Scene Graph Adapter

An efficient Architecture to Improve Interoperability between 3D Formats and 3D Applications Engines

Rozenn Bouville Berthelot - Jérôme Royan (Orange Rennes, France)
Thierry Duval - Bruno Arnaldi (IRISA Rennes, France)
Scene Graph Adapter - Purpose
Goal:
- mix several 3D formats in a single 3D application

Requirements:
- preventive against functionality loss
- extensible to most existing or coming 3D formats
- compliant with most existing or coming 3D application engine
Mix several 3D formats in a single 3D application, why?

1. many, many 3D formats
Issue of 3D format mixing

- Mix several 3D formats in a single 3D application, why?
  1. many, many 3D formats
  2. reuse 3D models
**Issue of 3D format mixing**

- Mix several 3D formats in a single 3D application, why?
  1. many, many 3D formats
  2. reuse 3D models
  3. increase user access to 3D environments
Mix several 3D formats in a single 3D application, why?
1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
Mix several 3D formats in a single 3D application, why?
1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
Mix several 3D formats in a single 3D application, why?
1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
Mix several 3D formats in a single 3D application, why?

1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
Mix several 3D formats in a single 3D application, why?

1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
4. increase industry investment in 3D environment
Mix several 3D formats in a single 3D application, why?

1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
4. increase industry investment in 3D environment
Mix several 3D formats in a single 3D application, why?

1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
4. increase industry investment in 3D environment
Mix several 3D formats in a single 3D application, why?

1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
4. increase industry investment in 3D environment
Mix several 3D formats in a single 3D application, why?
1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
4. increase industry investment in 3D environment
Mix several 3D formats in a single 3D application, why?
1. many, many 3D formats
2. reuse 3D models
3. increase user access to 3D environments
4. increase industry investment in 3D environment

Solution:
allowing interoperability between 3D formats and 3D application engines
3 types of solution as proposed in [Haslhofer and Klas 2010]:

1. **model agreement**
   - WebGL [Khronos Group 2011]
   - X3DOM [Berh et al. 2010]
   - Collada
   - MPEG4
   - OGC
   - ...
3 types of solution as proposed in [Haslhofer and Klas 2010]:

1. model agreement
2. **metamodel agreement**
   - no existing solution
3 types of solution as proposed in [Haslhofer and Klas 2010]:

1. model agreement
2. metamodel agreement
3. **model reconciliation**
   - MPEG-V Part 2 [Gelissen 2008]
   - vWRAP [Bell et al. 2010]
   - PLUG [Hu and Jiang 2008]
Interoperability solutions summary

- None of these solutions allows **reusing** directly existing 3D contents
- Most of them require **transcoding**
- All of them imply format **functionality loss**
Scene Graph Adapter - Overview

- A model reconciliation **interoperability** solution
- Compliant with most **3D formats**
- Compliant with most **rendering components**
Scene Graph Adapter - Overview

- A model reconciliation **interoperability** solution
- Compliant with most **3D formats**
- Compliant with most **3D application engine**

**Web**

- Plugin 1
- Plugin 2
- Plugin 3

**Web Browser**

**Scene Graph Adapter**

- Rendering engine
- Physics engine
- Other engine

**Web browser engine**

**3D application**
Scene Graph Adapter - Architecture
Scene Graph Adapter - Architecture

Format\textsubscript{1} decoder

Format\textsubscript{n} decoder

format scene graph

3D Application

Renderer\textsubscript{x}

API

Renderer\textsubscript{x}
Scene Graph Adapter - Architecture

- Similarities between format decoder and renderer:
  - scene-graph-based structure
  - same model and data organization in scene graph
Scene Graph Adapter - Architecture

- Format_1 decoder
- Format_n decoder

Scene Graph Adapter

3D Application
- RenderAPI
- Renderer_x

format scene graphs
engine scene graph
Scene Graph Adapter - Architecture

- Format\textsubscript{1} decoder
- Format\textsubscript{n} decoder

Scene Graph Adapter:
- Format Adapter API
- Renderer Adapter API
- Node Indexer

3D Application
- Renderer\textsubscript{x}

Format scene graphs

engine scene graph
Scene Graph Adapter - Architecture

- Format_1 decoder
- Format_1 Wrapper
  - Adapter
  - Adaptee
- Format_n Wrapper
  - Adapter
  - Adaptee
- Scene Graph Adapter
  - Format Adapter API
  - Renderer Adapter API
  - Node Indexer
- Renderer_x Wrapper
  - Adaptee
  - Adapter
- 3D Application
  - Renderer_x
  - Renderer_x API

