
# GraphiCon'98 Tutorial:

# Introduction to VRML 97

Ralf Dörner, Colette Elcacho, Arno Schäfer
Fraunhofer Institute for Computer Graphics
Darmstadt, Germany


# Outline (I)

| | |
|---|---|
| 15 min | Introduction<br>Overview |
| 60 min | VRML'97 Basic Concepts |


# Outline (II)

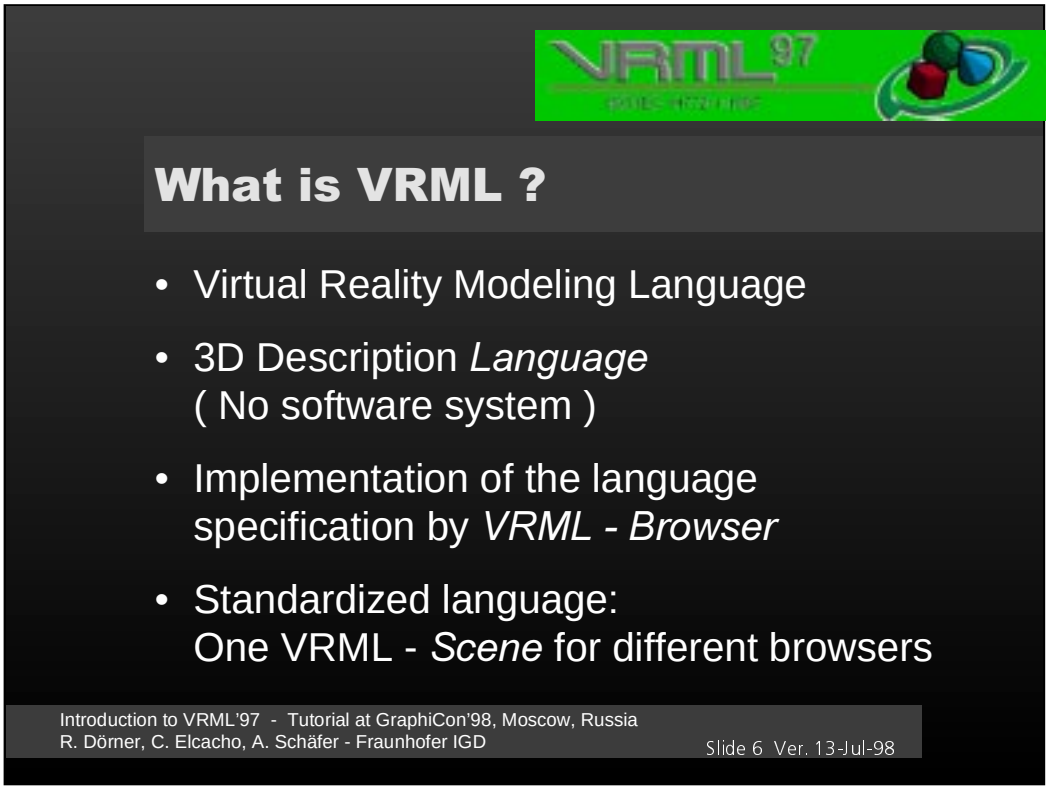
| | |
|---|---|
| 20 min | VRML'97 Features |
| 60 min | VRML and Java:<br>Programming of 3D Worlds |


# Outline (III)

25 min

VRML Browsers and Tools
Current Developments
Literature
Questions & Answers


# INTRODUCTION


VRML 97
What is VRML ?
• Virtual Reality Modeling Language
• 3D Description *Language*
( No software system )
• Implementation of the language
specification by *VRML - Browser*
• Standardized language:
One VRML - *Scene* for different browsers
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 6 Ver. 13-Jul-98

# VRML Example (I)

# VRML Example (II)


# VRML: History (I)

- 1994 Mark Pesce, Tony Parisi, Gavin Bell start with the idea of extending the Internet standard HTML in San Francisco
- Mid 1995: VRML 1.0
  based on SGI's Open Inventor after an "Internet vote" descision
- Start of the support from SGI, Netscape and Microsoft

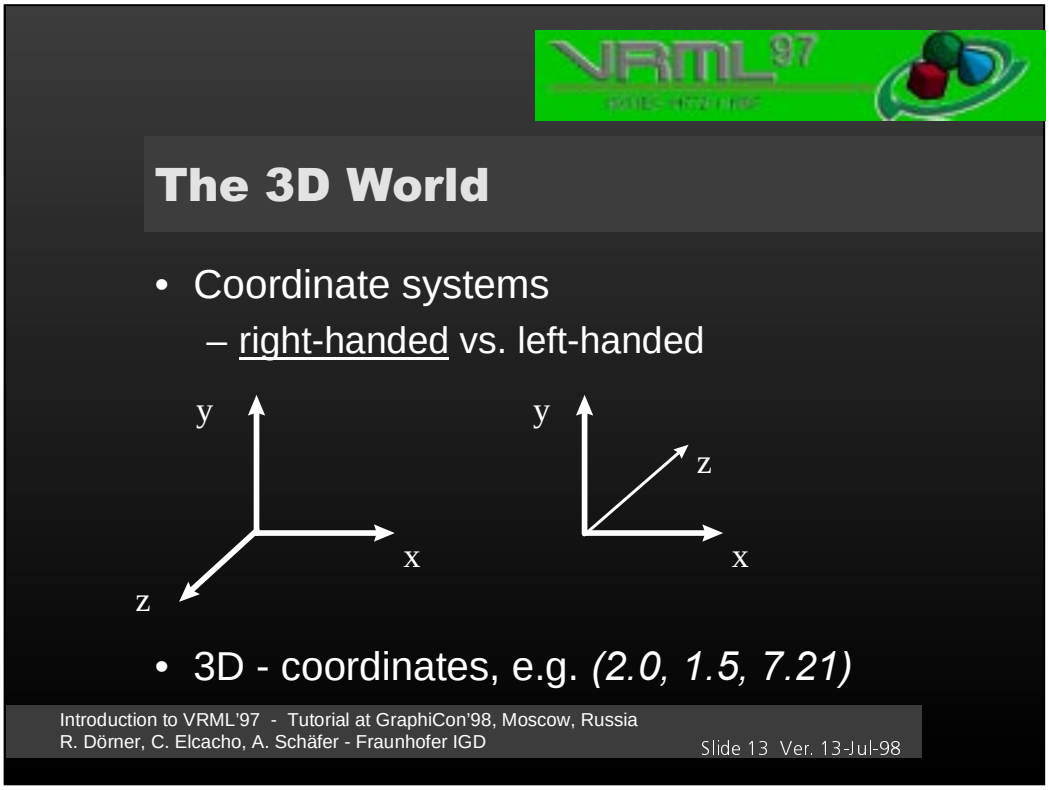

# VRML: History (II)

- VRML Architecture Group (VAG) founded at SIGGRAPH'95
- Spring 1996 Call for proposals concerning VRML 2.0 by the VAG
- VRML 2.0 specification release at SIGGRAPH'96 after open vote (based on SGI's Moving Worlds proposal)
- Foundation of the VRML consortium


