Behavior3D:
An XML-Based Framework for 3D Graphics Behavior

Raimund Dachselt, Enrico Rukzio
Dresden University of Technology, Multimedia Technology Group
Outline

- Motivation and Vision
- Related Work
  - X3D: Behavior Definitions and Extensibility
- BEHAVIOR3D
  - Basic Node Concept
  - Declaration, Usage, Implementation
  - Demonstration
- Conclusion & Future Work
Motivation and Vision

Current Situation
- Increasing number of 3D enhanced Web applications
- Need for media-rich and highly interactive content
- Variety of 3D formats, associated modeling and authoring tools

Problems
- Tools & behavior definitions tailored to specific domains
- Limited in producing interactive and dynamic scenes, basically simple animation and behaviors
- Complex behaviors & extensions only through script languages
- Non-programmers remain excluded, authoring still tedious work
- Few concepts of reusing behavior building blocks
Motivation and Vision

Future Vision & Requirements

- Extensible, flexible and unifying description format for 3D graphics behaviors and interactions
- Integrate well into X3D standard
- Rich and extensible set of predefined and classified behavior modules → reuse of high-level 3D Behaviors
- Reduction of programming efforts → declarative format (XML)

CONTIGRA - Framework [Dachselt et al. 2002]

- Document-centered, declarative 3D component architecture
- XML-documents describe interfaces, implementation, configuration, and assembly of components
- High-level view, hides scene graph details, based on X3D
CONTIGRA

XML Schema

CONTIGRA Application

<CoApplication>
3D Scene Description

CONTIGRA Component

<CoComponent>
Component Interface Declaration

CONTIGRA Component Implementation

<CoComponentImplementation>
Scene Graph Integration and Linking
Component Hierarchy

Anthony

Audio
Graph

Geometry
Graph

Behavior
Graph

SceneGraphs

Child Components

CONTIGRA Documents

X3D, Audio3D, Behavior3D
Related Work

- Four levels of behavior [Roehl 1995]
- Independent behavior graph [Döllner & Hinrichs 1998]
- Declarative languages (partly XML-based)
  - VRML97, X3D as a basis: built-in nodes + behavior extensions, e.g. [Seidman 1998]
  - SMIL 2.0 - intuitive time and animation concepts, also sketch of integration into X3D [Kemkes 2001]
  - Viewpoint - scene interactors, state machine paradigm
- Object-Oriented Extensions Working Group [OOE-VRML] and VRML++ [Diehl 1997]
Related Work: VRML97 / X3D

- Built-in behavior-related nodes
  - For defining simple object animations and interactions
    - time, sensors, interpolators, triggers, and sequencers
  - X3D-Components: functionally related X3D objects/nodes
    - Environmental Sensor, Event Utilities, Interpolation, Key device sensor, Networking, Point Device Sensor, Scripting, Time
  - Steps towards node hierarchy: X3D-Schema, SAI
  - Insufficient for complex animations, state-based modeling
Related Work: VRML97 / X3D

<ExternProtoDeclare name="AnimateRotation" url="File.x3d">
  <field accessType="field" name="key" type="Floats"/>
  <field accessType="field" name="to" type="Rotations"/>
  ...
</ExternProtoDeclare>

...

<ProtoInstance name="AnimateRotation">
  < fieldValue name="key" value="0 1"/>
  < fieldValue name="to" value="1 0 0 -1.7, 1 0 0 0"/>
</ProtoInstance>
## Basic Node Concept

### Combinations

<table>
<thead>
<tr>
<th></th>
<th>configurable</th>
<th>receives Events</th>
<th>generates Events</th>
<th>corresponds to X3D field access type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td><strong>outputOnly</strong> (eventOut)</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td><strong>inputOnly</strong> (eventIn)</td>
</tr>
<tr>
<td>4</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td><strong>initializeOnly</strong> (field)</td>
</tr>
<tr>
<td>6</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td><strong>inputOutput</strong> (exposedField)</td>
</tr>
</tbody>
</table>

