Multi Frequency Rendering

Description

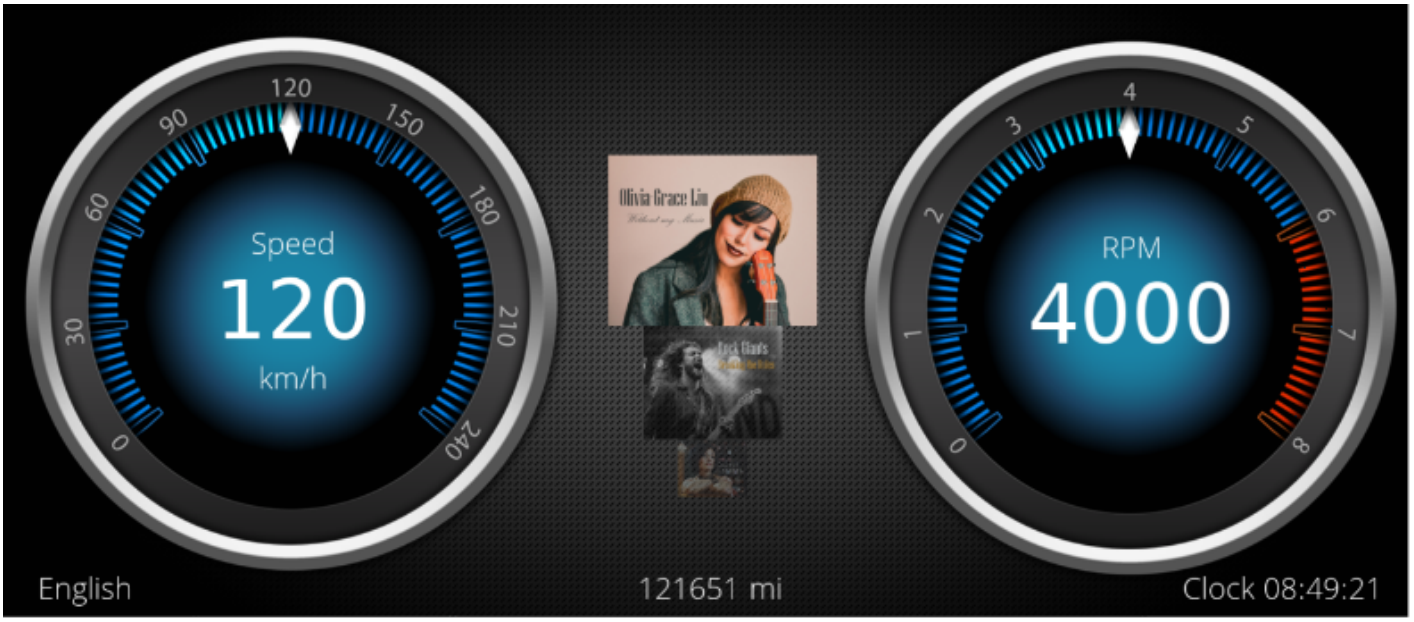
This chapter briefly describes the Multi Frequency Rendering technique supported by Candera.
It allows to split rendering of complex parts to several frames, to achieve an overall higher frame rate.

Introduction

Multi Frequency Rendering Introduction

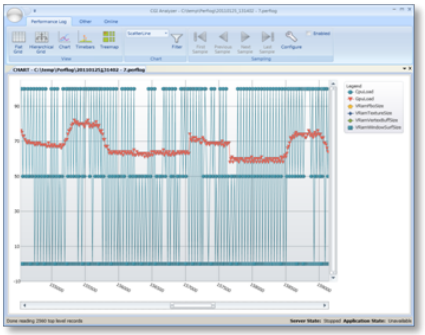
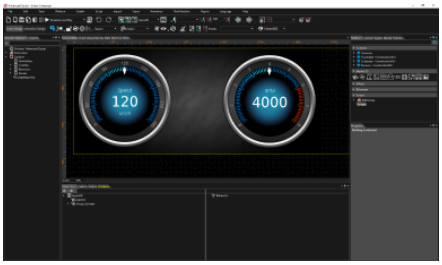
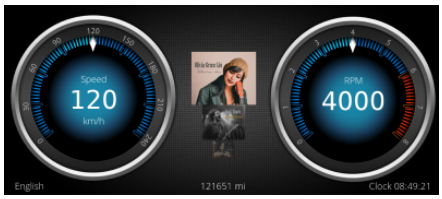
Multi Frequency Rendering allows using different update rates for multiple render targets. For example

- Fix framerate for tubes.
- Adaptive framerate for center use case.
- Details of entire content are preserved.