# VRML: History (III)

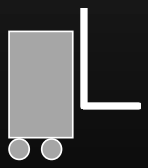
- Early 1997 Start of ISO - standardization (“VRML 97”)
- End 1997 VRML 97 is standardized as ISO/IEC DIS 14772-1


VRML 97
ISO/IEC 14772-1:1997
BASIC CONCEPTS
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 12 Ver. 13-Jul-98


VRML 97
The 3D World
• Coordinate systems
– right-handed vs. left-handed
y
x
z
y
z
x
• 3D - coordinates, e.g. (2.0, 1.5, 7.21)
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 13 Ver. 13-Jul-98


# Global vs. Local Coordinates

- Hierarchy of coordinate systems
- Example: Fork of forklift truck
  - Coordinates of fork with regard to truck
  - Coordinates of truck with regard to world



- Top of the hierarchy: World Coordinates


# The Scene Graph

- Visualization of the scene hierarchy
- Edges: Dependency relation
- Nodes:
  - Geometry
  - Transformations
  - Material attributes
  - ...
- Fields


# A VRML File

```
#VRML V2.0 utf8
Shape {
        appearance      Appearance {
                                material Material { }
                        }
        geometry        Cone {
                                bottomRadius  2.4
                                height 5.0
                        }
        }
```


# Structure of a VRML File (I)

- VRML Header
  - Version
  - Character Set,
    e.g. UTF-8 ( ISO 10646-1:1993 )
- Line comments (Beginn mit #)
- VRML nodes and fields
- Relations with curly braces { }


# Grouping of Nodes

- Combination of nodes to a group: Group Node
- *Example:*

```
Group{
        children [
                Shape{ ...}
                Shape{ ...}
        ]
}
```


# Transformations (I)

- Translation
  Translation vector: *( x, y, z )*


# Transformations (II)

- Rotation
  - Rotation axis: *( x, y, z )*
    Note: only direction relevant
  - Rotation angle: φ
    Note: sign of angle is determined with the "Right-Hand-Rule"
    Note: Angels are measured in radian
    ( π = 3, 14... entsprechen 180° )
  - Rotation center: *( x, y, z )*


# Transformations (III)

- Scaling
  - Scaling factors: ( $s_x$ , $s_y$ , $s_z$ )
  - Rotation axis: ( x, y, z )
  - Rotation angle: φ

  Note:
  1. Scaling rotation
  2. Scaling
  3. Back rotation


# Transform Node

- Representation of transformations in VRML by the use of a *Node*
- Values for specifying the transformation are provided in *Fields*
  *Example:* rotation 4.0 0.0 0.0 2.37
- Special field with list of nodes affected by the transformation
  *Example:* children [ ]


# Transform Node: An Example

```
Transform{
   children [  Shape {
                        appearance Appearance{
                                material Material { }
   }
                        geometry    Box { }  }
            ]
   translation  0.0  4.0  0.3
   rotation     1.0  0.0  1.0  1.57  }
```


# VRML Types (I)

- Single Field Values (SF)
  Multiple Field Values (MF)
- SFBool TRUE, FALSE
- SFInt32 42
  MFInt32
- SFFloat -124.567
  MFFloat


# VRML Types (II)

- SFString MFString "forklift truck"
- SFTime 65 (specifies 12 a.m. 1 Minute 5 Sec. GMT on 1.1.1970)
- SFNode MFNode Transform
- SFImage


# VRML Types (III)

- SFVec2f MFVec2f — 1.3 4.5
- SFVec3f MFVec3f — 34.5 -4.9 9.0
- SFRotation MFRotation — 1.0 1.0 4.0 3.2
- SFColor MFColor — 1.0 0.0 0.0


# DEF - USE Mechanism

- Nodes may be named
  *Example:* DEF my_box Box { ... }
- Names
  - consist of letters, digits and underscore
  - start with capital letter
  - distinguish capitilization
  - Nodes may be used arbitrary times
    *Example:* USE my_box


# Reserved Names in VRML

| | | |
|---|---|---|
| DEF | EXTERNPROTO | FALSE |
| IS | eventIn | TRUE |
| TO | eventOut | PROTO |
| NULL | exposedField | ROUTE |
| USE | field | |


# Events and Routes (I)

- Event
  - Change of a value
  - User interaction
- Route
  - Connection of two nodes in order to exchange events
  - Start node (source)
  - End node (drain)


# Events and Routes (II)

- Routes are related to a field of a node
- Field categories:
  - eventIn
  - eventOut
  - exposedField
    - set_XXX
    - XXX_changed
- Events and routes are typed


# Routing: Example

```
DEF myCube Transform{
                    ...
                }
DEF myBox   Transform{
                    ...
                }
ROUTE myCube.translation_changed TO
        myBox.set_scale
```


# Syntax of VRML Nodes

- Node name
- Field list
- For each field
  - Name
  - Defaultvalue
  - field, eventIn, eventOut, exposedField
  - Type


# Example: TimeSensor Syntax

TimeSensor

| | | | |
|---|---|---|---|
| exposedField | SFBool | enabled | TRUE |
| exposedField | SFTime | startTime | 0 |
| exposedField | SFTime | stopTime | 0 |
| exposedField | SFTime | cycleInterval | 1 |
| exposedField | SFBool | loop | FALSE |
| eventOut | SFBool | isActive | |
| eventOut | SFBool | time | |
| eventOut | SFTime | cycleTime | |
| eventOut | SFFloat | fraction_changed | |


# TimeSensors

| | | |
|---|---|---|
| loop = TRUE | stopTime <= startTime | Endless cycles |
| loop = TRUE | startTime < stopTime | Cycles till stopTime |
| loop = FALSE | stopTime <= startTime | 1 Cycle, Stop at startTime + cycleInterval |
| loop = FALSE | startTime < stopTime | 1 Cycle, Stop at startTime + cycleInterval or stopTime, if it is earlier |


# Animation

- Timer
- Interpolator
  - Positioninterpolator
  - Colorinterpolator
  - Scalarinterpolator
  - ...
- Routing: Timer > Interpolator
  Interpolator > Animated Value


# PositionInterpolator

- Mapping of a value out of an interval (usually [0.0, 1.0] ) to a position
- Input: set_fraction (SFFloat)
- Output: value_changed (SFVec3f)
- Calculation of values using linear interpolation based on a table
  - key [ 0.0, 1.0 ]
  - keyValue [ 1.0 0.0 0.0, 5.0 0.0 0.0 ]


# Routing Example for Animation

*Box* is a Transform Node with Shape

*BoxPath* is a PositionInterpolator Node

*Timer* is a TimeSensor Node

ROUTE Timer.fraction_changed TO BoxPath.set_fraction

ROUTE BoxPath.value_changed TO Box.set_translation


# Inlines

- Including VRML files in a VRML file
- Example:
  Inline {
  url “example.wrl”
  }
- Different name space concerning DEF / USE


# Prototypes (I)

- Encapsulation of parts of a scene graph
- Own node definition
- Parameterizable
- Syntax:
  PROTO *name* [ *interface* ] { *body* }
- Building a connection between *interface* and *body* using IS syntax


# Prototypes (II)

```
PROTO  Box [
   field SFVec3f dimension 1.0 1.0 1.0
] {
   Shape { appearence Appearence{
                                material Material{ } }
           geometry Box{
                                size IS dimension } }
}
```


