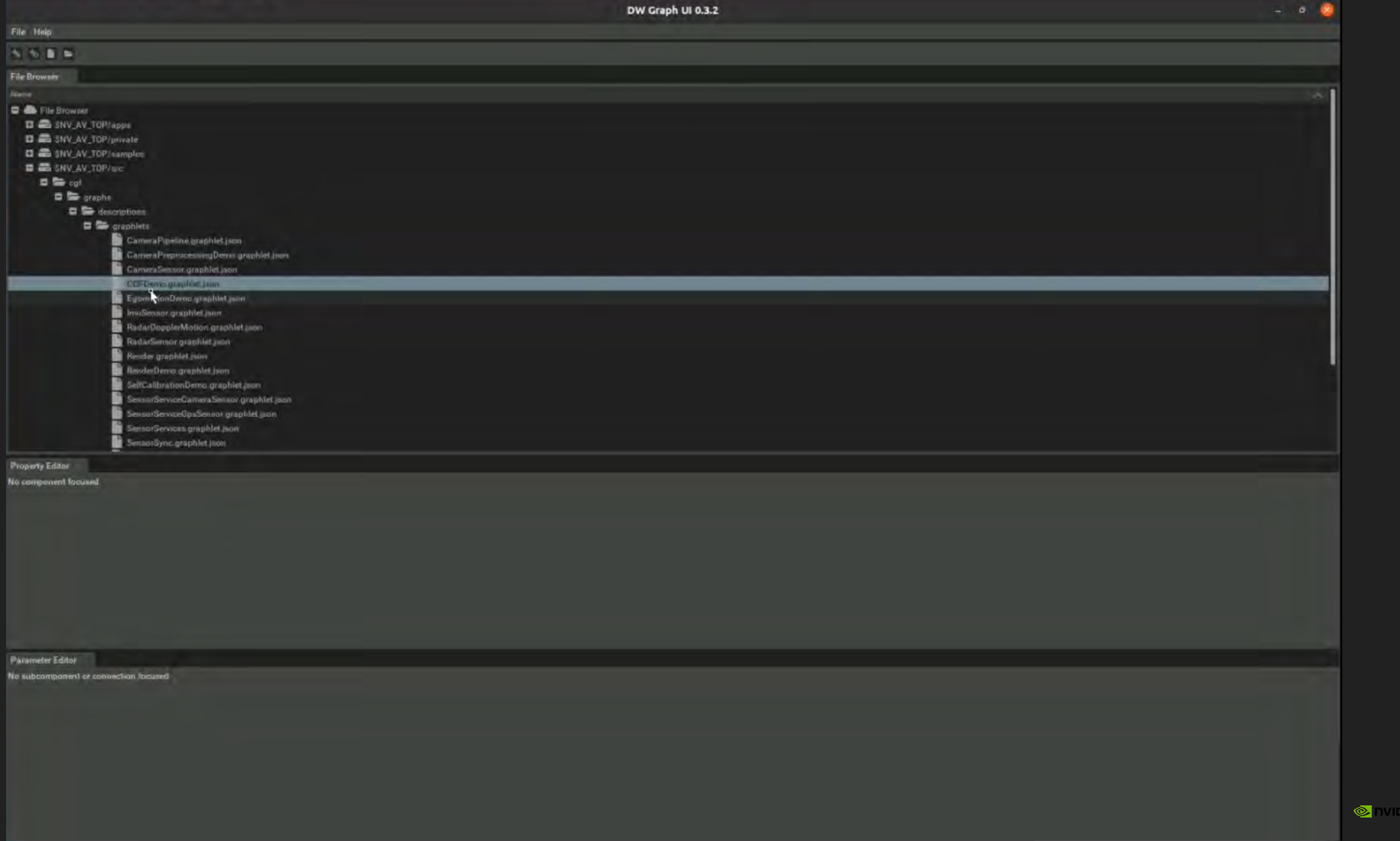
DW Graph UI — Introduction to Nodes and Graphlets



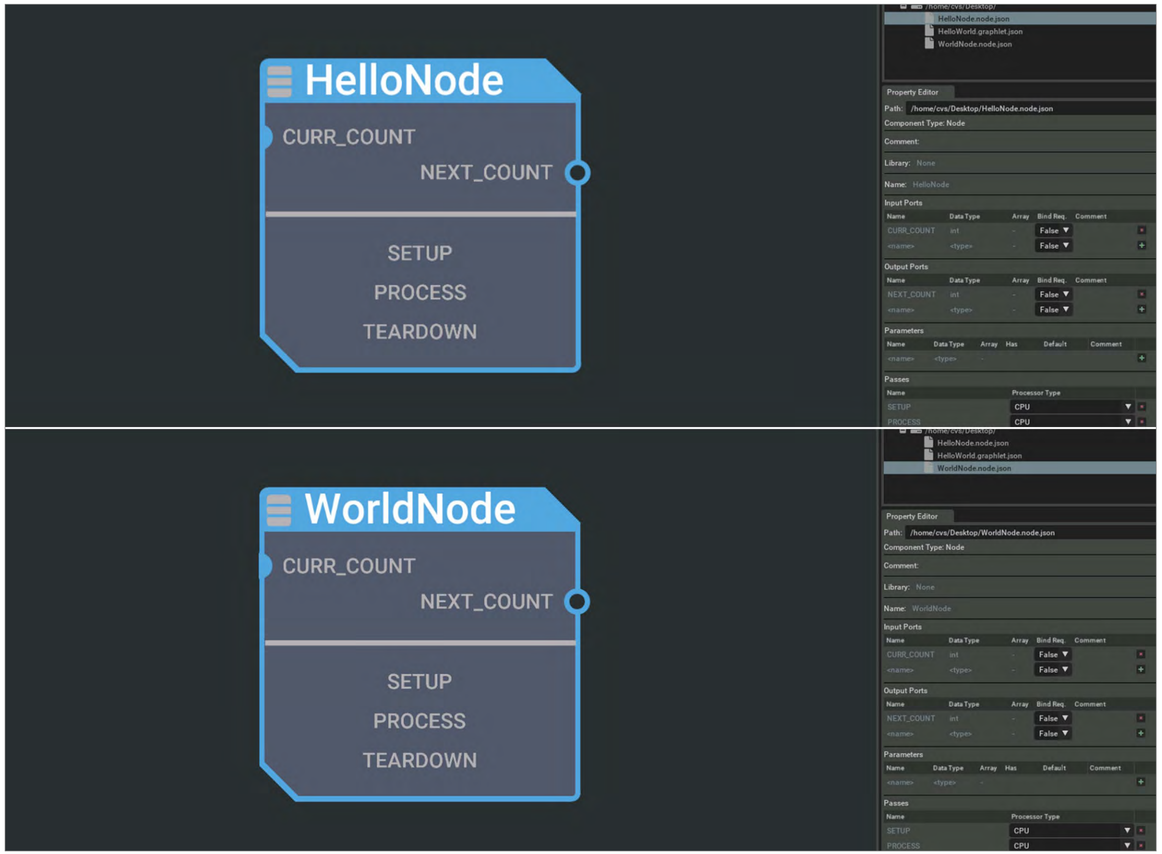
Please Refer:
developer.nvidia.com/docs/drive/drive-os/6.0.6/public/driveworks-nvcgf/cgf_details_node.html

