CHALLENGES IN REAL-TIME RENDERING AND SOFTWARE DESIGN FOR INTERACTIVE IMMERSIVE VISUALIZATION

Andreas Dietrich, ESI Group
Galen Faidley, Caterpillar Inc
March 28, 2018
AGENDA

Introduction
Helios Rendering Architecture (ESI)
End User Implications (Caterpillar)
Demo
INTRODUCTION
“A pioneer and world-leading provider in Virtual Prototyping.”

– www.esi-group.com
Introduction
Virtual Engineering and Design

Combination explosion of visualization scenarios

• Various sources of data
  • Engineering data (e.g., CAD), simulation data (e.g., CFD), measured data (e.g., materials), ...

• Diverse visualization use cases
  • Engineering reviews, exploratory simulation, physically based rendering, scientific visualization, ...

• Different kinds of display devices
  • Desktop, Powerwall, Cave, HMD, mobile devices, ...
Introduction
Helios Architecture Requirements

• Lightweight
  • Not a full-blown scene graph, let application handle this
  • Replicate as few data as possible

• High performance
  • Exploit specific/low-level GPU features
  • Must be able to get close to the metal

• Extensible
  • Leverage external rendering frameworks, such a NVIDIA RiX or OptiX

• Support for various rendering algorithms
  • Rasterization, ray tracing, hybrid modes
  • CAD rendering, photo-realistic rendering, scientific visualization, ...
HELIOS RENDERING ARCHITECTURE
Helios Rendering Architecture

Overview

Application

ICIDO

Helios

Renderer

RiXGL Backend (DLL) ...

OptiX Backend (DLL)
Helios Rendering Architecture

Interfaces

Application API

- Load and unload rendering backends (DLLs can be opened at run-time)
- Switch between backends (e.g., ray tracing or rasterization based)
- Render graph control (e.g., hybrid rendering, frame composition)
- Provide original (unoptimized) scene data

Backend API

- Set scene geometry and transformations (flattened two-level scene graph)
- Set rendering parameters (e.g., materials, lights, whitted ray tracing or GI)
Helios Rendering Architecture

Shadow Data Model

- **Application**
  - Scene hierarchy + original primitives
  - Provides accessor to primitives

- **Helios**
  - Copy of scene hierarchy
  - Does not copy primitives

- **Backend**
  - Flattened scene hierarchy
  - Supported / converted primitives
Graph Optimization
Scene Hierarchy Flattening

Application graph

Backend graph

Transform  Group  Object
Helios Objects
Shared Geometry and Appearance
OptiX Node Graph