# Prototypes (III)

- DEF / USE has its own name space
- Connection interface and body

| | | Prototyp-Declaration | | | |
|---|---|---|---|---|---|
| | | exposedField | field | eventIn | eventOut |
| Prototyp-Definition | exposedField | + | + | + | + |
| | field | - | + | - | - |
| | eventIn | - | - | + | - |
| | eventOut | - | - | - | + |

- Prototypes may be nested


# External Prototypes

- Syntax:
  EXTERNPROTO *name* [ *interface* ] { *urls* }
- Creation of prototype libraries
- Example:

```
EXTERNPROTO Box [
        field SFVec3f dimension 1.0 1.0 1.0
] {
        “lib.wrl#Box”
}
```


# Structure of a VRML File (II)

- VRML - Header
- Comments
- Prototype - Definitions
- Nodes and Fields
- Routes


# VRML Basics: Conclusion

- Scene graph concept:
  Nodes and Fields
- Routing concept:
  EventIn, EventOut, Routes
- VRML Syntax:
  Data types, field types
- VRML file structure
- Naming mechanism, Prototypes, Inlining


VRML 97
ISO/IEC 14772-1:1997
VRML FEATURES
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 45 Ver. 13-Jul-98


# Features: Geometry (I)

- Predefined Shapes
  - Cube
  - Cylinder
  - ...
- Polygonal Shapes
  - Coordinate node
  - Line Sets / Indexed Line Sets
  - Face Sets / Indexed Face Sets


# Features: Geometry (II)

- Elevation Grids
  (esp. for terrains)
- Extrusion nodes
  (extruding cross sections along a spine)
- Level of detail (LOD node)


# Features: Appearance

- Material node
  - Specularity, Shininess, ...
  - Transparency
  - Color
- Appearance node
  - Material
  - Texture


# Features: Texturing

- ImageTexture node
- PixelTexture node
- MovieTexture node
- TextureCoordinate node
- TextureTransform node (controls mapping)


# Features: Text

- Text node
  - Maximum extent, length
  - Text itself
- FontStyle node
  - Font family, style
  - Spacing
  - Justification


# Features: Transformation

- Rotation
- Scaling
- Positioning


# Features: Grouping Nodes

- Group node
- Transform node
- Switch node
  (activates different parts of scene graph)
- Billboard node
  (groups z-axis always points to viewer)


# Features: Animation

- TimeSensor
- Interpolators
  - PositionInterpolator
  - ColorInterpolator
  - CoordinateInterpolator
  - OrientationInterpolator
  - ScalarInterpolator


# Features: Sensors

- TouchSensor node
- PlaneSensor node
- SphereSensor node
- CylinderSensor node
- VisibilitySensor node
- ProximitySensor node
- Collision node


# Features: Environment

- Background node
  - Sky angle (upper half sphere)
  - Ground angle (lower half sphere)
  - Color gradients
- Fog node


# Features: Lights

- Light sources:
  - PointLight node
  - DirectionalLight node
  - SpotLight node
- Default light (Headlight) mounted to the viewer
- NO shadows


# Features: Shading

- Normal node
- NormalInterpolator node


# Features: Viewer and Infos

- Viewpoint node
  (defines viewer position and view)
- NavigationInfo node
  (type, speed, size of viewer avatar)
- WorldInfo node
  (title, info, e.g. copyright info)


# Features: Sound

- AudioClip node
  (sound source, duration, ...)
- Sound node
  (intensity, range, spatialization, ...)


# Features: Hypermedia

- Anchor node
- Semantic: if user clicks shape then new world is loaded


# Features: Reusing

- Inline node
- PROTO construct
- EXTERNPROTO construct


# Features: Program logic

- Script node
- Including Java or ECMAScript programs


# VRML-Features: Conclusion

- VRML is a description language for interactive 3D worlds
- VRML integrates animation, multimedia and hypermedia
- VRML may be transmitted via Internet / WWW
- VRML may be implemented using immersive technologies


VRML 97
ISO/IEC 14772-1:1997
VRML & PROGRAMMING
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 64 Ver. 13-Jul-98


# VRML and Program Logic (I)

- Integration of programs of arbitrary complexity
- Programs may manipulate the VRML scene
- Application examples
  - Multi-User Systems
  - Database Linkage
  - Simulations

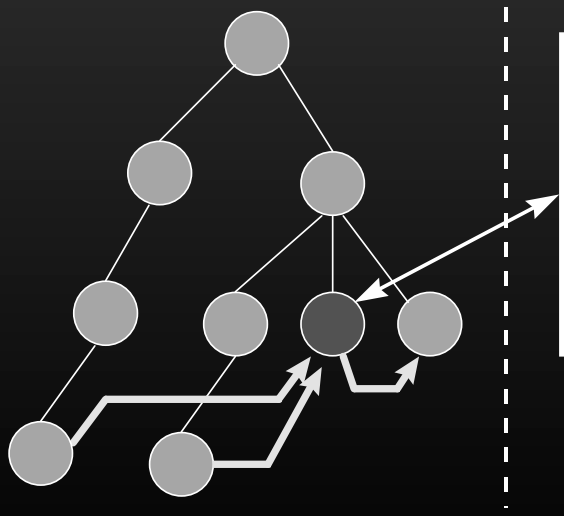

# VRML and Program Logic (II)

- Two different approaches
  - VRML Scripting Interface
  - External Authoring Interface (EAI)


# VRML Scripting

- Embedding of programmed behavior in a VRML scene
- Supported programming languages:
  - Java
  - ECMAScript (JavaScript)
- Programming interfaces (“APIs”) are part of the VRML specification

# VRML Scripting: Idea



**myScript.java:**

```
public class myScript extends Script {
    public void initialize () {
        // ...
    }

    public void processEvent (Event e) {
        // ...
    }
}
```


# VRML Scripting: Idea

- On the VRML side:
  Script Node with
  - user-defined interface analogous to PROTO definitions
  - reference to the Java program
- On the Java side:
  - dedicated API for communication with the VRML scene


# The Script Node: Syntax

```
Script {
   exposedField MFString url          []
   field        SFBool   directOutput FALSE
   field        SFBool   mustEvaluate FALSE

   # and an arbitrary number of
   eventIn      eventType eventName
   field        fieldType fieldName initialValue
   eventOut     eventType eventName
}
```


# The url-Feld

- Reference to an external file

```
url “http://bla.fasel.de/meinScript.class”
url “meinScript.js”
```

- Inline-Program

```
url “javascript: function bla() {...}”
```

- Alternative URLs

```
url [ “javascript: ...”
  “java_version.class”
]
```

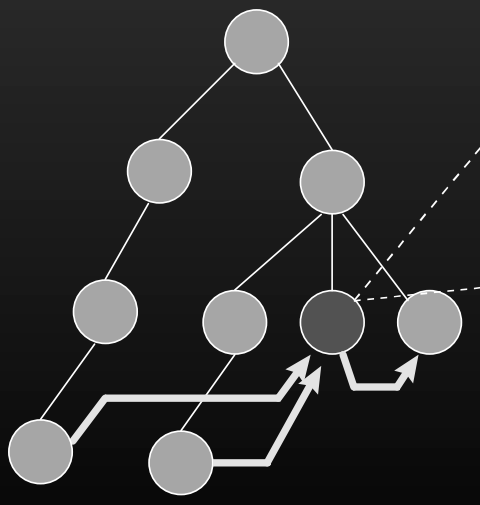

# directOutput und mustEvaluate

- directOutput
  - has to be TRUE if the script manipulates nodes directly (instead of using routing mechanism)
- mustEvaluate:
  - if FALSE, browser may defer event delivery to the script under special circumstances