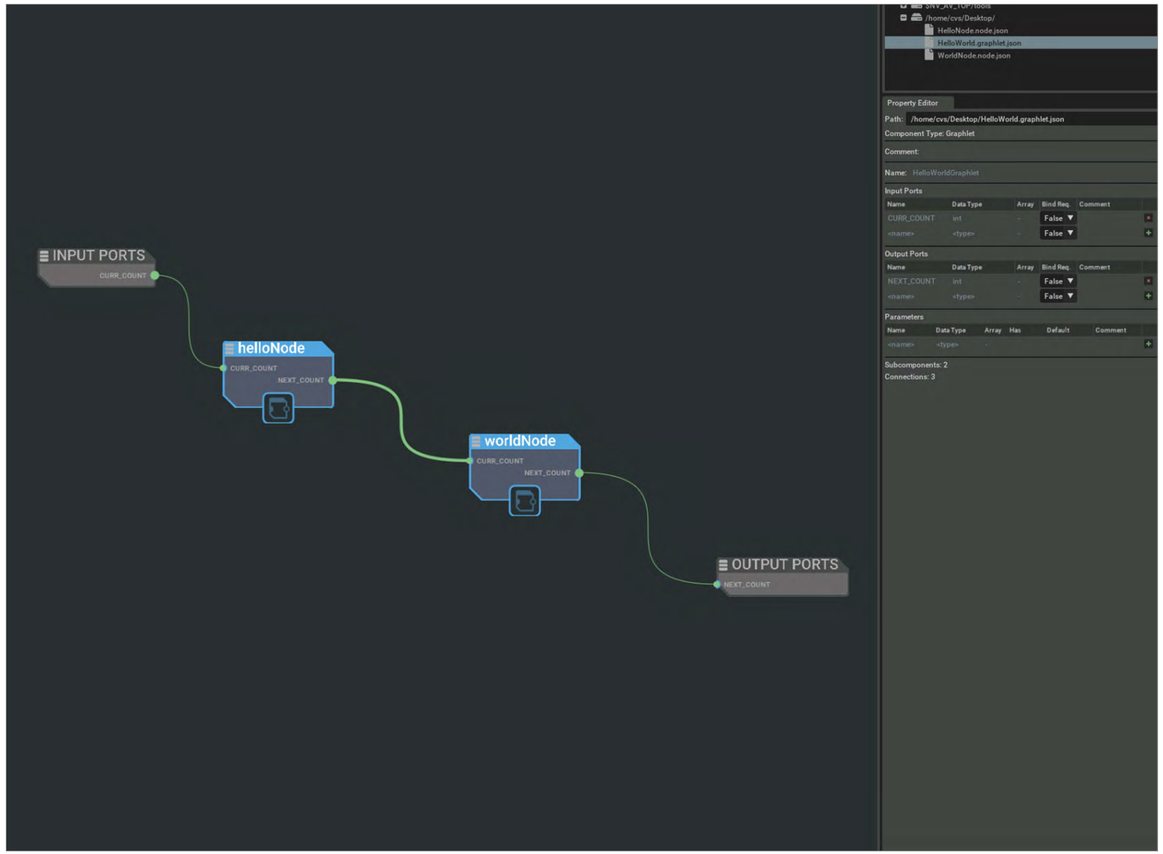


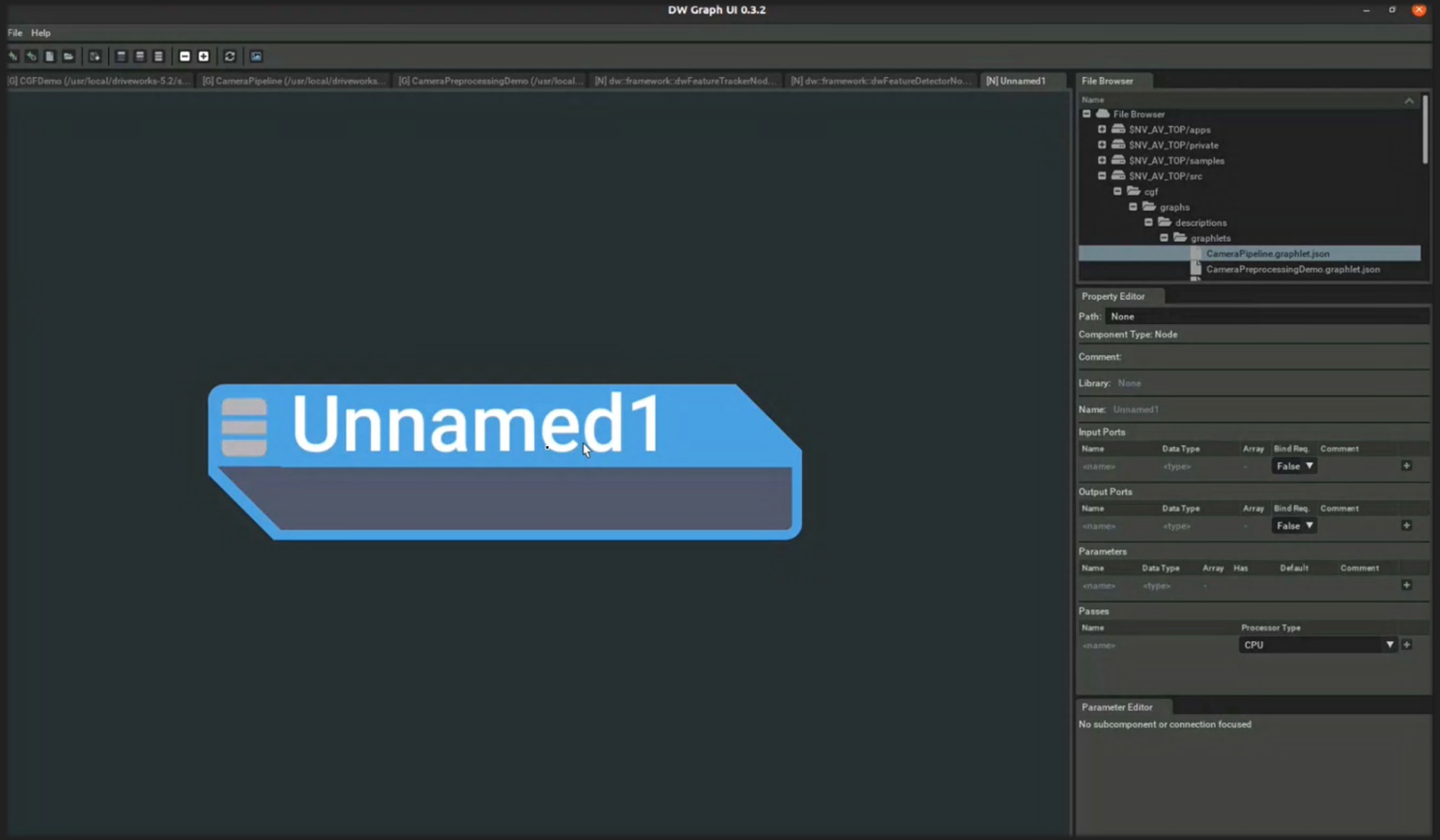
Please Refer:
developer.nvidia.com/docs/drive/drive-os/6.0.6/public/driveworks-nvcgf/cgf_details_graphlet.html