16

Introduction
State of the art
Contribution
Conclusion
Scene Graph Adapter - Architecture

- Format Wrapper
  - Adapter
    - addXNode(...);
  - Adaptee
    - onClick(...){
      ...
    }

- Node Indexer

- Renderer x Wrapper
  - Adaptee
  - Adapter

- 3D Application
  - Renderer x

- Format Wrapper Adapttee Adapter

- Format n Wrapper
  - Adapter
  - Adaptee

16
Scene Graph Adapter - Architecture
Loading a file:
Loading a file:
Loading a file:

- Scene Graph Adapter
- Node Indexer
- Format Adapter API
- Renderer Adapter API

3D Application

Renderer_x

API

Adaptee

Adapter

Renderer_x

Wrapper
Loading a file:
Loading a file:
Loading a file:

- Format₁ decoder
- Format₁ Wrapper
  - Adapter
  - Adaptee

Scene Graph Adapter
- Format Adapter API
- Renderer Adapter API
- Node Indexer

Rendererₓ Wrapper
- Adaptee
- Adapter

3D Application

Format₁ decoder connects to Format₁ Wrapper, which adapts to the Format Adapter API. The Adaptee is then connected to the Node Indexer. The Renderer Adapter API is used to adapt to the Rendererₓ Adapter and API.
Loading a file:

- Scene Graph Adapter
- Node Indexer
- Format₁ Wrapper
- Adapter
- Adaptee

Format₁ decoder

3D Application

Renderer Adapter
API

Rendererₓ Adapter
API

Rendererₓ

Format Adapter
API

Rendererₓ Wrapper
Adaptee
Adapter

Adapter

Adaptee

Format₁ Wrapper
Loading a file:
Loading a file:

- Format_1 decoder
- Format_1 Wrapper
- Adapter
- Adaptee

Scene Graph Adapter

Node Indexer

3D Application

Renderer Adapter

API

Format Adapter

API

Renderer_{x} Wrapper

Adaptee

Adapter

Renderer_{x}

API

Renderer_{x}
Loading a file:

- Scene Graph Adapter
- Node Indexer
- Renderer Adapter
- API
- Adaptee
- Adapter
- Wrapper
- Format 1
- decoder
- Format 1 Wrapper
- Renderer, API
- Renderer, x Adapter
- Renderer, x Wrapper
- 3D Application
Loading a file:

- Scene Graph Adapter
- Node Indexer
- Renderer Adapter
- Format Adapter API
- Renderer x API
- Format 1 Wrapper
- Adapter
- Adaptee
- Format 1 decoder

3D Application

Renderer x

Format 1

decoder
Loading a file:

- Scene Graph Adapter
- Node Indexer
- 3D Application
Loading a file:

- **Format\(_i\) Wrapper**
  - **Format\(_i\) decoder**
  - **Adapter**
  - **Adaptee**

- **Format\(_n\) Wrapper**
  - **Format\(_n\) decoder**
  - **Adapter**
  - **Adaptee**

**Scene Graph Adapter**

**Node Indexer**

**Renderer Adapter**

**API**

**Renderer**

**Renderer\(_x\)**

**Renderer\(_x\) Wrapper**

**3D Application**
Object picking:

- Scene Graph Adapter
- Node Indexer
- Format\textsubscript{1} Wrapper
  - Adapter
  - Adaptee
- Format\textsubscript{n} Wrapper
  - Adapter
  - Adaptee
- Format\textsubscript{1} decoder
- Format\textsubscript{n} decoder
- 3D Application
- Renderer\textsubscript{x}
  - Wrapper
  - Adaptee
  - Adapter
- Renderer\textsubscript{x}, API
- Format Adapter API
- Renderer Adapter API

Functioning 2
Object picking:

- Scene Graph Adapter
  - Format Wrapper (Format<sub>1</sub> Wrapper) Adapter Adaptee
  - Format Wrapper (Format<sub>n</sub> Wrapper) Adapter Adaptee

- Node Indexer

- Renderer Adapter API
  - Renderer<sub>x</sub> Wrapper Adaptee Adapter

- 3D Application

- Format Decoder (Format<sub>1</sub> decoder)
- Format Decoder (Format<sub>n</sub> decoder)
Object picking:

- Scene Graph Adapter
  - Format Adapter API
  - Renderer Adapter API
  - Node Indexer

- Renderer\_x Wrapper
  - Adaptee
  - Adapter

- Format\_1 Wrapper
  - Adapter
  - Adaptee

- Format\_n Wrapper
  - Adapter
  - Adaptee

- 3D Application
  - Renderer\_x

- Format\_1 decoder
  - : ...

- Format\_n decoder
  - : ...
Object picking:

- Format_1 decoder
- Format_n decoder
- Scene Graph Adapter
- Node Indexer
- 3D Application
- Renderer Adapter
- Format Adapter

- Format_1 Wrapper
  - Adapter
  - Adaptee
- Format_n Wrapper
  - Adapter
  - Adaptee
- Renderer_x Wrapper
  - Adaptee
  - Adapter
Object picking:

Scene Graph Adapter

Node Indexer

Format_1 Wrapper
- Adapter
- Adaptee

Format_2 Wrapper
- Adapter
- Adaptee

Format_n Wrapper
- Adapter
- Adaptee

3D Application

Renderer_x Wrapper
- Adaptee
- Adapter

Renderer API

Format Adapter

Renderer Adapter
Object picking:

- Scene Graph Adapter
- Node Indexer
- Format\(_1\) Wrapper
  - Adapter
  - Adaptee
- Format\(_n\) Wrapper
  - Adapter
  - Adaptee

3D Application

Renderer\(_x\)

Renderer\(_x\) Wrapper

Adaptee

Adapter

Format Adapter API

Renderer Adapter API

Format\(_1\) decoder

Format\(_n\) decoder

Renderer Adapter

Adaptee

Adapter
Object picking:

- Scene Graph Adapter
- Node Indexer
- 3D Application
- Format converter
  - Format\textsubscript{1} decoder
  - Format\textsubscript{n} decoder
- Adapter
- Adaptee
- Renderer Adapter API
- Renderer\textsubscript{x} Wrapper
- Renderer\textsubscript{x} API

Diagram showing the flow of data and adapters connecting different components.
Object picking:

- Format_1 decoder
- Format_1 Wrapper
  - Adapter
  - Adaptee
- Format_n decoder
- Format_n Wrapper
  - Adapter
  - Adaptee
- Scene Graph Adapter
  - Adapter
  - Adaptee
- Node Indexer
- Renderer Adapter
  - Adapter
  - Adaptee
- Renderer_x Wrapper
  - Adapter
  - Adaptee
- 3D Application
- **Object picking:**

  ![Diagram showing object picking](image)

  - **Scene Graph Adapter**
    - **Format Adapter API**
    - **Renderer Adapter API**
    - **Node Indexer**

  - **3D Application**
    - **Renderer Adapter**

  - **Format Wrapper**
    - Format$_1$ Wrapper
      - Adapter
      - Adaptee
    - Format$_n$ Wrapper
      - Adapter
      - Adaptee

  - **Renderer$_x$ Wrapper**
    - Adaptee
    - Adapter

  - **Renderer$_x$ API**

  - **Format$_1$ decoder**
  - **Format$_n$ decoder**
Object picking:

- Scene Graph Adapter
- Node Indexer
- Format Wrapper
  - Adapter
  - Adaptee
- Renderer Adapter
- Renderer Wrapper
  - Adapter
  - Adaptee
- 3D Application
- Format Decoder
- Format Wrapper
  - Adapter
  - Adaptee
Object picking:

- Scene Graph Adapter
- Node Indexer
- Format 1 Wrapper
  - Adapter
  - Adaptee
- Format n Wrapper
  - Adapter
  - Adaptee
- Format 1 decoder
- Format n decoder
- 3D Application
- Renderer
- Renderer Adapter
- Format Adapter
- API
- Format Decoder
- Renderer x Wrapper
  - Adaptee
  - Adapter
- Renderer x API
- Renderer x
Object picking:

- Format_1 Wrapper
  - Adapter
  - Adaptee

- Format_n Wrapper
  - Adapter
  - Adaptee

Scene Graph Adapter

Node Indexer

3D Application

Renderer

Renderer Adapter

API

Renderer_{x} Wrapper

Renderer_{x} API
Object picking:

- Scene Graph Adapter
- Node Indexer
- Format Wrapper
- Adapter
- Adaptee
- 3D Application
- Renderer Adapter
- API

Format_1 decoder
Format_n decoder
Format_1 Wrapper
Format_n Wrapper
Renderer_x Wrapper
Renderer_x API
Renderer Adapter
Format Adapter
Node Indexer
Scene Graph Adapter - Node Indexer

Node Indexer

Format Scene Graph 1

Format Scene Graph 2

Engine Scene Graph 1

Engine Scene Graph 2
Scene Graph Adapter - Node Indexer

Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F_1 f_1</th>
<th>E_1 t_1</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>F_1 f_1</td>
<td>E_2 t_2</td>
<td>m</td>
</tr>
<tr>
<td>i</td>
<td>F_2 f_2</td>
<td>E_1 t_1</td>
<td>l</td>
</tr>
</tbody>
</table>

Format Scene Graph 1

Format Scene Graph 2

Engine Scene Graph 1

Engine Scene Graph 2
## Scene Graph Adapter - Node Indexer

### Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F₁ f₁</th>
<th>E₁ t₁</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>F₁ f₁</td>
<td>E₂ t₂</td>
<td>m</td>
</tr>
<tr>
<td>i</td>
<td>F₂ f₂</td>
<td>E₁ t₁</td>
<td>l</td>
</tr>
</tbody>
</table>

### Format Scene Graph 1

- F₁
- i

### Format Scene Graph 2

- F₂
- i

### Engine Scene Graph 1

- E₁
- k
- l

### Engine Scene Graph 2

- m
Scene Graph Adapter - Node Indexer

Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F_1 f_1</th>
<th>E_1 t_1</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>F_1 f_1</td>
<td>E_2 t_2</td>
<td>m</td>
</tr>
<tr>
<td>i</td>
<td>F_2 f_2</td>
<td>E_1 t_1</td>
<td>l</td>
</tr>
</tbody>
</table>

Format Scene Graph 1

Format Scene Graph 2

Engine Scene Graph 1

Engine Scene Graph 2
Scene Graph Adapter - Node Indexer

Format Scene Graph 1

Format Scene Graph 2

Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F1 f1</th>
<th>E1 t1</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>F1 f1</td>
<td>E2 t2</td>
<td>m</td>
</tr>
<tr>
<td>i</td>
<td>F2 f2</td>
<td>E1 t1</td>
<td>l</td>
</tr>
</tbody>
</table>

Engine Scene Graph 1

Engine Scene Graph 2
Scene Graph Adapter - Node Indexer

Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F₁ f₁</th>
<th>E₁ t₁</th>
<th>F₂ f₂</th>
<th>E₁ t₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Format Scene Graph 1

Format Scene Graph 2

Engine Scene Graph 1

Engine Scene Graph 2

Introduction

State of the art

Contribution

Conclusion
Scene Graph Adapter - Node Indexer

Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F₁ f₁</th>
<th>E₁ t₁</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>F₁ f₁</td>
<td>E₂ t₂</td>
<td>m</td>
</tr>
<tr>
<td>i</td>
<td>F₂ f₂</td>
<td>E₁ t₁</td>
<td>l</td>
</tr>
</tbody>
</table>

Format Scene Graph 1

Format Scene Graph 2

Engine Scene Graph 1

Engine Scene Graph 2
Scene Graph Adapter - Node Indexer

Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F₁ f₁</th>
<th>E₁ t₁</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>F₁ f₁</td>
<td>E₂ t₂</td>
<td>m</td>
</tr>
<tr>
<td>i</td>
<td>F₂ f₂</td>
<td>E₁ t₁</td>
<td>l</td>
</tr>
</tbody>
</table>

