Samples Pack

PIXEL_LIGHTING

TABLE OF CONTENTS

1. INTRODUCTION 2
1.1 DESCRIPTION 2
2. QUICK START 3
2.1 LAUNCH PROCESS 3
3. EXPLANATION 4
3.1 GENERAL PRINCIPLE 4
3.1 SENSORS CONFIGURATION 4
3.2 SENSORS VIEWER 4
3.3 VIDEO CAPTURE 4
3.4 NIGHT TEST MANAGER 4
3.5 VISUAL PLUGIN 4
3.6 DEPENDANCIES 4

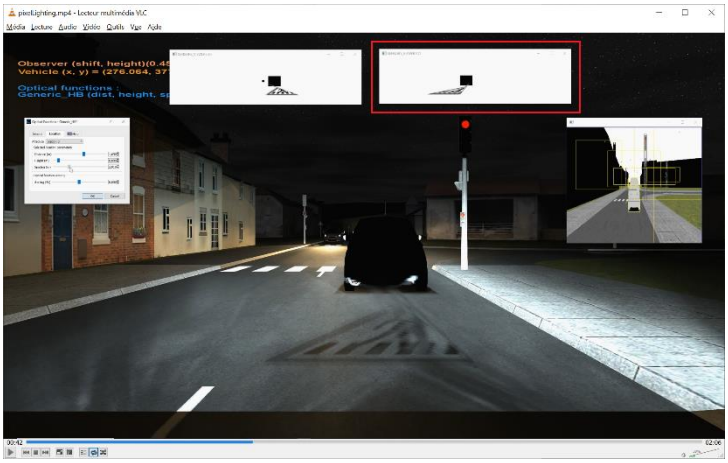
1. INTRODUCTION

1.1 Description

This sample shows a way to modify headlamps photometry in real time in order to simulate a pixel lighting feature.

The current demo produces headlamps with 100% brightness except on detected vehicles, where a 0% brightness rectangle is applied.

(Ultimately the demo will apply a mask to an actual photometry, and add some pictogram display feature – work in progress, preview below)



2. QUICK START

2.1 Launch process

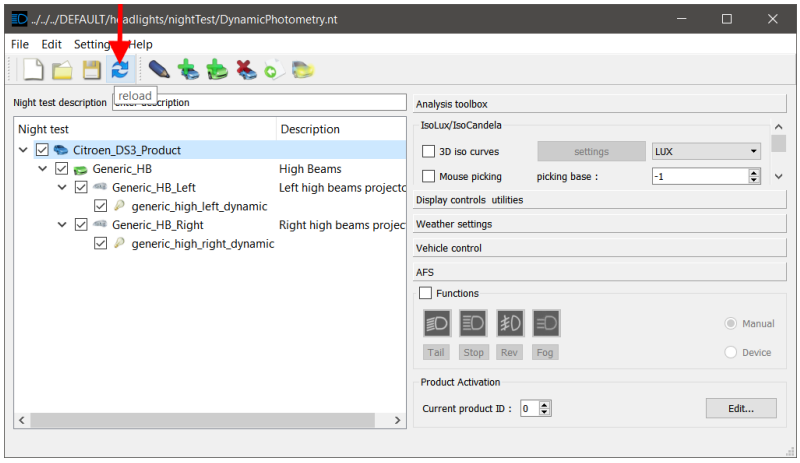
→ Select configuration EVAL_19_PIXEL_LIGHTING
CONFIGURATION > Configuration Manager… > EVAL_19_PIXEL_LIGHTING

The necessary modules start automatically.

→ Open the scenario targets.sce

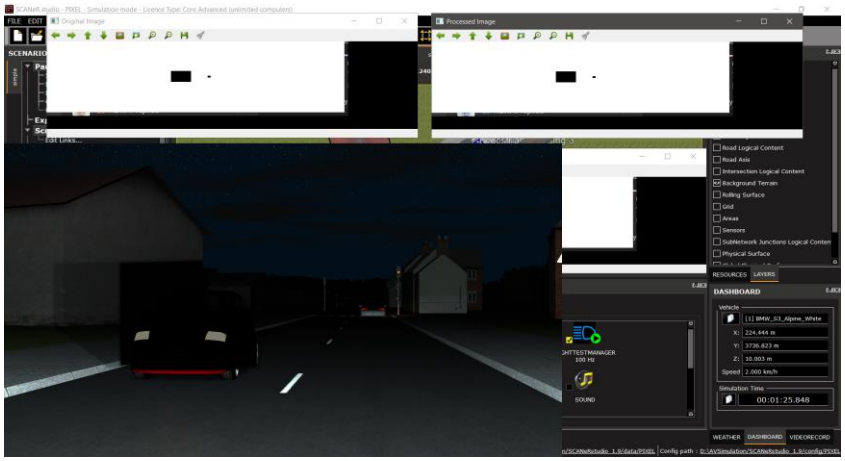
→ Start the simulation

→ In the NIGHTTESTMANAGER window, load the headlamps



→ The result is visible in the VISUAL window

The EGO headlamps output 100% brightness for all the surface, except where other vehicle stand. A “zero brightness rectangle” is applied according to the bounding box from the sensor.