Helios Objects as OptiX Subgraph

Selector

Switch scenes

Group

Transform

Geometry Group

Geometry Instance

Material

Acceleration

Transform

Geometry Group

Geometry Instance

Geometry
OptiX Node Graph

OptiX Programs

- Transform
- Geometry Group
- Acceleration
- Geometry
  - Geometry Instance
  - Material
    - ClosestHit Program
    - AnyHit Program
  - Intersection Program
  - BoundingBox Program
class RenderBackEnd
{
    public:
    // Rendering
    virtual void initialize(bool isTest) = 0;
    virtual void initializeWithContext() = 0;
    virtual void shutdown() = 0;
    virtual void render(RenderParams const &) = 0;
    // ... 
    // Scene setup
    virtual void handleObjectAdded(ObjectAddedData const &data) = 0;
    virtual void handleObjectRemoved(ObjectRemovedData const &data) = 0;
    virtual void handleTransformChanged(TransformChangedData const &data) = 0;
    virtual void handleGeometryAdded(GeometryAddedData const &data) = 0;
    virtual void handleGeometryRemoved(GeometryRemovedData const &data) = 0;
    virtual void handleGeometryChanged(GeometryChangedData const &data) = 0;
    virtual void handleGeometryPositionsChanged(GeometryPositionsChangedData const &data) = 0;
    virtual void handleGeometryNormalsChanged(GeometryNormalsChangedData const &data) = 0;
    virtual void handleGeometryModified(GeometryModifiedData const &data) = 0;
    virtual void handleAppearanceTypeAdded(AppearanceTypeAddedData const &data) = 0;
    virtual void handleAppearanceInstanceAdded(AppearanceInstanceAddedData const &data) = 0;
    virtual void handleAppearanceInstanceRemoved(AppearanceInstanceRemovedData const &data) = 0;
    virtual void handleAppearanceChanged(AppearanceChangedData const &data) = 0;
    virtual void handleAppearanceParameterChanged(AppearanceParameterChangedData const &data) = 0;
    virtual void handleAppearanceTextureParameterChanged(AppearanceTextureParameterChangedData const &data) = 0;
    // ... 
    // Rendergraph passes
    virtual void handleClearPass(ClearPassRequestedData const &data) = 0;
    virtual void handleBlitPass(BlitPassRequestedData const &data) = 0;
    virtual void handleExternalFramebufferBlitPass(ExternalFramebufferBlitPassRequestedData const &data) = 0;
    virtual void handleUploadPass(UploadPassRequestedData const &data) = 0;
    virtual void handleDownloadPass(DownloadPassRequestedData const &data) = 0;
    virtual void handleFspPass(FspPassRequestedData const &data) = 0;
    virtual void handleScenePassMono(ScenePassRequestedData const &data) = 0;
    virtual void handleScenePassStereo(ScenePassRequestedData const &data) = 0;
    // ... 
};
 struct ObjectAddedData
{
    sceneId_t sceneId;
    NodeId objectId;
    GeoId geoId;
    InternalAppearanceId appearanceId;
    Mat4 transform;

    ObjectProperties objectProperties;
    bool visible;
};

void OptiXBackEnd::handleObjectAdded(ObjectAddedData const & data)
{
    /* Generate optix node graph for object */

    optix::Context context = getOptiXContext();

    optix::Material material = getOptiXMaterial(appearanceId);
    optix::Geometry geometry = getOptiXGeometry(geoId);

    // Create geometry instance
    optix::GeometryInstance geometryInstance = context->createGeometryInstance();
    geometryInstance->addMaterial(material);
    geometryInstance->setGeometry(geometry);

    // ...
}
RiX Backend

Overview

• Based on RiX::GL (part of NVPro-Pipeline):
  • Rendering API abstraction from OpenGL
  • Hides implementation details which generate OpenGL streams
  • Reduces CPU cost of rendering
  • Can be used in conjunction with NVIDIA VRWorks
    • Single pass stereo, VR SLI, lens matched shading,…

• Supports occlusion culling
Helios RiX Backend

- **MDL Parser**
  - Reads MDL files
  - Generates material parameter lists

- **RiX Backend**
  - (MDL) GLSL Builder
    - Generates GLSL vertex and fragment shaders
    - Stiches (MDL) code snippets

- **MDL SDK**
- **RiX::GL**
- **OpenGL**
Helios RiX Backend
Example: GPU Occlusion Culling

- Raster „Invisible“ bounding boxes
  - Disable color/depth writes
  - Geometry shader creates box sides
  - Fragment shader writes visible[object] = 1
- Only visible objects are drawn next frame
- Bombadier train model:
  - 155,000,000 triangles
  - 200,000 nodes
OptiX Backend

Overview

• Based on NVIDIA OptiX:
  • Programmable GPU ray tracing pipeline
  • Single-ray programming model using C++
  • AI accelerated rendering

• Implements a range of physically based rendering algorithms
  • Whitted ray tracing, ambient occlusion, global illumination
  • Generates precomputed lighting data (e.g., texture baking)

“An Introduction to NVIDIA OptiX”
Detlef Roettger, NVIDIA
— GTC 2018, S8518
Helios OptiX Backend

MDL Parser
- Reads MDL files
- Generates material parameter lists

OptiX Backend

(MDL) CUDA Builder
- Generates material traversers (CUDA C++)
- Compiles bindless callable programs

MDL SDK
OptiX
NVRTC
Helios OptiX Backend

Example: AI Accelerated Global Illumination

- Global illumination path tracer
- Deep neural network based Monte-Carlo noise filtering
- OptiX postprocessing pipeline
Rendergraph

Motivation

• Final image needs to be generated in a multi-pass process
• Multiple backends participate in image computation
• Application of multiple and diverse rendering algorithms
• 2D/3D composition of partial results
• Image postprocessing steps

→ Helios exposes control over the interplay of backends using a graph structure
Rendergraph
Image Composition

- 3D widget (RIXGL)
- Raytraced body (OptiX)
- Rasterized wheels (RIXGL)
- Color postprocessing (RIXGL)
Scaling passes are used to resolve/expand multi-sample targets
A rendergraph can be specified in a text file.