Format Scene Graph 1

Format Scene Graph 2

F'
Scene Graph Adapter - Node Indexer

Node Indexer

<table>
<thead>
<tr>
<th>i</th>
<th>F1 f1</th>
<th>E1 t1</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>F1 f1</td>
<td>E2 t2</td>
<td>m</td>
</tr>
<tr>
<td>i</td>
<td>F2 f2</td>
<td>E1 t1</td>
<td>l</td>
</tr>
</tbody>
</table>

Format Scene Graph 1

Format Scene Graph 2

Engine Scene Graph 1

Engine Scene Graph 2
APIs details

- **Format Adapter API:**
  - scene graph loading:
    - loadFile, loadNode, ...
  - frame events:
    - setTime, setViewpoint, ...
  - user events:
    - onClick, onDrag, ...
  - ...

- **Renderer Adapter API:**
  - environment settings:
    - setViewpoint, setBackground, setFog, ...
  - camera settings:
    - createCamera, setCamera, ...
  - scene graph settings:
    - createNode, deleteNode, ...
  - geometry settings:
    - createSphere, createPolygonMesh
  - transform settings:
    - createTransformNode, ...
  - material settings:
    - createMaterial, setMaterial, ...
  - ...

Introduction | State of the art | Contribution | Conclusion
Rendering engine:
- Ogre

Input formats:
- X3D
- Collada
Implementation details

Scene Graph Adapter

- X3D Wrapper
  - Adapter
  - Adaptee

- Collada Wrapper
  - Adapter
  - Adaptee

Node Index

- Format Adapter API
- Renderer Adapter API

Ogre Wrapper
- Adaptee
- Adapter

3D Application

- Ogre API

X3D decoder

Collada decoder

Ogre
Implementation details

- **X3D Wrapper**
  - Adapter
    - if (node==SphereNode) {
      - setX3DDefaultValues();
      - createSphereNode(...);
    }
  - Adaptee
    - onClick(int nodeIndex) {
      - updatePointingSensors(...);
    }

- **Scene Graph Adapter**
  - Format Adapter API
  - Renderer Adapter API

- **3D Application**
  - Ogre Wrapper
    - Adaptee
  - Ogre API

- **Collada decoder**
- **X3D decoder**
  - Adaptee

```java
if (node==SphereNode) {
    setX3DDefaultValues();
    createSphereNode(...);
}
```

```java
Adaptee
onClick(int nodeIndex) {
    updatePointingSensors(...);
}
```
Implementation details

Scene Graph Adapter

Node Index

Collada Wrapper

if (node == SphereNode) {
    setColladaDefaultValues();
    createSphereNode(...); ...
}

Adaptee

onClick(int nodeIndex) {};

X3D Wrapper

Adapter

Adaptee

X3D decoder

Collada decoder

3D Application

Ogre Wrapper

Adapter

Adaptee

Ogre

Ogre API

Ogre Wrapper

Adapter

Adaptee

Ogre

Ogre API
Introduction

State of the art

Contribution

Conclusion

Scene Graph Adapter

Node Index

Collada

Wrapper

Adapter

Adaptee

X3D

decoder

X3D

Wrapper

Adapter

Adaptee

Collada

decoder

Collada

Wrapper

Adapter

Adaptee

Ogre

Wrapper

Adaptee

createSphereNode(...){
    Ogre::createEntity(...);
    ...
}

Adapter

onUserEvent( int nodeIndex){
    FormatNodeDescriptor fnd =
        SGA::GetFormatNode(nodeIndex);
    fnd.getFormatWrapper().onClick(fnd.getNodeIndex());
    ...
}
Scene Graph Adapter - Benefits

- Works with every scene-graph-based 3D formats
- Works with every scene-graph-based rendering engine
- Extensible solution
- Reusable components: cumulative effort
Future work

- Use other kinds of engines
  - physics engine, network engine, ...

- Investigate a WebGL-based rendering

- Enable file inlines without formats constraints
  - `<inline url="duck.dae"/>

- Handle interaction between input files
  - use an X3D interpolator to animate a Collada model
Thanks for your attention.

Questions?

Contact: rozennb@gmail.com

Rozenn Bouville Berthelot - Jérôme Royan (Orange Rennes, France)
Thierry Duval - Bruno Arnaldi (IRISA Rennes, France)