Frame Debugger

Overview

Frame Debugger

The CGI Studio Analyzer contains a Frame Debugger. This feature can be used to analyze the rendering process of Candera on a target device. It provides an overview about the composition of one frame (rendered nodes), shows detailed OpenGL information and adds the ability to capture the color, depth and stencil buffers.

Requirements

- Windows Host running CGI Analyzer
- Application
  - Monitor and Frame Debugger enabled (CMake)
  - TCP/IP connection
  - Enough free RAM during execution on the target device to copy framebuffer contents.

Supported Platforms

- Host Simulation
- iMX6 Integrity
- Experimental support for other OpenGL ES 2.0 and OpenGL ES 3.0 based devices

Features

- Capture frame timeline (rendered cameras, nodes, OpenGL calls, ...)
- Capture while application is running or in paused mode
- Single step to next frame
- Capture single node/single camera
  - View color, depth and stencil buffer before & after a node was rendered
  - Highlight pixels changed by the selected node
  - Save captured images as PNG
- Visualize intermediate results of rendered nodes.

Setup

Build

To use the Frame Debugger, build the application with these CMake flags enabled:

FEATSTD_MONITOR_ENABLED	Enables basic Monitor features.
MONITOR_FRAME_DEBUGGER_ENABLED	Enables basic frame debugger (frame timeline capture, single step mode, no OpenGL specific features).
MONITOR_FRAME_DEBUGGER_OPENGL_CAPTURE_ENABLED	Enables OpenGL specific Frame Debugger features.
COURIER_RENDERING_MONITOR_ENABLED	Additional flag required for Courier apps to enable monitor.

Also make sure **MONITOR_COM_TYPE** is set to **FEATSTD_COM_TYPE_TCPIP** and **MONITOR_TCPIP_ADDRESS** and **MONITOR_TCPIP_PORT** are set to appropriate values.

Additional CMake flags: The flag **MONITOR_SINGLE_STEP_INTERVAL** can be used to set the time in milliseconds that is used when incrementing the animation time in single-step mode.

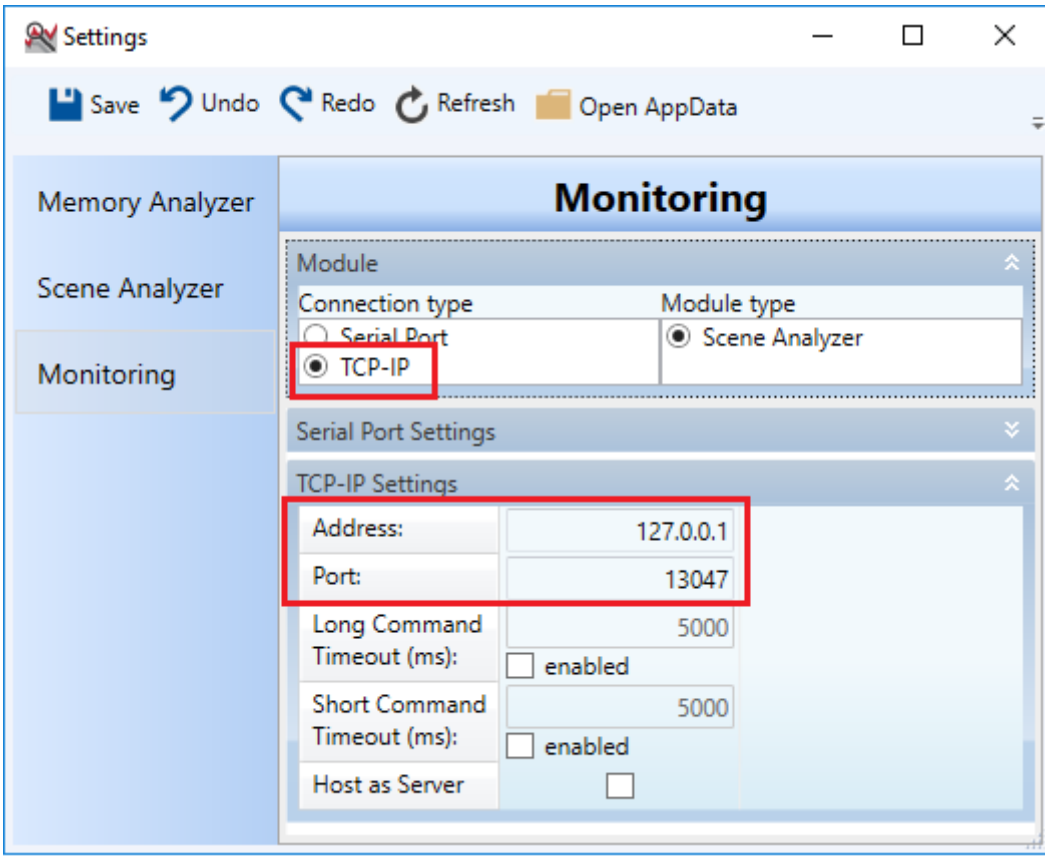
Render loop

When using an application with a custom render loop (anything else than Player) add these macros to the beginning and end of the loop (defined in Candera/System/Monitor/MonitorPublicIF.h):