DW Graph UI 0.3.2
File Help
File Browser
Name
File Browser
$NV_AV_TOP/apps
$NV_AV_TOP/private
$NV_AV_TOP/samples
$NV_AV_TOP/src
cgf
graphs
descriptions
graphlets
CameraPipeline.graphlet.json
CameraPreprocessingDemo.graphlet.json
CameraSensor.graphlet.json
CGFDemo.graphlet.json
EgomotionDemo.graphlet.json
ImuSensor.graphlet.json
RadarDopplerMotion.graphlet.json
RadarSensor.graphlet.json
Render.graphlet.json
RenderDemo.graphlet.json
SelfCalibrationDemo.graphlet.json
SensorServiceCameraSensor.graphlet.json
SensorServiceGpsSensor.graphlet.json
SensorServices.graphlet.json
SensorSync.graphlet.json
Property Editor
No component focused
Parameter Editor
No subcomponent or connection focused

DW Graph UI — Introduction to Nodes and Graphlets






DW Graph UI 0.3.2
File Help
[G] CGFDemo (/usr/local/driveworks-5.2/s...
[G] CameraPipeline (/usr/local/driveworks...
[G] CameraPreprocessingDemo (/usr/local...
[N] dw::framework::dwFeatureTrackerNod...
[N] dw::framework::dwFeatureDetectorNo...
[N] Unnamed1
Unnamed1
File Browser
Name
File Browser
$NV_AV_TOP/apps
$NV_AV_TOP/private
$NV_AV_TOP/samples
$NV_AV_TOP/src
cgf
graphs
descriptions
graphlets
CameraPipeline.graphlet.json
CameraPreprocessingDemo.graphlet.json
Property Editor
Path: None
Component Type: Node
Comment:
Library: None
Name: Unnamed1
Input Ports
Name Data Type Array Bind Req. Comment
<name> <type> - False
Output Ports
Name Data Type Array Bind Req. Comment
<name> <type> - False
Parameters
Name Data Type Array Has Default Comment
<name> <type> -
Passes
Name Processor Type
<name> CPU
Parameter Editor
No subcomponent or connection focused

Nodestub — Autogenerating Code for Custom Nodes

```
/usr/local/driveworks/tools/nodestub/nodestub.py \
--output-path ./hello-world-cgf/ ./HelloNode.node.json dw::framework::ExceptionSafeProcessNode
Generating files in: hello-world-cgf
* /usr/local/driveworks/tools/nodestub/Node.thpp -> hello-world-cgf/HelloNode.hpp
Unknown data type 'int', additional #include directives might be needed in the node header
* /usr/local/driveworks/tools/nodestub/Node.tcpp -> hello-world-cgf/HelloNode.cpp
* /usr/local/driveworks/tools/nodestub/NodeImpl.thpp -> hello-world-cgf/HelloNodeImpl.hpp
* /usr/local/driveworks/tools/nodestub/NodeImpl.tcpp -> hello-world-cgf/HelloNodeImpl.cpp
```

```
/usr/local/driveworks/tools/nodestub/nodestub.py \
--output-path ./hello-world-cgf/ ./WorldNode.node.json dw::framework::ExceptionSafeProcessNode
Generating files in: hello-world-cgf
* /usr/local/driveworks/tools/nodestub/Node.thpp -> hello-world-cgf/WorldNode.hpp
Unknown data type 'int', additional #include directives might be needed in the node header
* /usr/local/driveworks/tools/nodestub/Node.tcpp -> hello-world-cgf/WorldNode.cpp
* /usr/local/driveworks/tools/nodestub/NodeImpl.thpp -> hello-world-cgf/WorldNodeImpl.hpp
* /usr/local/driveworks/tools/nodestub/NodeImpl.tcpp -> hello-world-cgf/WorldNodeImpl.cpp
```

```
cvs@cvs-dt:~/Desktop$ /usr/local/driveworks/tools/nodestub/nodestub.py --output-path ./hello-world-cgf/ ./WorldNode.node.json dw::framework::ExceptionSafeProcessNode
```

HelloNode.node.json | WorldNode.node.json | HelloWorld.graphlet.json | **HelloNodeImpl.cpp** | WorldNodeImpl.cpp

```
1  ////////////////////////////////////////////////////////////////////////////////////////
2  //
3  // Notice
4  // ALL NVIDIA DESIGN SPECIFICATIONS AND CODE ("MATERIALS") ARE PROVIDED "AS IS" NVIDIA MAKES
5  // NO REPRESENTATIONS, WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO
6  // THE MATERIALS, AND EXPRESSLY DISCLAIMS ANY IMPLIED WARRANTIES OF NONINFRINGEMENT,
7  // MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8  //
9  // NVIDIA CORPORATION & AFFILIATES assumes no responsibility for the consequences of use of such
10 // information or for any infringement of patents or other rights of third parties that may
11 // result from its use. No license is granted by implication or otherwise under any patent
12 // or patent rights of NVIDIA CORPORATION & AFFILIATES. No third party distribution is allowed unless
13 // expressly authorized by NVIDIA. Details are subject to change without notice.
14 // This code supersedes and replaces all information previously supplied.
15 // NVIDIA CORPORATION & AFFILIATES products are not authorized for use as critical
16 // components in life support devices or systems without express written approval of
17 // NVIDIA CORPORATION & AFFILIATES.
18 //
19 // SPDX-FileCopyrightText: Copyright (c) 2022 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
20 // SPDX-License-Identifier: LicenseRef-NvidiaProprietary
21 //
22 // NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
23 // property and proprietary rights in and to this material, related
24 // documentation and any modifications thereto. Any use, reproduction,
25 // disclosure or distribution of this material and related documentation
26 // without an express license agreement from NVIDIA CORPORATION or
27 // its affiliates is strictly prohibited.
28 //
29 ////////////////////////////////////////////////////////////////////////////////////////
30
31 #include " /HelloNodeImpl.hpp"
32
33
34 constexpr char HelloNodeImpl::LOG_TAG[];
35
36 HelloNodeImpl::HelloNodeImpl(const dwContextHandle_t ctx)
37 {
38     initInputPorts();
39     initOutputPorts();
40     registerPasses();
41 }
42
43 HelloNodeImpl::~HelloNodeImpl()
44 {
45 }
46
47 void HelloNodeImpl::initInputPorts()
48 {
49     using namespace dw::core;  // for operator"" _sv
50     NODE_INIT_INPUT_PORT("CURR_COUNT"_sv);
51 }
52
53 void HelloNodeImpl::initOutputPorts()
54 {
55     using namespace dw::core;  // for operator"" _sv
56     {
57         dw::framework::parameter_traits<int>::SpecimenT ref{};
58         NODE_INIT_OUTPUT_PORT("NEXT_COUNT"_sv, ref);
59     }
```