# Definable Fields

- In a Script node an arbitrary number of fields, eventIns and eventOuts (no exposedFields) can be defined
- *Fields* can be read and modified by the script
- *Fields* are therefore well suited for parameterizing of the script

# Script Node Example (I)



```
Script {
    url "myScript.class"
    field SFFloat height 10.0
    eventIn SFBool click
    eventOut SFTime start
}
```


# Script Node Example (II)

```
Script {
   url "myScript.class"

   # field typ name wert
   field SFFloat height 10.0
   # eventIn typ name
   eventIn SFBool click
   # eventOut type name
   eventOut SFTime start
}
```


# The Java Programming Interface

- Specifies the interaction between Java programs refered in Script nodes and the browser / scene
- Currently supports Java 1.0
- Enables the use of the complete Java functionality, such as network or GUI functions


# Data Types

- Every VRML data type has two corresponding Java classes:
  - Read Only Type
    e.g. ConstMFString
  - Read / Write Type
    e.g. SFTime


# Read Only Types

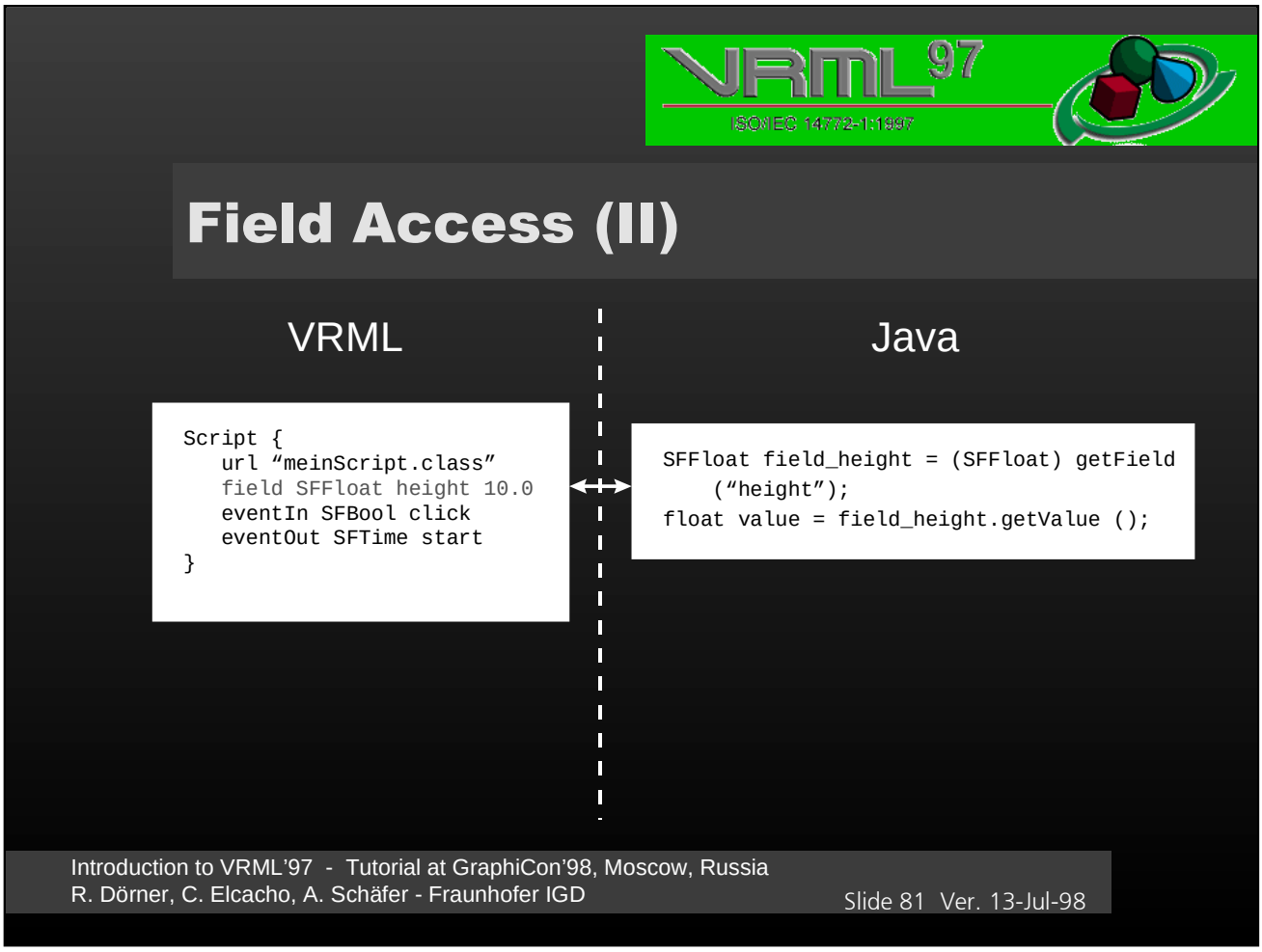
```
public abstract class ConstMField extends ConstField {
   public abstract int getSize();
}

public class ConstMFString extends ConstMField {
    public ConstMFString(int size, String s[]);
    public ConstMFString(String s[]);

    public void getValue(String values[]);
    public String get1Value(int index);
    public String toString();
}
```


# Read / Write Types

```
public class SFTime extends Field {
   public SFTime();
   public SFTime(double time);

   public double getValue();

   public void setValue(double time);
   public void setValue(ConstSFTime time);
   public void setValue(SFTime time);

   public String toString();
}
```


# Field Access (I)

- The fields defined in the Script node can be accessed using the “getField”, “getEventIn”, and “getEventOut” methods of the Script class (type casting may be necessary)
- The values of the fields can be read using the respective “getValue” methods


VRML 97
ISO/IEC 14772-1:1997
Field Access (II)
VRML
Java
Script {
url "meinScript.class"
field SFFloat height 10.0
eventIn SFBool click
eventOut SFTime start
}
SFFloat field_height = (SFFloat) getField
("height");
float value = field_height.getValue ();
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 81 Ver. 13-Jul-98


# Field Access: Example

```
// SF-Felder
SFFloat field_height = (SFFloat) getField
  (“height”);
float value = field_height.getValue ();

// MF-Felder
MFString field_strings = (MFString) getField
  (“strings”);
String[] value2 =
  new String[field_strings.getSize ()];
field_strings.getValue (value2);
```


# Initialization

- In the script a method “initialize” can be defined, that is called by the Browser before the script receives the first event
- Often the initialize method creates references to the fields and event interfaces defined in the script node and stores them for later use