Workflow

CGI Analyzer	SceneComposer	Application

		
CGI Analyzer determines benchmark values for each node of a scene given. The benchmarks are exported to a *.crsms file.	The benchmarks are assigned to nodes benchmark property. SceneComposer stores nodes incl. benchmark values in Asset File.	The Application renders multiple cameras with different framerates with help of Benchmark-CameraRenderStrategy.

Candera::CameraRenderStrategy

The following sections explain how an application can use the Candera::CameraRenderStrategy to implement multi frequency rendering based on given benchmark values within the scene tree.

References

For details about how to calculate benchmark values and how to assign them to a scene tree, please refer to:

- **CGI Analyzer**: Calculate benchmark values and export to a *.crsms file.
- **SceneComposer Help**: Import benchmark values from a *.crsms file into a given solution.

Candera - Camera Render Strategy

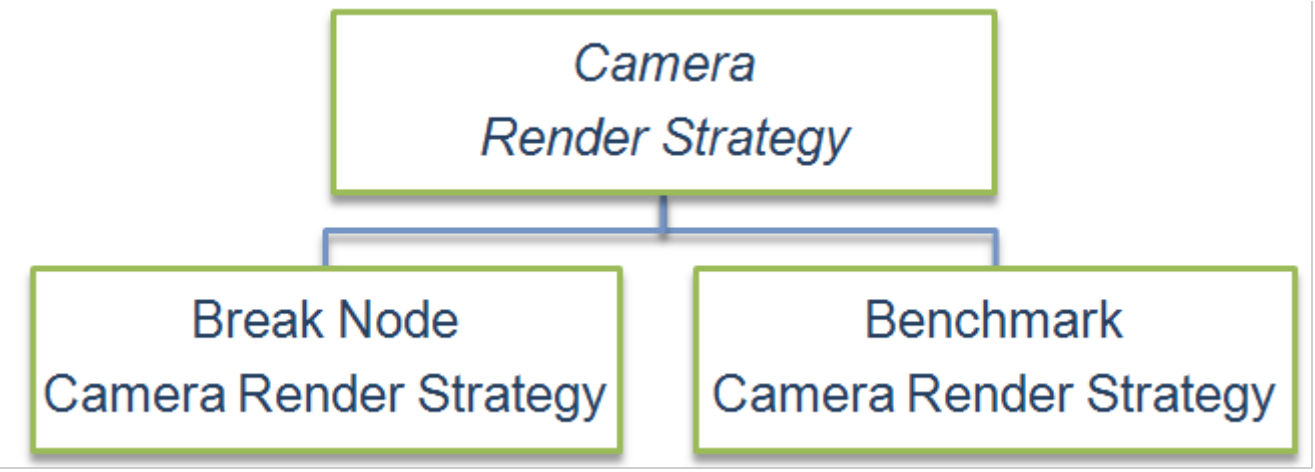
Partitioning Render Passes

A Camera render pass renders all scene nodes in one sweep.

Candera::CameraRenderStrategy enables to partition a Camera render pass, if e.g. complete processing in one frame is not possible.

Examples of Usage

- Omit rendering of nodes with low priority like decorations.
- Multi Frequency Rendering: Suspend and continue Camera render pass next time Camera renders. Unless render pass is complete, render target is not swapped.



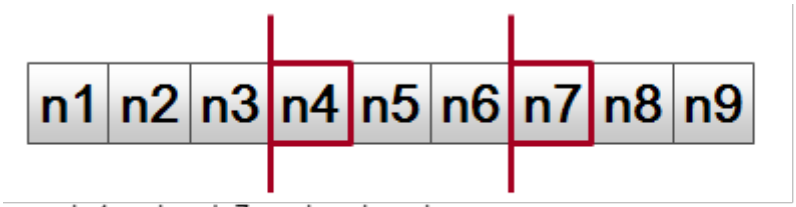
Both, Candera::BreakNodeCameraRenderStrategy and Candera::BenchmarkCameraRenderStrategy are concrete implementations of the interface Candera::CameraRenderStrategy to cover most typical Camera render pass partitioning.