C++ ▾ | Tab Width: 8 ▾ | Ln 1, Col 1 ▾ | INS

NVIDIA

Adding Custom Logic Within Autogenerated Code

```
70
71 dwStatus WorldNodeImpl::processPass()
72 {
73
74     return DW_SUCCESS;
75 }
76
```

Add logic in the auto-generated stub code

- `HelloNodeImpl.cpp`
- `WorldNodeImpl.cpp`

Please Refer:
developer.nvidia.com/docs/drive/drive-os/6.0.6/public/driveworks-nvcgf/cgf_tutorials_node.html

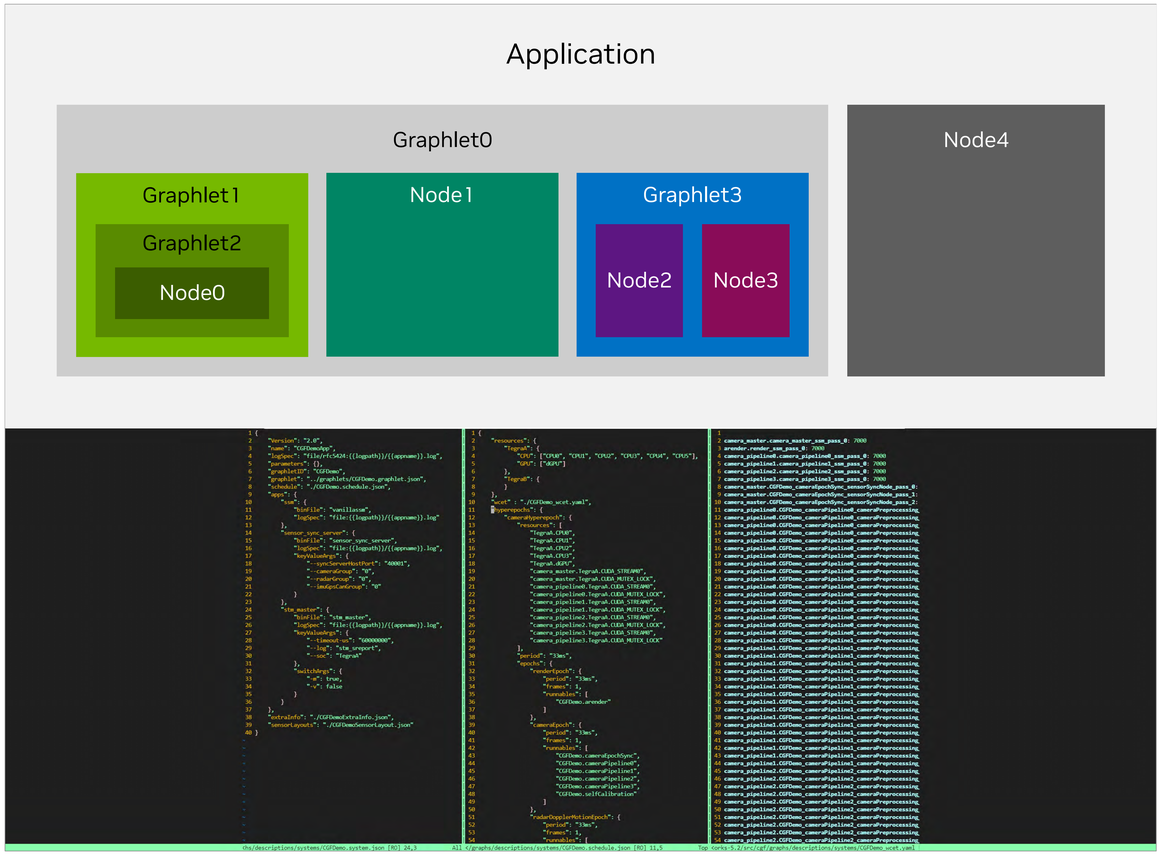
Especially section “Impl Source”

- `NODE_GET_INPUT_PORT()`
- `NODE_GET_OUTPUT_PORT()`

Compiling Custom Logic Within Custom Nodes

```
cvs@cvs-dt:~/Downloads/demo-6.0.2.0-linux/driveworks_workspace/build-linux-aarch64$ make clean
cvs@cvs-dt:~/Downloads/demo-6.0.2.0-linux/driveworks_workspace/build-linux-aarch64$ make cgf_custom_nodes -j8
[  3%] Generating '_data' symbolic link
[  7%] Building CXX object 3rdparty/src/lodepng/CMakeFiles/lodepng-src.dir/src/lodepng.cpp.o
[ 11%] Building CXX object src/framework/CMakeFiles/samples_allocator.dir/Allocator.cpp.o
[ 14%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/monitor.c.o
[ 14%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/init.c.o
[ 18%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/vulkan.c.o
[ 22%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/context.c.o
[ 22%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/input.c.o
[ 22%] Built target create-data-symlink
[ 22%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/window.c.o
[ 25%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/x11_init.c.o
[ 25%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/x11_monitor.c.o
[ 29%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/x11_window.c.o
[ 33%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/xkb_unicode.c.o
[ 33%] Linking CXX static library libsamples_allocator.a
[ 33%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/linux_joystick.c.o
[ 37%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/posix_time.c.o
[ 37%] Built target samples_allocator
[ 37%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/posix_tls.c.o
[ 40%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/glx_context.c.o
[ 40%] Building C object 3rdparty/src/glfw/CMakeFiles/glfw-x11.dir/3.2-screen/src/egl_context.c.o
                                    • • •
[ 81%] Building CXX object src/framework/CMakeFiles/samples_framework.dir/ScreenshotHelper.cpp.o
[ 85%] Building CXX object src/framework/CMakeFiles/samples_framework.dir/RenderUtils.cpp.o
[ 88%] Building CXX object src/framework/CMakeFiles/samples_framework.dir/SamplesDataPath.cpp.o
[ 88%] Building CXX object src/framework/CMakeFiles/samples_framework.dir/WindowEGL.cpp.o
[ 92%] Building CXX object src/framework/CMakeFiles/samples_framework.dir/WindowLinuxEGL.cpp.o
[ 92%] Linking CXX static library libsamples_framework.a
[ 92%] Built target samples_framework
[ 96%] Building CXX object src/cgf_nodes/CMakeFiles/cgf_custom_nodes.dir/HelloWorldNodeImpl.cpp.o
[ 96%] Building CXX object src/cgf_nodes/CMakeFiles/cgf_custom_nodes.dir/HelloWorldNode.cpp.o
[ 96%] Building CXX object src/cgf_nodes/CMakeFiles/cgf_custom_nodes.dir/SumNodeImpl.cpp.o
[100%] Building CXX object src/cgf_nodes/CMakeFiles/cgf_custom_nodes.dir/SumNode.cpp.o
[100%] Linking CXX shared library libcgf_custom_nodes.so
[100%] Built target cgf_custom_nodes
cvs@cvs-dt:~/Downloads/demo-6.0.2.0-linux/driveworks_workspace/build-linux-aarch64$ ll ./src/cgf_nodes/libcgf_custom_nodes.so
-rwxrwxr-x 1 cvs cvs 412400 Jul 28 12:19 ./src/cgf_nodes/libcgf_custom_nodes.so*
```