Consists of three parts:

- Definition of render targets
- Definition of render passes
- Definition of links between passes

**rendertarget** TARGET
{
  width : 1920
  height : 1080
  colorFormat0 : RGBA16F
  colorFormat1 : RGBA16F
  depthFormat : D24S8
  buffer : offscreen
  sampleCount : 1
}

**pass_fsq** COLOR_GRADING
{
  backend : rixgl
  shader : fsq_postprocess
  tweakables
  {
    exposure : #f32 0.0 1.0 0.0 #name : type min max default
    white : #f32 0.0 1.0 0.0
    tint : #f32 0.0 1.0 0.0
    brightness : #f32 0.0 1.0 0.0
    saturation : #f32 0.0 1.0 0.0
    contrast : #f32 0.0 1.0 0.0
  }
}

**CLEAR_GL** c0 -> c0 SCENE_GL
**CLEAR_GL** c1 -> c1 SCENE_GL
**CLEAR_GL** d -> d SCENE_GL
**SCENE_GL** c0 -> c0 SCALE_DOWN_S
**SCENE_GL** c1 -> c1 SCALE_DOWN_M
**SCENE_GL** d -> d WIDGET
**SCALE_DOWN_S** c0 -> c0 SCENE_RT
**SCALE_DOWN_M** c1 -> c1 SCENE_RT
**SCENE_RT** c0 -> c0 RECONSTRUCTION
**RECONSTRUCTION** c0 -> c0 COLOR_GRADING
**COLOR_GRADING** c0 -> c0 SCALE_UP
**SCALE_UP** c0 -> c0 WIDGET
**WIDGET** c0 -> c0 SCALE_DOWN_G
**SCALE_DOWN_G** c0 -> c0 GAMMA
Material Handling

Overview

• Helios supports a number of standard material appearances
  • Wood, metal, glass, car paint, ...

• Backend responsible for implementation
  • RiXGL backend: GLSL shaders
  • OptiX backend: CUDA/PTX shaders
  • Shader kernels can be JIT compiled/modified (e.g., include/exclude bump mapping)
  • Separation of BxDF and integrator code (surface shading / light transport)
  • Similar code for GLSL and CUDA results in consistent shading across backends
MDL Materials

Motivation

DEFINITION

VISUALIZATION

MATERIAL SHARING (LIBRARY)

Helios
MDL Materials

Example

• MDL: NVIDIA Material Definition Language
  • Not a shading language
  • Declarative description of material appearance
  • Specification defines BxDF models

```mdl
mdl 1.2;
import df::*;

export material my_diffuse( uniform float par_roughness = 0.0,
                          uniform color par_color = color(0.5)
) =
  material(
    surface: material_surface(
      scattering: df::diffuse_reflection_bsdf(
        roughness: par_roughness,
        tint: par_color
      )
    )
  );
```

Courtesy of NVIDIA
MDL Materials
Preliminary Results

Iray

Helios / RiXGL backend
Challenges in Real-Time Rendering and Software Design for Interactive Immersive Visualization
End User Implications

Galen Faidley
Sr. Engineering Project Team Leader
Caterpillar Inc.