# Initialization: Example

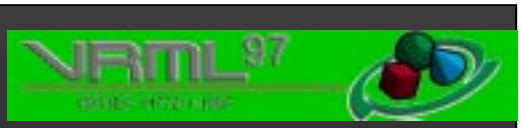
```
import vrml.*;
import vrml.node.*;
import vrml.field.*;

public class myScript extends Script {

   public void initialize () {
        SFFloat height = (SFFloat) getField (“height”);
        float value = height.getValue ();

        // ...
   }
}
```


# Manipulation of a VRML Scene

- Reading of data
- Reception of events
- Sending of events and changing values
  - Manipulation via routing mechanism
  - Direct manipulation
- Dynamic scene graph manipulation
  - adding / deleting nodes
  - adding / deleting routes


## Script EventIns

- EventIns of a Script node are connected with the scene via ROUTEs
- When an event is passed via a ROUTE into an Script node, the “processEvent” method of the script is called



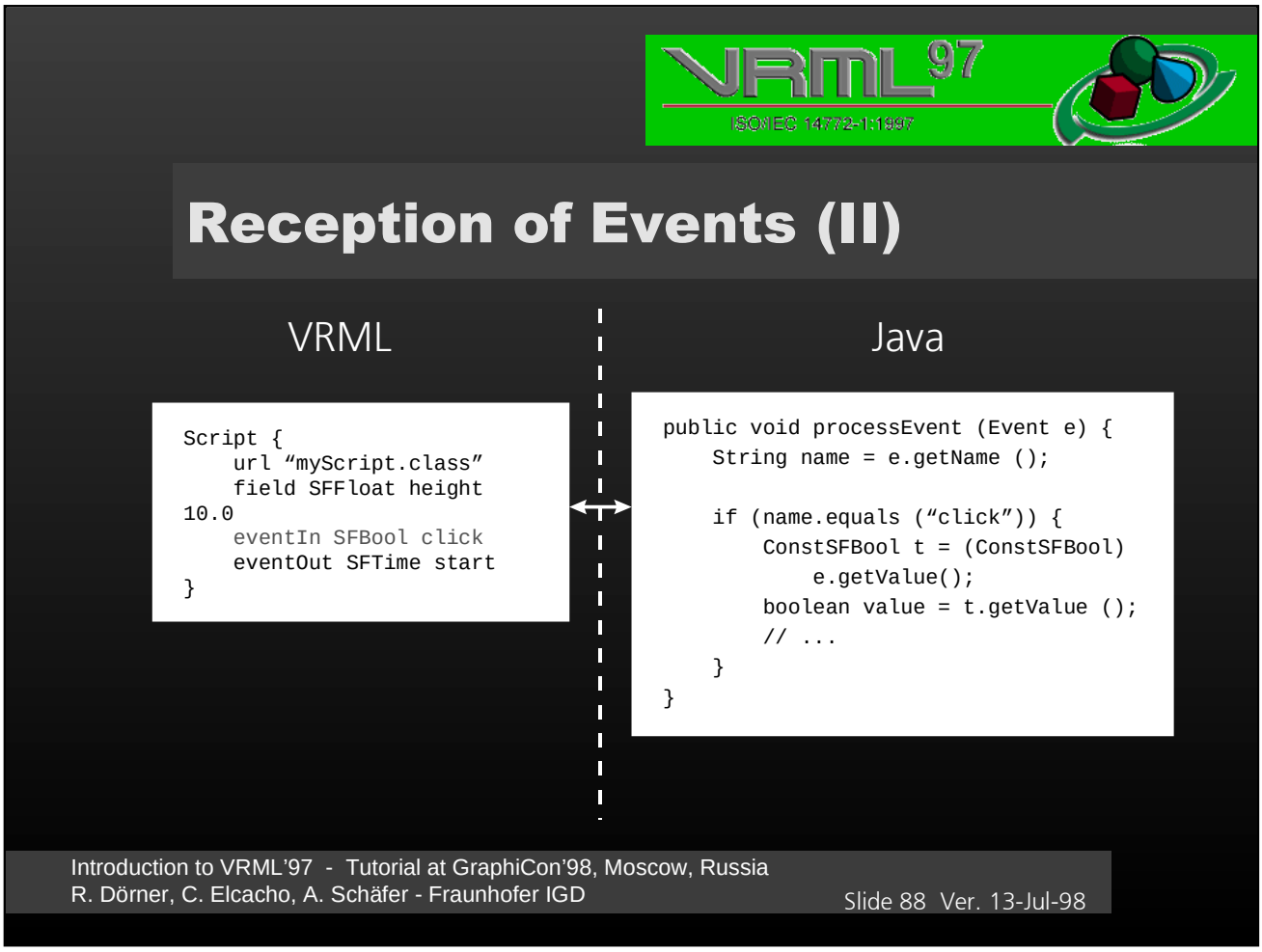

# Reception of Events (I)

- Methods

```
public void processEvent (Event e) {}
```

- Event Class

```
public class Event {
    public String getName ();
    public double getTimeStamp ();
    public ConstField getValue ();
}
```