Introduction to an Application

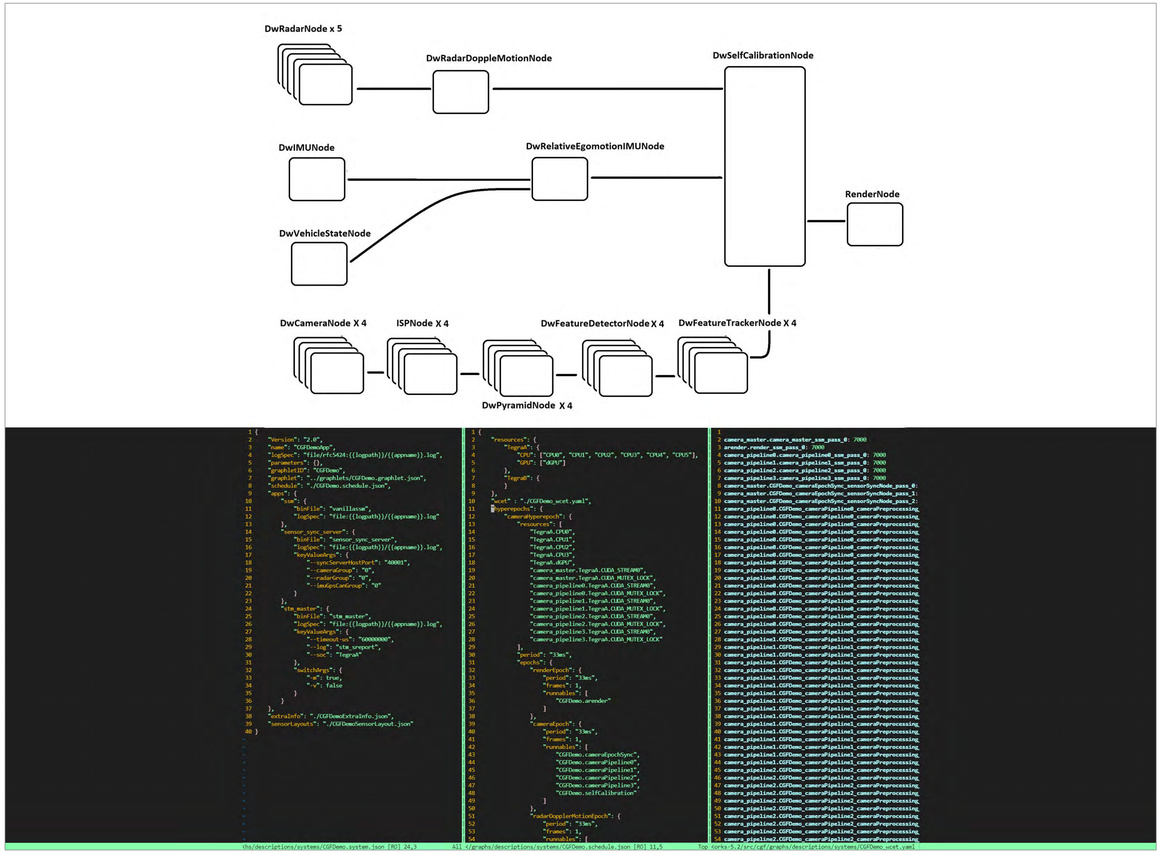


Graphlet Hierarchy

An application is composed of...

- a **data pipeline**
  described by one or multiple graphlets defining a DAG
- an **execution model**
  described by the scheduling and the process layout
- various **configuration parameters**
  eg. configuring logging

CGFDemo Application



Please Refer:

developer.nvidia.com/docs/drive/drive-os/6.0.6/public/driveworks-nvcgf/cgf_samples_demo.html