- MONITOR_MARKER_BEGIN_FRAME();
- MONITOR_MARKER_END_FRAME();

CGI Analyzer

To start the Frame Debugger, configure the connection settings in CGI Analyzer and then connect to the running application:

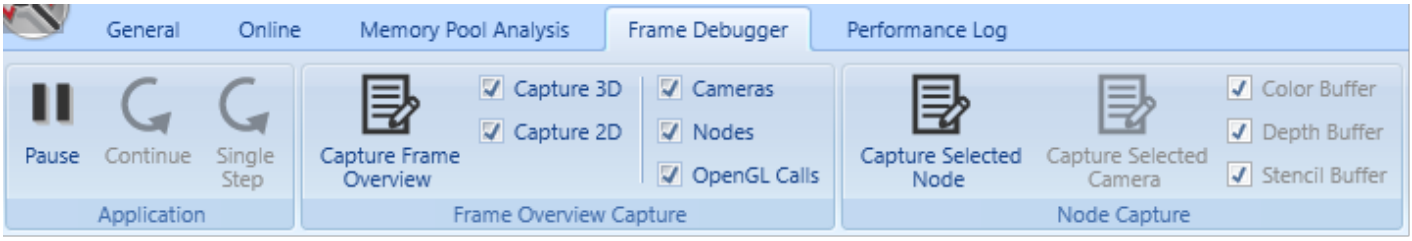

Settings
Save
Undo
Redo
Refresh
Open AppData
Memory Analyzer
Scene Analyzer
Monitoring
Monitoring
Module
Connection type
Serial Port
TCP-IP
Module type
Scene Analyzer
Serial Port Settings
TCP-IP Settings
Address:
127.0.0.1
Port:
13047
Long Command Timeout (ms):
5000
enabled
Short Command Timeout (ms):
5000
enabled
Host as Server

Usage

Main Toolbar

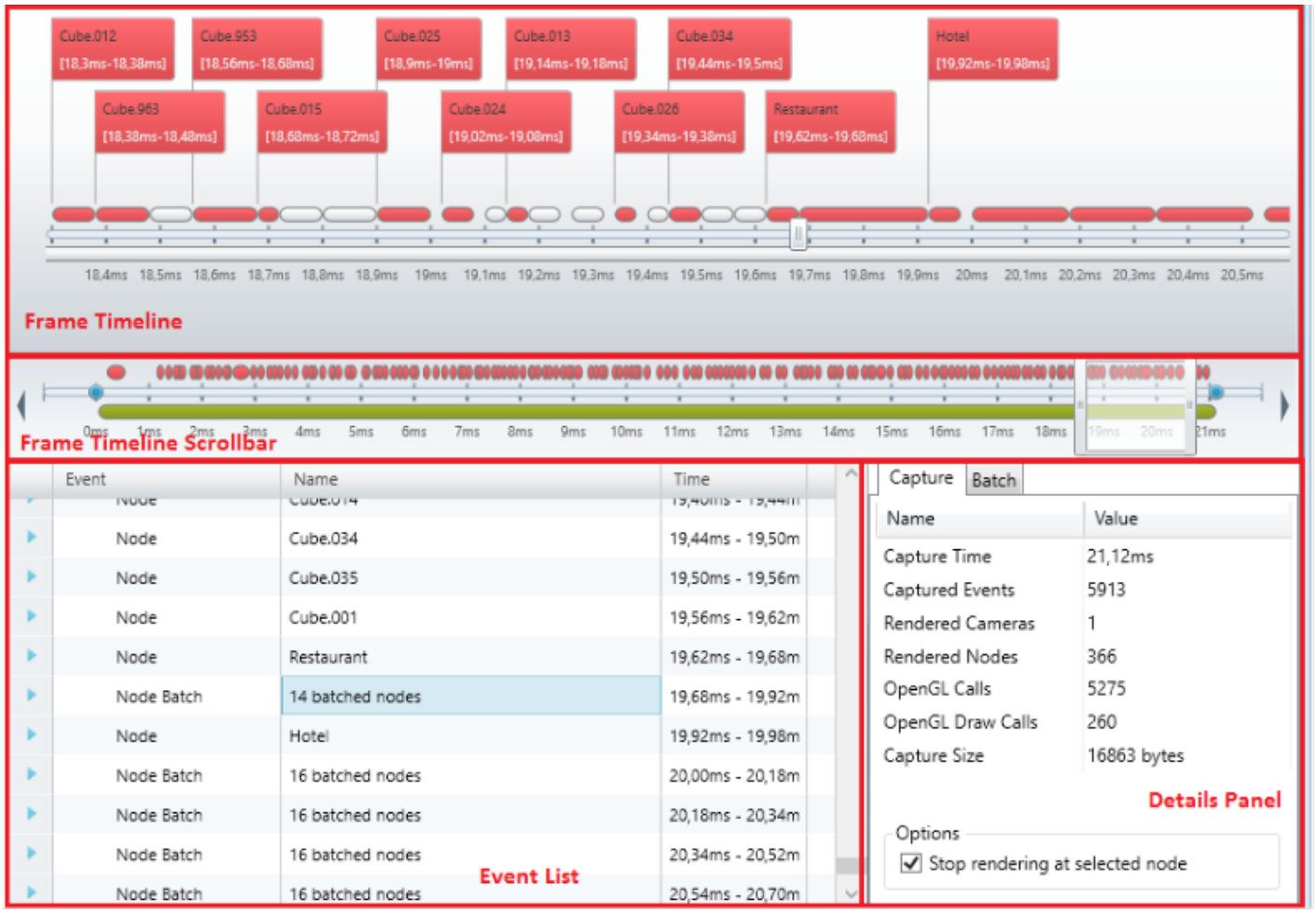
After the connection to the application has been established, the main Frame Debugger toolbar will become active.