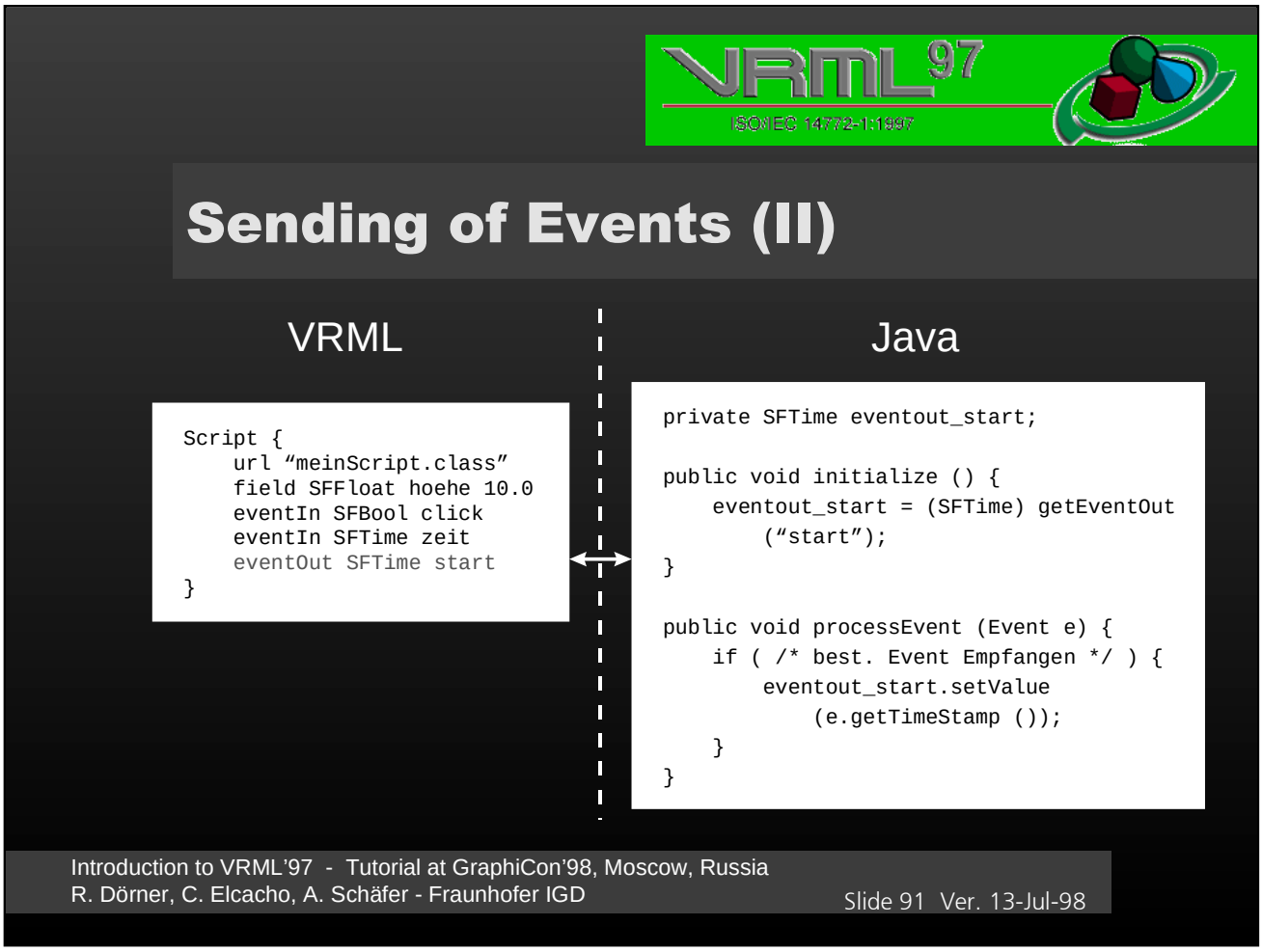

VRML97
ISO/IEC 14772-1:1997
Reception of Events (II)
VRML
Java
Script {
url "myScript.class"
field SFFloat height
10.0
eventIn SFBool click
eventOut SFTime start
}
public void processEvent (Event e) {
String name = e.getName ();
if (name.equals ("click")) {
ConstSFBool t = (ConstSFBool)
e.getValue();
boolean value = t.getValue ();
// ...
}
}
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 88 Ver. 13-Jul-98


# Reception of Events: Example

```
public void processEvent (Event e) {
   String name = e.getName ();

   if (name.equals (“click”)) {
        ConstSFBool t = (ConstSFBool) e.getValue();
        if (t.getValue () == false) {
               // ...
        }
   }
   else if (name.equals (“foo...”)) {
        // ...
   }
}
```


# Sending of Events (I)

- eventOuts of a Script node are connected with the scene via ROUTEs
- The script gets a pointer to the eventOut using the “getEventOut” method
- By calling “setValue” on the eventOut object, the event is sent to the scene


VRML 97
ISO/IEC 14772-1:1997
Sending of Events (II)
VRML
Java
Script {
url "meinScript.class"
field SFFloat hoehe 10.0
eventIn SFBool click
eventIn SFTime zeit
eventOut SFTime start
}
private SFTime eventout_start;
public void initialize () {
eventout_start = (SFTime) getEventOut
("start");
}
public void processEvent (Event e) {
if ( /* best. Event Empfangen */ ) {
eventout_start.setValue
(e.getTimeStamp ());
}
}
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 91 Ver. 13-Jul-98


# Sending of Events (III)

- EventOut declaration in Script node

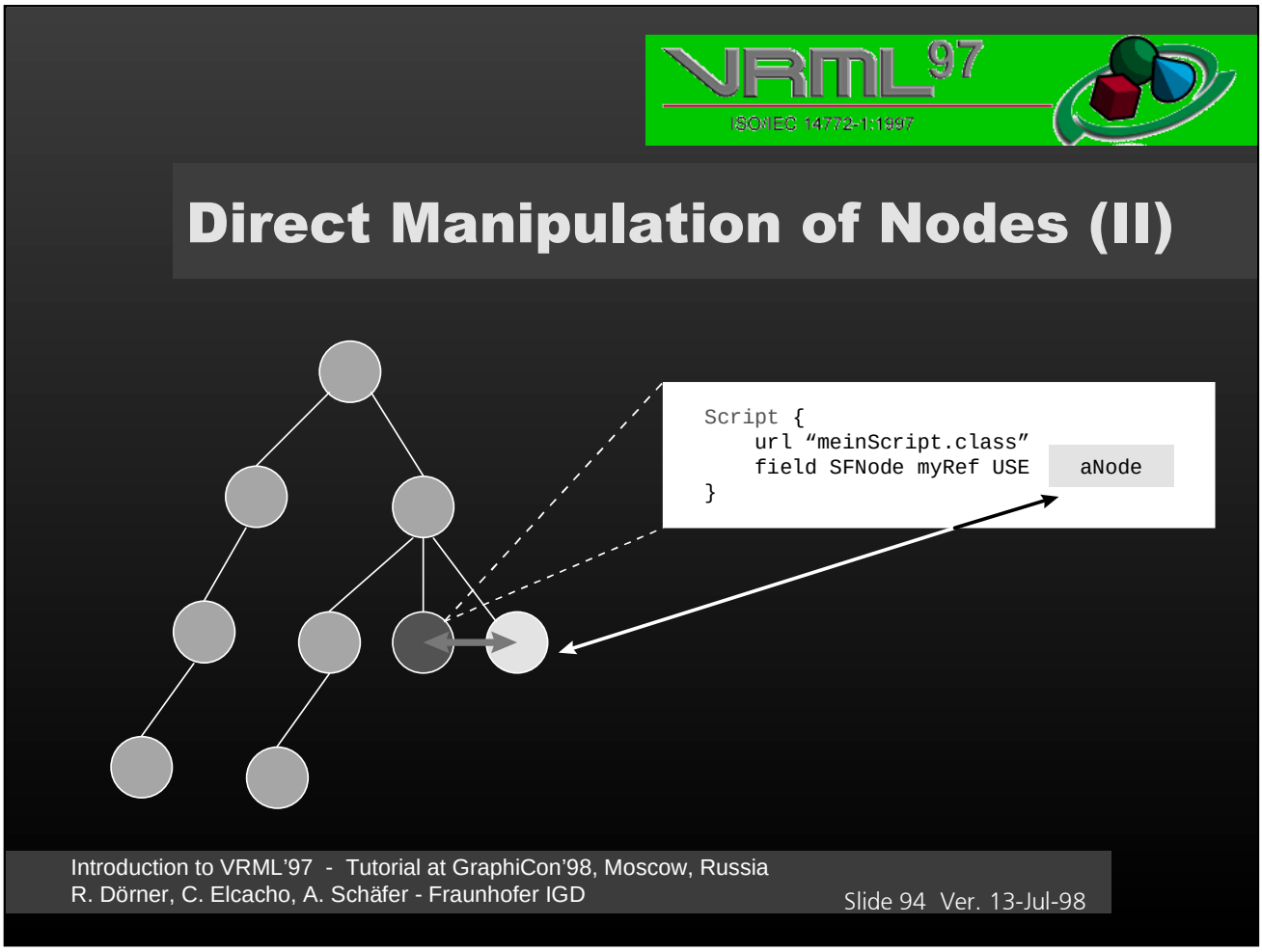
```
eventOut SFTime start
```

- Manipulation in script-code

```
private SFTime eventout_start;
public void initialize () {
    eventout_start = (SFTime) getEventOut (“start”);
}
public void processEvent (Event e) {
    if ( /* certain event received */ ) {
        eventout_start.setValue (e.getTimeStamp ());
    }
}
```