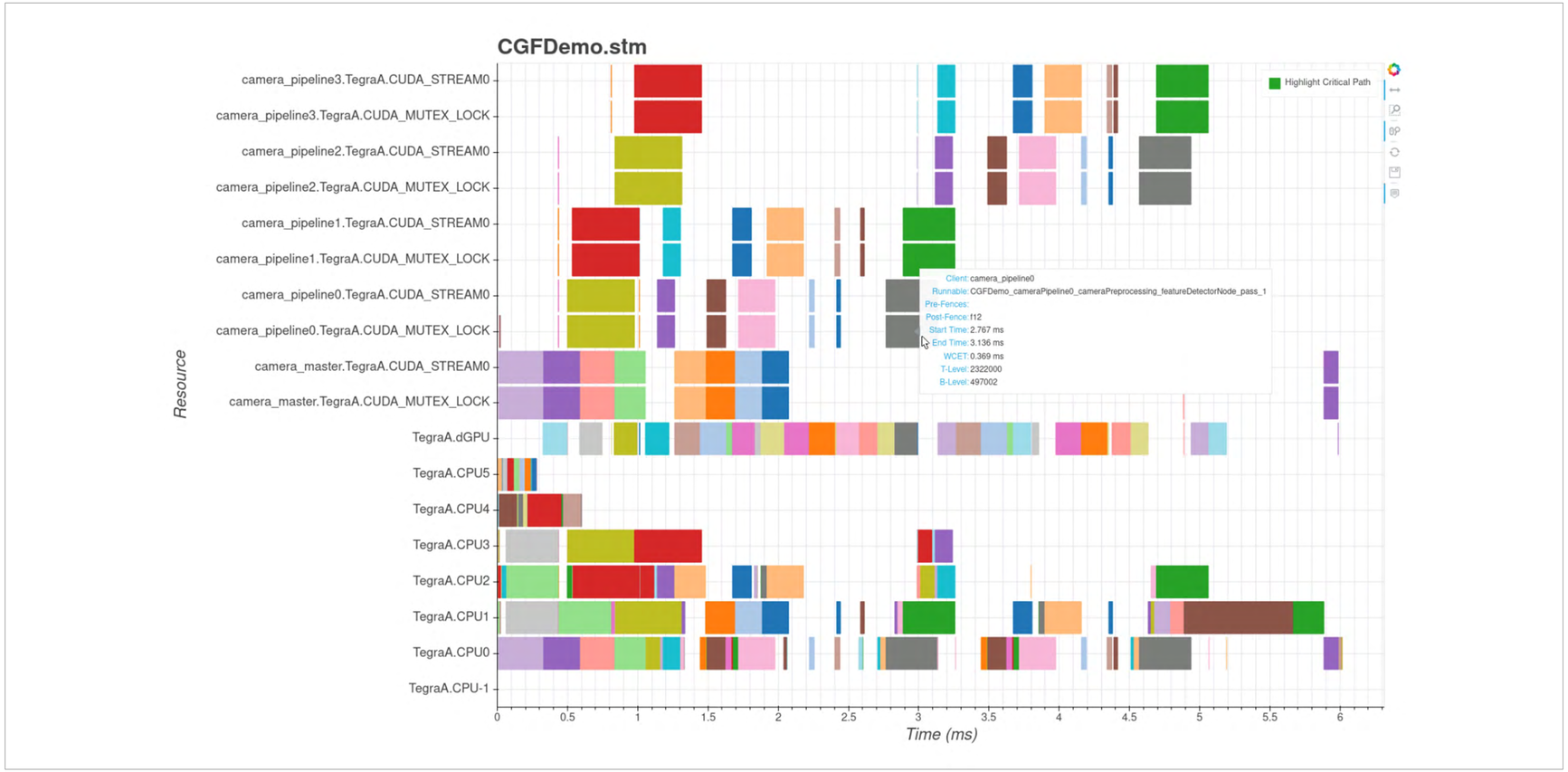
Nodes in CGFDemo pipeline

- **RenderingCGFDemoNode**: Rendering node for the demo pipeline. The node is described by the RenderingCGFDemoNode.node.json file
- **dwRadarNode**: 5 instantiations of this node to create 5 radar inputs. Each node is described by the dwRadarNode.node.json file
- **dwIMUNode**: 1 instantiation of this node to create 1 IMU input. This node is described by the dwIMUNode.node.json file
- **dwVehicleStateNode**: 1 instantiation of this node, described by the dwVehicleStateNode.node.json file
- **dwCameraNode**: 4 instantiations of this node to create four camera inputs. The node is described by the dwCameraNode.node.json file
- **SensorSyncNode**: this node is responsible for sensor inputs synchronization. This node is described by the SensorSyncNode.node.json file
- **dwRelativeEgomotionIMUNode**: this node is responsible to generate egomotion, described by the dwRelativeEgomotionIMUNode.node.json file
- **dwRadarDopplerMotionNode**: this node post processes radar sensor outputs for self-calibration, described by dwRadarDopplerMotionNode.node.json
- **dwPyramidNode**: this node prepares the images in pyramid representation for feature detection purposes, described in dwPyramidNode.node.json file
- **dwFeatureDetectionNode**: this node is responsible for feature detection, described by the dwFeatureDetectorNode.node.json file
- **dwFeatureTrackerNode**: this node is responsible for feature tracking, described by the dwFeatureTrackerNode.node.json file
- **dwSelfCalibrationNode**: this node is responsible to generate self-calibration results, described by the dwSelfCalibrationNode.node.json

Graphlets in CGFDemo pipeline

- **RenderDemo graphlet**: this graphlet contains RenderingCGFDemoNode node
- **RadarSensor graphlet**: this graphlet contains dwRadarnode node
- **ImuSensor graphlet**: this graphlet contains dwIMUnode node
- **VehicleStateConsumer graphlet**: this graphlet contains dwVehicleStatenode node
- **CameraSensor graphlet**: this graphlet contains dwCameraNode and ISPNode nodes. Camera frames are fed through dwCameraNode and ISPNode to provide demosaiced image outputs to post processing in later pipeline
- **SensorSync graphlet**: this graphlet contains SensorSyncNode node
- **EgomotionDemo graphlet**: this graphlet contains two dwRelativeEgomotionIMUNode nodes. One of the egomotion is responsible for outputting egomotion state as odometry whereas the other egomotion node generates egomotion state that later feeds into self calibration graphlet
- **RadarDopplerMotion graphlet**: this graphlet contains eight instantiation of dwRadarDopplerMotionNodes
- **CameraPreprocessingDemo graphlet**: this graphlet contains dwPyramidNode, dwFeatureDetectionNode, and dwFeatureTrackerNode. The data feeds through these three nodes to produce feature tracking that are fed into self calibration and rendering nodes
- **CameraPipeline graphlet**: this graphlet contains three sub-graphlets, CameraSensor, CameraPreprocessingDemo, and CameraObjectDetectorDemo graphlets
- **SelfCalibrationDemo graphlet**: this graphlet contains dwSelfCalibrationNode node
- **CGFDemo graphlet**: this graphlet is the main graphlet that connects all graphlets in this demo