### Improved field concept:
- name, type, possible default value, 3 change modes
Declaration of new Behavior3D Nodes

- XML Schema grammar *Behavior3DNode*

- **Header:** name, documentation
- **Fields:** none-node datatypes (Color, Rotation)
- **ChildNodes:** node datatypes (TimeBase)
- **UsedNodes:** node composition
<Behavior3DNode>
  <Header name="TimeContainer"/>
  <Interface nodeType="abstract" extends="TimeBase">
    <ChildNodes>
      <Field dataType="TimeBase"
            minOccurs="0" maxOccurs="unbounded">
        <ChangeMode configurable="true" receivesEvents="false"
                     generatesEvents="false"/>
      </Field>
    </ChildNodes>
  </Interface>
</Behavior3DNode>
BEHAVIOR3D - Collections

Collections

- Group functionally and semantically related nodes
- Include all behavior-related X3D nodes
- Completely declared and implemented Collections: 
  StateMachine, Animation

- AnimateTranslation
- AnimateRotation
- AnimateColor
- AnimateCoordinate
- AnimateScalar
- AnimateNormal
- TimeBase {abstract}
- TimeContainer {abstract}
- Sequential
- Parallel
BEHAVIOR3D - Levels

Level

Behavior Node Usage

XML-Grammar

General Scene Graph Grammar (e.g. CONTIGRA, X3D)

XML Schema Behavior3D

conforms to

Scene Graph Instance (e.g. Contigra, X3D)

BehaviorGraph

is generated from all

Behavior3D Node Definitions

XML Schema Behavior3DNode

Node A

Node B

Node C

Collection C1

conform to

Node K

Node L

Node M

Collection C2

XML-Instance

Behavior Node Development
Node Declaration

```xml
<Behavior3DNode>
    <Header name="AnimateRotation"/>
    <Interface nodeType="public" extends="Animation">
        <Fields>
            <Field name="key" dataType="Floats" default="[]">
                <ChangeMode configurable="true" receivesEvents="true" generatesEvents="true"/>
            </Field>
        </Fields>
    </Interface>
</Behavior3DNode>
```
BEHAVIOR3D - Levels

- Resulting Grammar

```xml
<element name="AnimateRotation" type="AnimateRotationType"
    substitutionGroup="Animation"/>

<complexType name="AnimateRotationType">
    <complexContent>
        <extension base="AnimationType">
            <attribute name="key" type="x3d:Floats"/>
            <attribute name="to" type="x3d:Rotations"/>
            <attribute name="by" type="x3d:Rotations"/>
        </extension>
    </complexContent>
</complexType>
```
Node Usage

<Sequential begin="5.0">
  <AnimateRotation key="0 1" to="1 0 0 0, 1 0 0 -1.5"/>
  <AnimateRotation key="0 1" to="1 0 0 -1.5 , 1 0 0 0"/>
</Sequential>
First implementation of Behavior3D nodes with VRML97/X3D
Demo

- Interactive Laptop
  - Entirely realized with Behavior3D nodes
  - Far easier and shorter coding than with X3D
  - Translated to VRML97/X3D with XSLT Stylesheets

```xml
<AnimateRotation DEF="OpenLaptop"
    key="0 1" to="1 0 0 0, 1 0 0 -1.7"/>

<Sequential DEF="OpenKeyboard">
  <AnimateTranslation DEF="Open_Translation"
      key="0 1" to="0 0 0, 0 0.05 0" />
  <AnimateRotation DEF="Open_Rotation"
      key="0 1" to="1 0 0 0, 1 0 0 -1.5" />
</Sequential>
```
<StateMachine stateCount="3" transitions=""
  1 2  LCD_Sensor.touchTime  OpenLaptop.startTime,
  2 1  LCD_Sensor.touchTime  CloseLaptop.startTime,
  2 3  Keyboard_Sensor.touchTime OpenKeyboard.startTime,
  3 2  Keyboard_Sensor.touchTime CloseKeyboard.startTime"/>

<bno:TouchSensor DEF="LCD_Sensor"/>
<bno:TouchSensor DEF="Keyboard_Sensor"/>
Conclusion & Future Work

Major Features

- Inheritance, strong typing, polymorphism
- Easy definition of new first-class nodes
- Automated implementation-code generation
- Smooth language integration through novel grammar generation mechanism

Future Work

- Visual Authoring tool for editing 3D graphics behavior
- Sets of predefined behavior nodes (collections) to be extended. Candidates for X3D-components?
- Dynamic scene graph grammar generation for X3D?
Discussion

Thank you for your attention!

www.CONTIGRA.com

Contact: dachselt@acm.org
Translation