# Direct Manipulation of Nodes (I)

- Sending events to a node without routing
- directOutput - field of Script node needs to be TRUE
- Script needs to have reference to the node
- Reference to the node may be obtained for example via a SFNode-Field where “USE - references” to another node are stored


VRML 97
ISO/IEC 14772-1:1997
Direct Manipulation of Nodes (II)
Script {
url "meinScript.class"
field SFNode myRef USE aNode
}
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 94 Ver. 13-Jul-98


# Direct Manipulation of Nodes (III)

- Fields/EventIns of a node are read using the “getExposedField” or “getEventIn” method respectively
- The “children” - field of a Group node offers the possibility to access its child nodes and thus to traverse the whole scene graph


# Direct Manipulation of Nodes (IV)

- Field declaration in a script node

```
field SFNode light USE aLightNode
directOutput TRUE
```

- Manipulation in the script code

```
SFNode f_light = (SFNode) getField (“light”);
Node lightNode = (Node) f_light.getValue ();
SFBool f_on = (SFBool) lightNode.getExposedField
  (“on”);
boolean value = f_on.getValue ();
f_on.setValue (!value); // Light on / off
```


# The Browser Class (I)

```
public class Browser
{
   public String getName();
   public String getVersion();
   // For identification of the browser

   public float getCurrentSpeed();
   // Navigationspeed (m/s)

   public float getCurrentFrameRate();
   // Frame-Rate (images/s)
```


# The Browser Class (II)

```
public String getWorldURL();
// URL of the current world

public void replaceWorld(BaseNode[] nodes);
public void loadURL(String[] url, String[] parameter)
     throws InvalidVRMLSyntaxException;
// replace current world

public void setDescription(String description);
// set world description of browser (e.g. window title)
```


# The Browser Class (III)

```
public BaseNode[] createVrmlFromString(String
vrmlSyntax)
     throws InvalidVRMLSyntaxException;
// create new VRML nodes

public void createVrmlFromURL(String[] url, BaseNode
node, String event)
     throws InvalidVRMLSyntaxException;
// insert external file into scene
```


# The Browser Class (IV)

```
   public void addRoute(BaseNode fromNode, String
   fromEventOut, BaseNode toNode, String toEventIn);
   public void deleteRoute(BaseNode fromNode, String
   fromEventOut, BaseNode toNode, String toEventIn);
   // Create / delete routes dynamically
}
```


# Dynamic Routing

- using “addRoute” and “deleteRoute”-methods of the browser class
- References to the participating nodes and the field names are needed


# Dynamic Routing: Example

```
private SFNode field_fromNode;
public void initialize () {
    feld_fromNode = (SFNode) getField (“fromNode”);
}
public void processEvent (Event e) {
    if (e.getName ().equals (“trigger_event”)) {
        getBrowser().addRoute (field_fromNode.getValue (),
          “isActive”, this, “clicked”);
    }
    else if (e.getName ().equals (“clicked”)) {
        // do something
    }
}
```


# Creation of new VRML Nodes

- using “createVrmlFromString” method
- complex VRML structure may be generated using the Java classes especially the script classes


# Creation of new VRML Nodes: Example

```
SFNode f_aGroupNode = (SFNode) getField (“aGroupNode”);
Node aNode = (Node) f_aGroupNode.getValue ();
MFNode f_children = (MFNode) aNode.getExposedField
   (“children”);

Browser myBrowser = getBrowser ();
BaseNode[] nodes = null;
try {
    nodes = myBrowser.createVrmlFromString (“Box {}”);
} catch (InvalidVRMLSyntaxException x) {}

f_children.addValue (nodes[0]);
```


## What is missing?

- Possibility to address VRML browser externally
- Possibility to address VRML scene from an applet
- Possibility to integrate VRML scene in a hypermedia context

➡ EAI (External Authoring Interface)

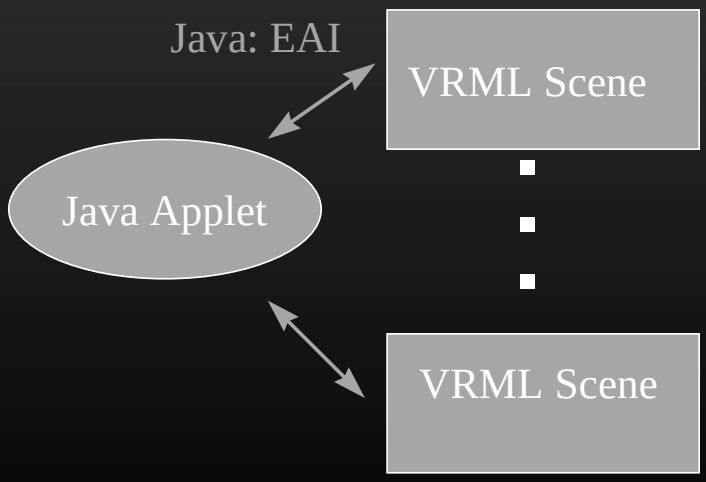

# EAI: History

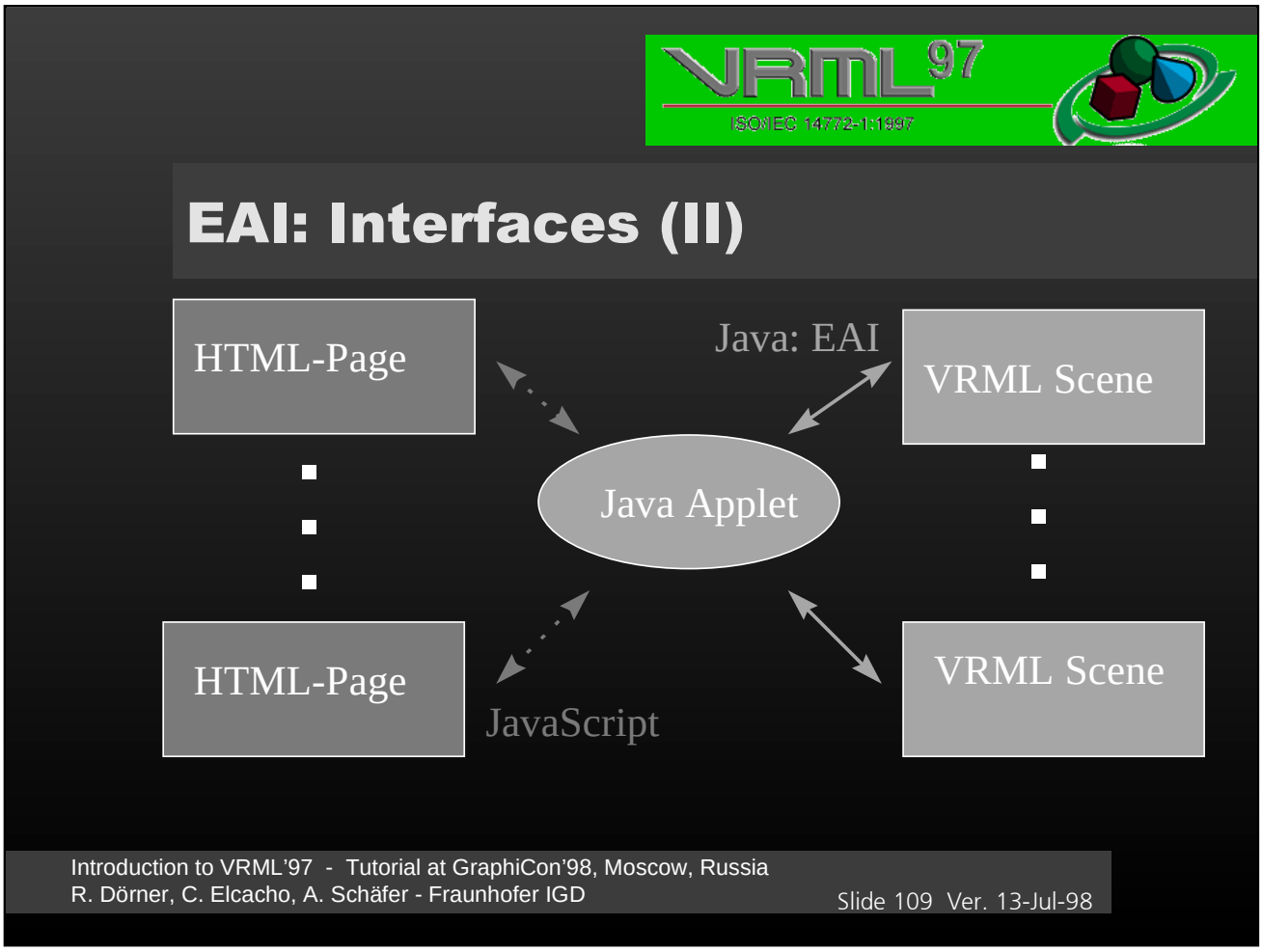
- Developed by Chris Marrin, Silicon Graphics, Inc.
- Proposal for extension of VRML97 standard
- First version: 01. April 1997
- Current version: 21. November 1997
- Revised version: 8. April 1998