Static Visualization — STMVIZSCHEDULE



CGFDemo Application

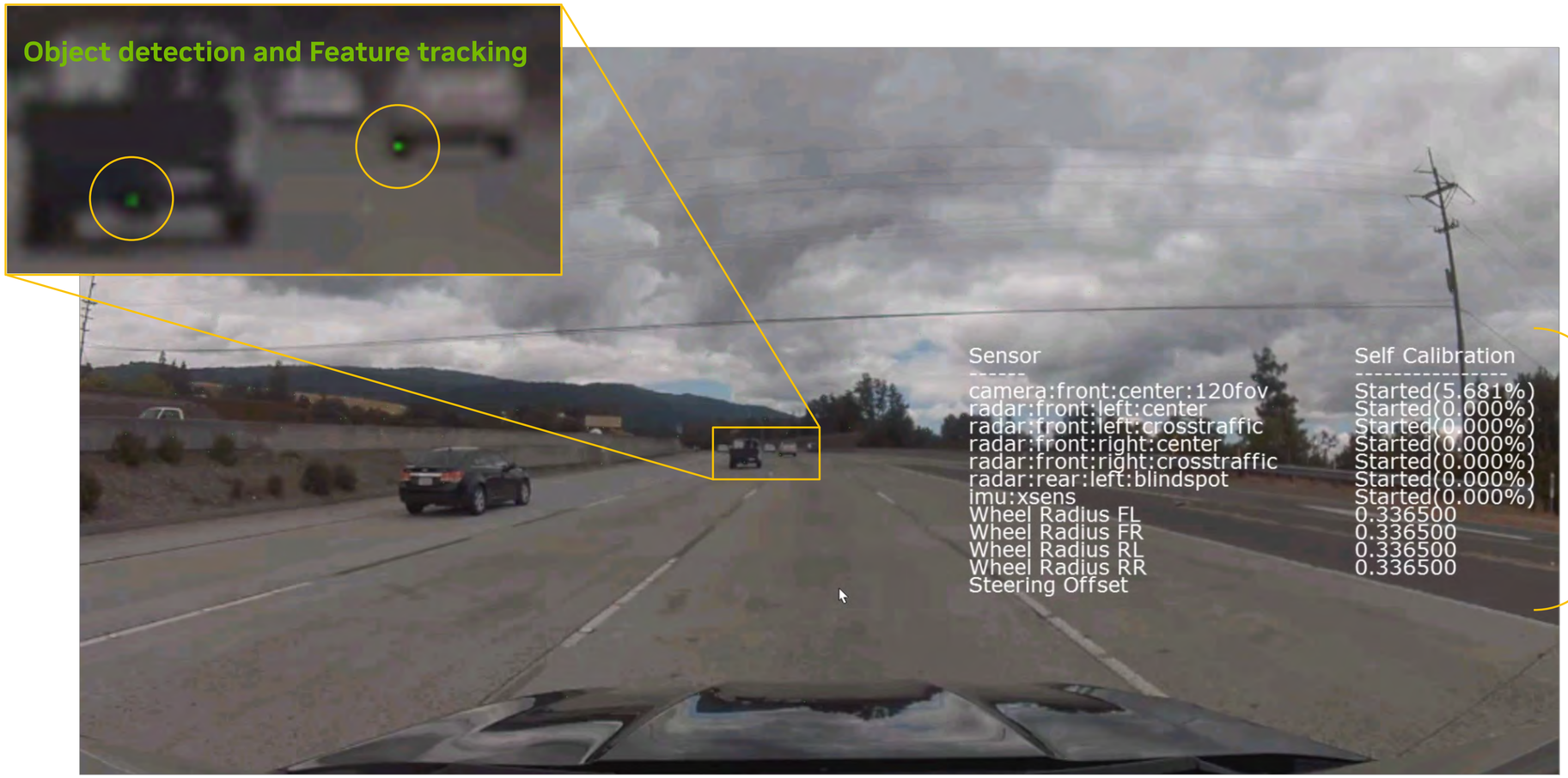
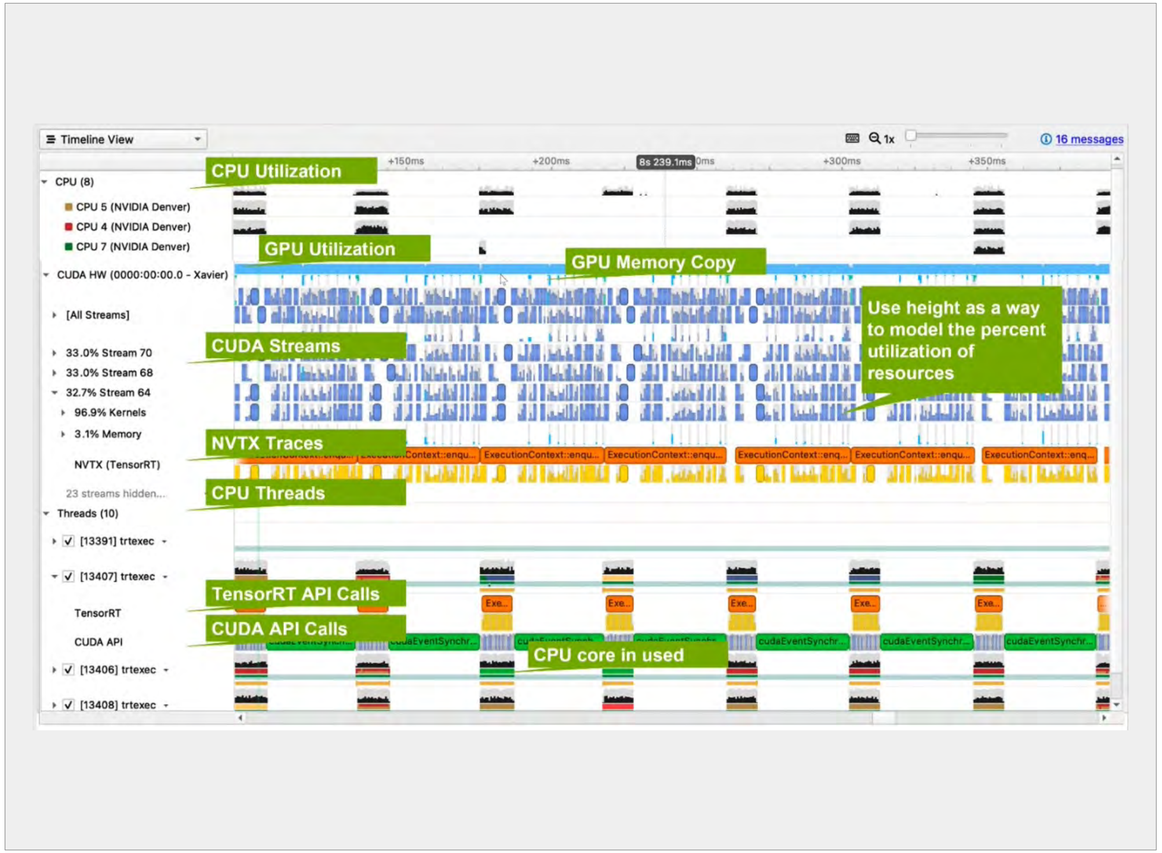