3. EXPLANATION

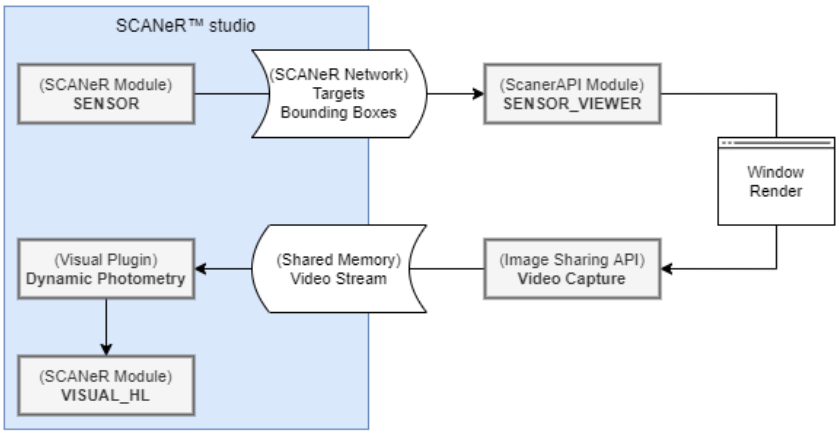
3.1 General principle

#SENSOR: The Ego vehicle sensors configuration includes a radar sensor. Outputs are the bounding boxes of detected vehicles, in the Ego vehicle referential (thus is does not matter where the sensor is placed).

#SENSOR_VIEWER: For each headlamp, a ScanerAPI module reads the SCANeR™ Network for the bounding boxes *and* Ego's headlamps properties. With the headlamps geometry (position), the bounding boxes are transformed to represent a mask of the detected targets *from the headlamp point of view*. The rendering is output in a window.

#Video Capture: The rendered video is captured by the program, and fed in the SCANeR™ Shared Memory using the Image Sharing API.

#Dynamic Photometry: The visual plugin retrieves the video from Shared Memory and applies it to the headlamps, rendered in VISUAL_HL, thanks to the AFS features of NIGHTTESTMANAGER.



3.1 Sensors configuration

3.2 Sensors viewer

3.3 Video Capture

3.4 Night Test Manager

3.5 Visual plugin

3.6 Dependancies

#Sources

- `APIs/samples/evaluation.sln`
  Visual Studio 2013 solution for SCANeR™ Evaluation Data Pack samples
  - `ScanerAPISensorsViewerPixelLighting`
    *Sources for **SENSOR_VIEWER** (ScanerAPI)*
- `APIs/samples/complete.sln`
  Visual Studio 2013 solution for SCANeR™ base samples

- ImageSharing/IS_VideoCapture
  *Sources for **Video Capture** (Image Sharing API)*
- VisualPluginAPI/DynamicPhotometryPlugin
  *Sources for **Dynamic Photometry** (Visual Plugin)*

#Binaries

- APIs\bin\x64\vs2013\
  - ScanerAPISensorsViewerPixelLighting.exe
    Output of project ScanerAPISensorsViewerPixelLighting
  - IS_VideoCapture.exe
    Output of project ImageSharing/IS_VideoCapture
  - plugins\DynamicPhotometryPlugin.dll
    Output of project VisualPluginAPI/DynamicPhotometryPlugin
- bin\x64\plugins\
  Visual plugins have to be here
  - DynamicPhotometryPlugin.dll
    Copied from APIs\bin\x64\vs2013\plugins

#Configuration

- IS_VIDEOCAPTURE_L
  - imageSharingVideoCaptureL.cfg
- IS_VIDEOCAPTURE_R
  - imageSharingVideoCaptureR.cfg
- SENSOR_VIEWER_L
  SENSOR_VIEWER_R
  - SensorViewerPixelLighting.cfg
- VISUAL_HL
  - VisualPlugin.cfg