28 March 2018
Agenda

• About Caterpillar
• Immersive Visualization at Caterpillar
• Motivation
• Test setup
• Results
• Implications
• Conclusions
CATERPILLAR: OUR GLOBAL FOOTPRINT

PRODUCT LINE

- Construction
- Mining Equipment
- Diesel & Natural Gas Engines
- Industrial Gas Turbines
- Diesel-Electric Locomotive

We Provide Solutions that Help Our Customers Build a Better World

3 Million+ Products at Work Around the World
98,400 Full-time Employment*
500,000+ Connected Assets*
171 Dealers Serving 192 countries*

*Based on 2017 year-end data
Immersive Visualization at Caterpillar

• Global deployment across Asia, North America, and Europe
• Most product development locations have access to the technology
• Primarily use projection-based systems
• Starting to also use head-mounted displays
• Diverse but focused use cases
Motivation

• For Computer Aided Design (CAD) data - practical performance has not kept pace with theoretical performance
  – Theoretical performance growing approximately with Moore’s law
  – Practical performance flat (not even linear growth)

• CPU bound rather than GPU bound
<table>
<thead>
<tr>
<th>Model</th>
<th>Launch Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>FX 5600</td>
<td>March 2007</td>
<td>342.00</td>
</tr>
<tr>
<td>FX 5800</td>
<td>Nov 2008</td>
<td>342.00</td>
</tr>
<tr>
<td>6000</td>
<td>Dec 2010</td>
<td>377.83</td>
</tr>
<tr>
<td>K6000</td>
<td>July 2013</td>
<td>391.03</td>
</tr>
<tr>
<td>M6000</td>
<td>March 2015</td>
<td>391.03</td>
</tr>
<tr>
<td>P6000</td>
<td>Oct 2016</td>
<td>391.03</td>
</tr>
</tbody>
</table>

*All of NVIDIA's® CUDA®-capable, top-end Quadro® graphics boards*
Actual Numbers

```
fps per generation

D6  D6 + D7  All

fx 5600  fx 5800  5000  K6000  M6000  P6000
```
Test Geometry

D6 Nodes: 42.3K; Triangles: 29.3M

D6, D7 Nodes: 92.7K; Triangles: 70.1 M

All Nodes: 377.7K; Triangles: 262.3 M
Results

IC.IDO Version
10.2 – Baseline (SceneX)
11.0 – Initial Helios Release
11.1 – Improved frustum culling,
       GPU optimized vertex format
11.2 – Occlusion culling

Take away – IC.IDO 11.2 on a P6000 is 9 to 30+ times faster than 10.2 on a Quadro 6000
### Implications

- Product reviews in HMDs are now feasible
- Greater performance overhead to be allocated between:

<table>
<thead>
<tr>
<th>More Data</th>
<th>Better user experience</th>
<th>Less data prep</th>
</tr>
</thead>
<tbody>
<tr>
<td>work sites, mines, factories, multiple full machines</td>
<td>Less simulator sickness, longer immersion times, more natural interaction</td>
<td>No need to simplify large data sets</td>
</tr>
</tbody>
</table>

- Immersive visualization is becoming a more useful tool and a better experience for our users
Conclusions

• A 7+ year old CPU bottle neck has been removed, unleashing the power of modern GPUs for CAD data
  – NVIDIA has shown the world how to address the issue
  – ESI has proven it can be implemented in a robust solution
  – We have confirmed that there is value in the performance gains

• This should be the new normal
  – Users should demand this level of performance
  – Developers need to provide it
Questions?
Caterpillar Inc.

For more than 90 years, Caterpillar Inc. has been making sustainable progress possible and driving positive change on every continent. Customers turn to Caterpillar to help them develop infrastructure, energy and natural resource assets.

With 2017 sales and revenues of $45.462 billion, Caterpillar is the world’s leading manufacturer of construction and mining equipment, diesel and natural gas engines, industrial gas turbines and diesel-electric locomotives. The company principally operates through its three primary segments - Construction Industries, Resource Industries and Energy & Transportation - and also provides financing and related services through its Financial Products segment.

For more information, visit caterpillar.com.

CAT, CATERPILLAR, their respective logos, “Caterpillar Yellow” and the POWER EDGE trade dress, Built For It, as well as corporate and product identity used herein, are trademarks of Caterpillar and may not be used without permission.

©2018 Caterpillar All Rights Reserved
DEMO
QUESTIONS ?
THANK YOU

www.esi-group.com
www.caterpillar.com