> The available features here depend on the enabled features in the application during build time and on the platform on which the application is running.



- **Application**
  - Control application execution.
  - **Pause**: Pause application
  - **Continue**: Continue application execution
  - **Single Step**: Continue application for one frame
- **Frame Overview Capture**
  - Starts recording of all events of the next rendered frame in the application. The checkboxes indicate which events will be captured.
  - **Capture 2D/3D**: Indicates whether 2D/3D rendering events will be captured.
  - **Cameras**: Track which cameras are rendered.
  - **Nodes**: Track which nodes are rendered.
  - **OpenGL**: Track all OpenGL calls issued during the frame.
- **Capture Node**
  - This button is enabled after a frame overview capture has been completed and a valid node is selected in the resulting frame timeline. The checkboxes indicate which buffers will be captured.
  - The Node Capture will capture the state of the render target before and after the selected node has been rendered.
- **Capture Camera**
  - This button is enabled after a frame overview capture has been completed and a valid camera is selected in the resulting frame timeline. The checkboxes indicate which buffers will be captured.
  - The Camera Capture feature will capture the state of the render target before and after the selected camera has been rendered.

Frame Overview Capture



The frame timeline displays the cameras and nodes rendered during one frame in chronological order. Timestamps are measured relative to the beginning of the frame. The event list displays all recorded events as hierarchical list (Frame -> Camera -> Nodes -> OpenGL calls).

The details panel displays multiple tabs with additional information, depending on the currently selected event in the event list. The “Capture” tab will always be displayed. It contains general information about the frame capture and additional display options. Further tabs like “Batch” are dynamic and depend on the selected node.