Candera - Break Node Camera Render Strategy

Splitting Render Pass on specific Break Nodes

The Candera::BreakNodeCameraRenderStrategy is populated with a list of arbitrary break nodes. Defined by RenderPassAction a break node can pause or stop a Camera render pass, if encountered.

```
breakNodeStrategy = new BreakNodeCameraRenderStrategy();
breakNodeStrategy->AddBreakNode(node4);
breakNodeStrategy->AddBreakNode(node7);
breakNodeStrategy->SetRenderPassActionOnBreakNode(renderPassAction);
camera->SetCameraRenderStrategy(breakNodeStrategy);
```



node4 and node7 are break nodes
The Renderer behaves like defined by given renderPassAction

Candera - Benchmark Camera Render Strategy

Splitting Render Pass on Benchmark Threshold Values

Each node has a benchmark property that describes a custom reference value for render duration. During Camera render pass benchmarks are accumulated until threshold of

Candera::BenchmarkCameraRenderStrategy is reached. RenderPassAction defines to suspend or stop render pass.

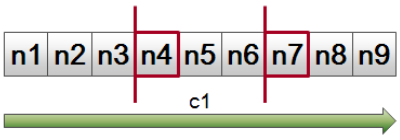
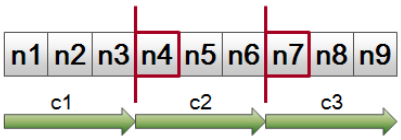
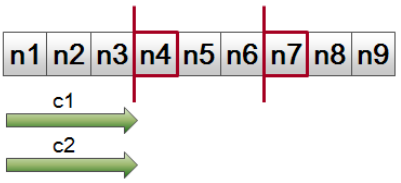
```
benchmarkStrategy = new BenchmarkCameraRenderStrategy();
node1->SetRenderBenchmark(250.0f);
node2->SetRenderBenchmark(330.0f);
node3->SetRenderBenchmark(220.0f);
node4->SetRenderBenchmark(400.0f);
node5->SetRenderBenchmark(390.0f);
benchmarkStrategy->SetThreshold(1000.0f);
benchmarkStrategy->SetRenderPassActionOnThreshold(renderPassAction);
camera->SetCameraRenderStrategy(benchmarkStrategy);
```



Benchmark of Node4 exceeds threshold and triggers RenderPassAction

Candera - Render Pass Actions

Render Pass Actions

		
Proceed Render Pass	**Pause Render Pass**	**Stop Render Pass**
Proceeds Rendering. Rendering is not interrupted. Break nodes are ignored.	Pauses rendering. The next time the Camera is rendered it restarts from the node that has been paused. Swaps render target when Camera completes. Usage: Entire content captured by Camera shall be displayed	Stops rendering. The next time the Camera renders it restarts from the very first node in render order. Swaps render target at each render cycle. Usage: Nodes with lower priority shall be omitted.

Example - Multi Frequency Rendering

Example: Different Frame Rates for several Cameras

- Fix framerate for speedometer.
- Adaptive framerate for car.
- Details of entire content is preserved.

Cycle 1



Camera 1 : Renders the speedometer and swaps render target.
Camera 2 : Renders car until benchmark threshold is exceeded. Render target is not swapped.

Cycle 2



Camera 1 : Renders the speedometer and swaps render target.
Camera 2 : Completes render pass and swaps render target. Thus Camera2 renders with half frames per second (FPS) than Camera1.

Multi Frequency Rendering in 2D

Analogous to Candera::CameraRenderStrategy, there is an abstract class Candera::Camera2DRenderStrategy available.

This class is intended to be derived to tell the Candera::Renderer2D, if rendering shall proceed, pause, or stop for a given node in a camera's render pass.

BreakNodeCamera2DRenderStrategy

In difference to 3D, Candera 2D Engine only provides a Candera::BreakNodeCamera2DRenderStrategy to split a 2D render pass along specific break nodes.

Limitations

MFR animation issue

Using an AnimationReq-Message in combination with MFR results in an wrong output since the animation progresses during each partial drawn frame. One idea is to check if the view uses MFR and if used the view would provide it's own AnimationTime-Dispatcher which only gets updated during the Update call. The animation for this view would then be added to this time dispatcher. This allows to use animations as long as they only affect a single scene.

MFR transition issue

Using a Scene-Transition in combination with MFR results in an wrong output since the transition progresses during each partial drawn frame. Transitions should check the status of the camera and only perform a modification if the Render-Pass is complete. This would allow transitions to be used if only one Scene is active per Render-Target. To allow multiple scenes per Render-Target (more common case) all Cameras in the transition need to be "synced".

Revision #11
Created 2023-02-17 08:08:55 UTC by Tsuyoshi.Kato
Updated 2026-08-17 23:35:08 UTC by Tsuyoshi.Kato