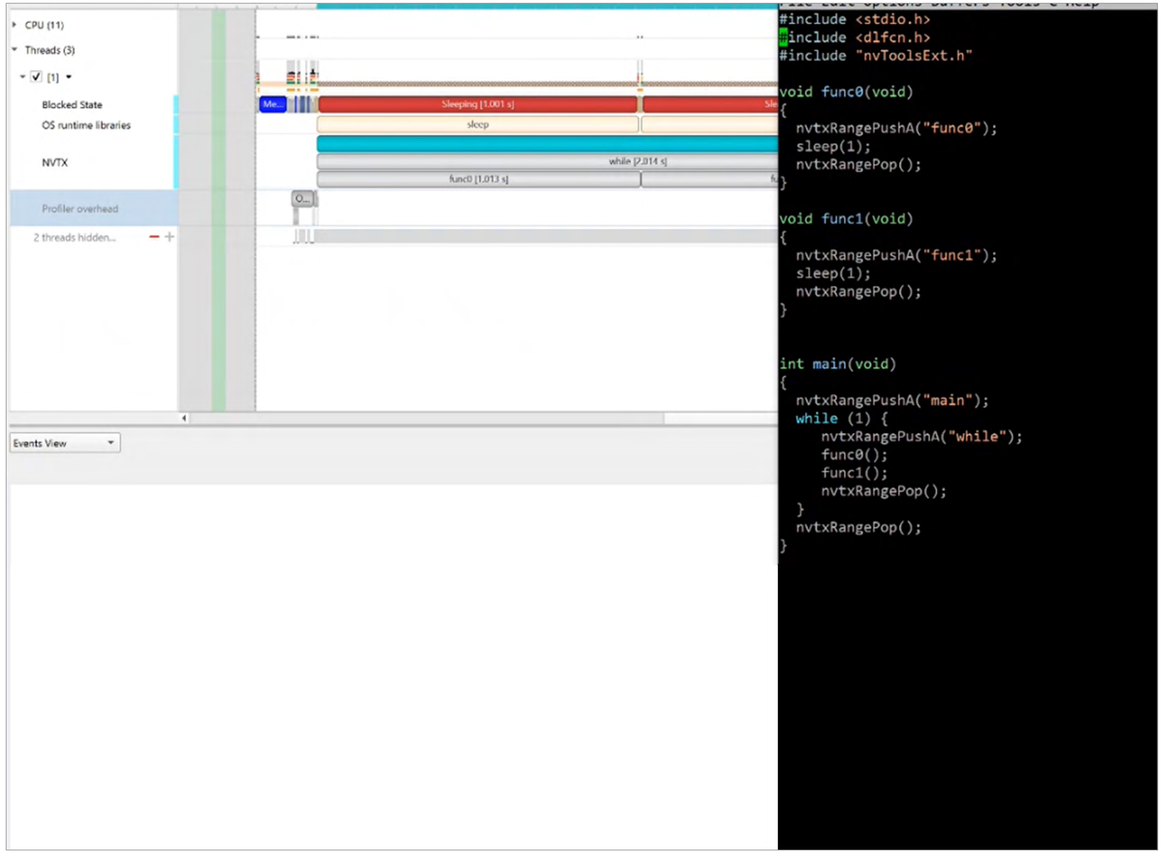
Table of Self-Calibration Status

Please Refer:

developer.nvidia.com/docs/drive/drive-os/6.0.6/public/driveworks-nvcgf/cgf_samples_demo.html

Profiling — Nsight Systems + NVTX

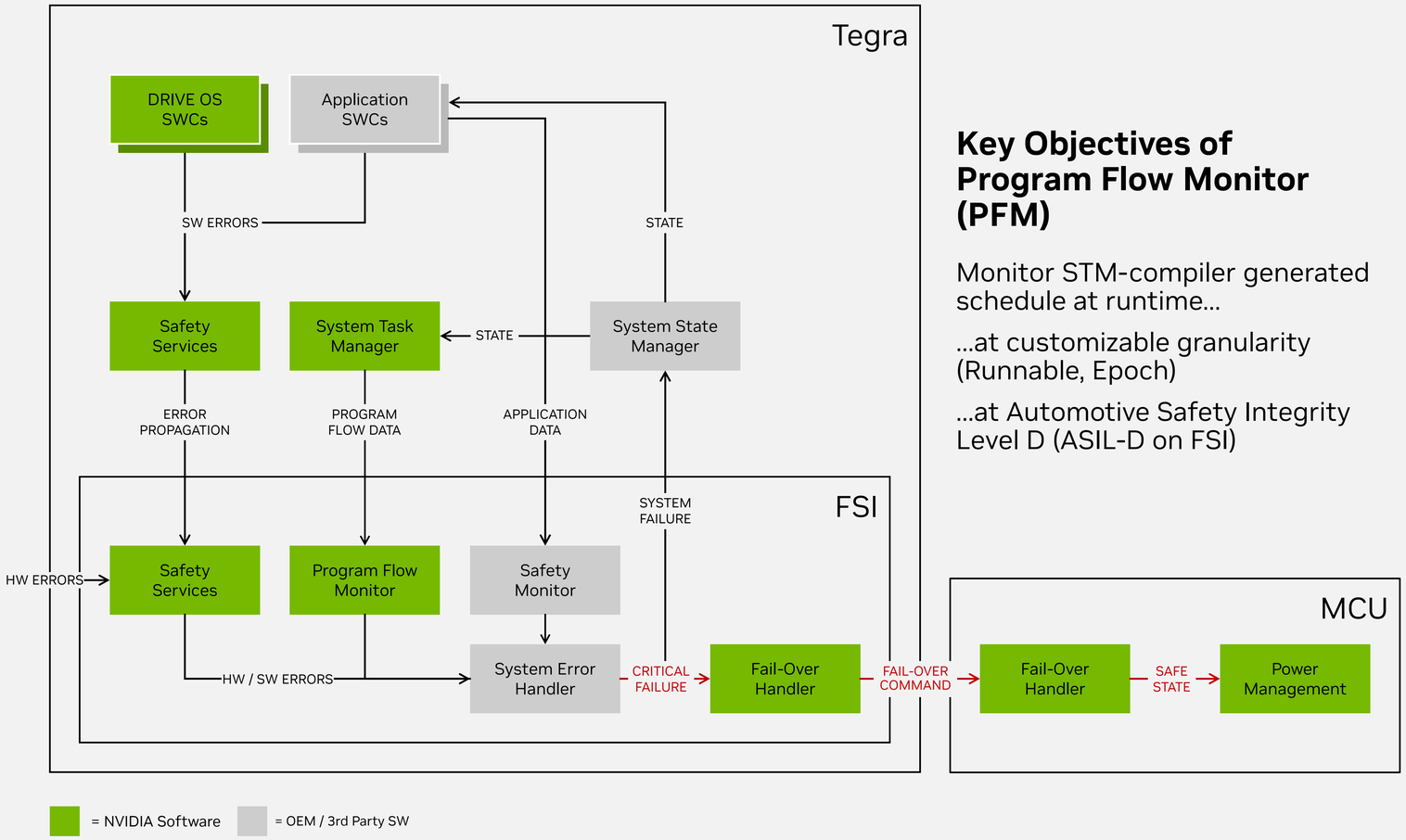




Please Refer : developer.nvidia.com/nsight-systems

docs.nvidia.com/gameworks/content/gameworkslibrary/nvtx/nsight-systems-nvtx-trace.htm

PFM in the Overall Safety Concept



Key Objectives of Program Flow Monitor (PFM)

Monitor STM-compiler generated schedule at runtime...

...at customizable granularity (Runnable, Epoch)

...at Automotive Safety Integrity Level D (ASIL-D on FSI)

Get Started with DRIVE SDK

Extensive documentation and training material available on NVIDIA Developer

Learn More

- Visit the DRIVE training page for webinars and other resources
- Check out information related to DRIVE Hyperion, DRIVE AGX Orin and DRIVE SDK

Get Access

- Join the DRIVE AGX SDK Program on NVIDIA Developer
- Read the docs for DRIVE OS and DriveWorks documentation
- Download DRIVE OS which includes DriveWorks, NvMedia, CUDA, cuDNN and TensorRT

Contact Us

- Contact your distributor or NVIDIA's Automotive Team


NVIDIA