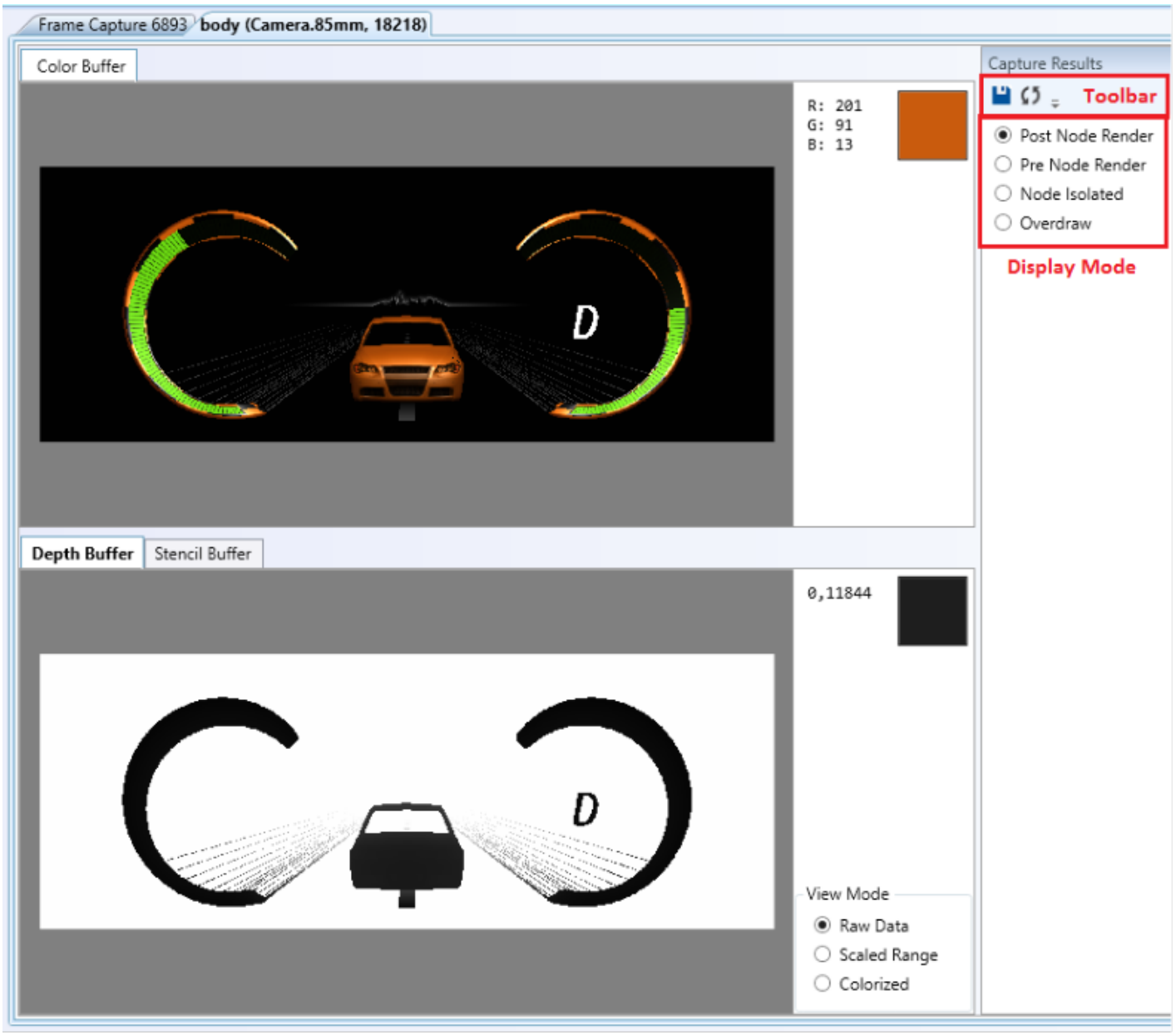
"Stop rendering at selected node": If this is selected and the target device is still connected, rendering on the target device will be stopped at the selected node. All nodes that appear after the selected one in the render order will be skipped.

> If an animation is running on the target device the render order might change dynamically and results will vary.

When a node is selected in the event list, the “Note Capture” feature will be available in the toolbar. When a camera is selected in the event list, the “Camera Capture” feature will be available in the toolbar.

> Due to the overhead associated with event recording the displayed timestamps might not accurately represent the render performance and should only be used for relative comparisons. For more accurate render performance measurements use the “Key Performance Indicators” feature of the CGI Analyzer.

Node/Camera Capture



The windows displays the result of a node/camera capture.

- **Color Buffer**:
  Displays the captured color data. To the right of the image the data of the currently hovered pixel is displayed.
- **Depth Buffer/Stencil Buffer**:
  Displays the captured depth/stencil data. The currently hovered pixel is displayed to the right. View Mode can be used to display the data in different ways:
  - **Raw Data**: Show original data
  - **Scaled range**: Scale the values in the image to use the maximum available greyscale range.
  - **Colorized**: Computes false-color data to better visualize different data values.
- **Toolbar**:
  - **Save**: Select a folder to save all captured images as PNG files.
  - **Refresh**: Capture the same node again.
- **Display Mode**:
  Changes the data set that is shown in the “Color Buffer” and “Depth/Stencil Buffer” sections.
  - **Post Node Render**: Framebuffer contents after the node was rendered.
  - **Pre Node Render**: Framebuffer contents before the node was rendered.
  - **Node Isolated**: Displays only pixel that were changed by the current node.
  - **Overdraw**: highlight (in pink) where the current node drew over already existing pixel data.

> This is not a full overdraw visualization. It is computed based on the recorded data and will not produce exact results in all circumstances (e.g. transparent pixel, depth buffer inaccuracies, ...)

Image Controls

The Color/Depth/Stencil buffer view support the following controls:

- **Mouse Scroll**: Zoom In/Out
- **Mouse Click + Drag**: Move view
- **Right Click**: Reset view

Additionally, the tabs “Color Buffer”, “Depth Buffer” and “Stencil Buffer” can be undocked (drag & drop) to move them into separate windows and change their size.