# EAI: Official VRML Working Group

- New proposal announced for SIGGRAPH'98 (Mid July)
- Java implementation supports
  - Netscape, MSIE
  - Cosmo Player (SGI), Intervista WorldView, VrWave, Blaxxun Interactive CC3D

# EAI: Interfaces (I)




VRML 97
ISO/IEC 14772-1:1997
EAI: Interfaces (II)
HTML-Page
Java: EAI
VRML Scene
Java Applet
HTML-Page
JavaScript
VRML Scene
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 109 Ver. 13-Jul-98


# EAI: The HTML Start Page

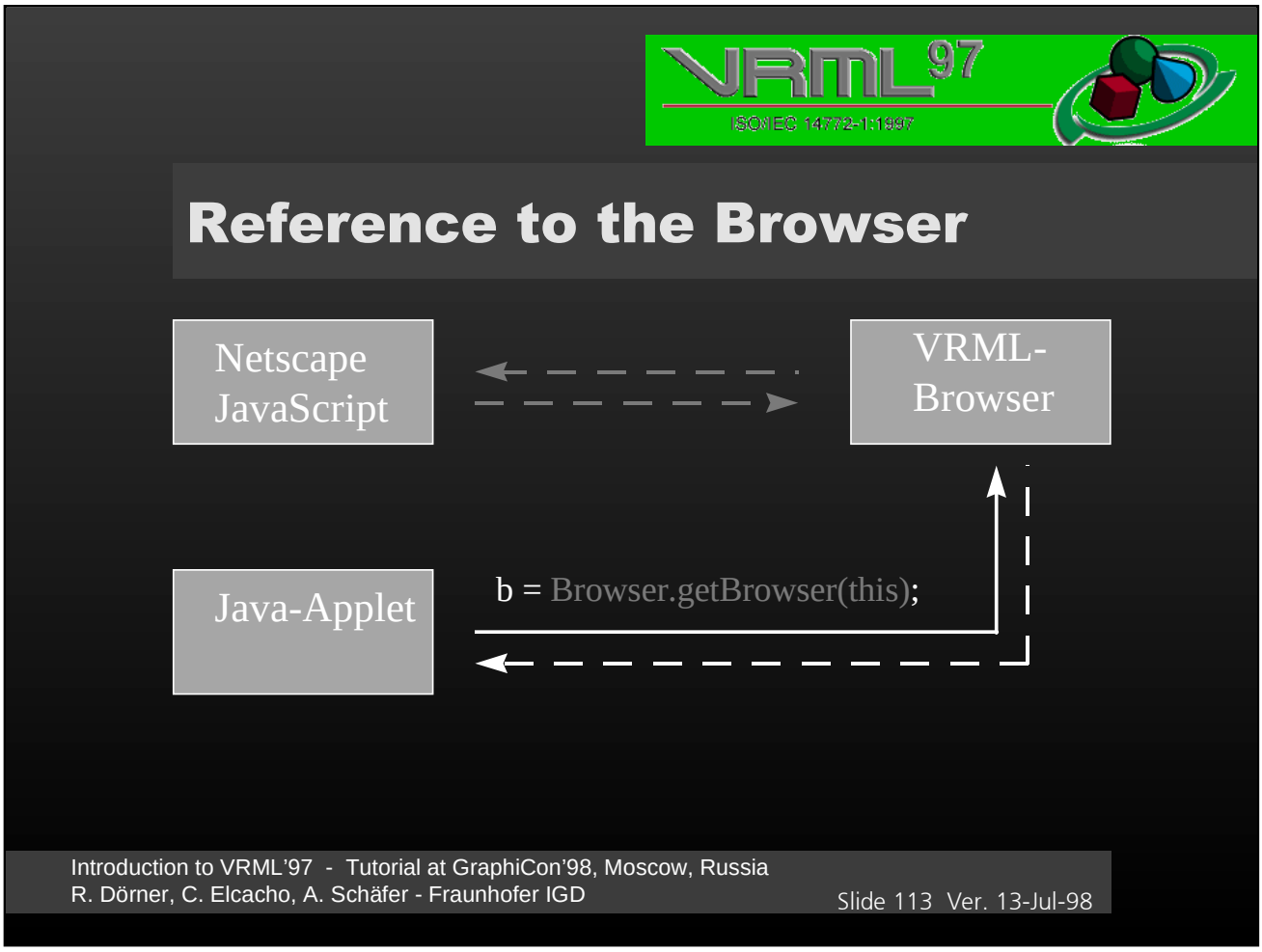
```
<HTML>
<TITLE>My First Page</TITLE>

<BODY>
<EMBED src="myScene.wrl">
<APPLET code="myApplet.class" mayscript>
    <PARAM name="..." value="...">
    ...
</APPLET>

</BODY> </HTML>
```


# EAI: The Interface Classes

- Using classes of the vrml.external.* package
  - vrml.external.Browser, vrml.external.Node
  - vrml.external.field (EventIn, EventOut, EventObserver)
  - vrml.external.exeption
- NOT identical to the Java scripting classes vrml, vrml.node.* and vrml.field.*


# The Java Applet

```
import java.applet.*;
import vrml.external.Browser;
...
public class myApplet extends Applet {
    Browser b;
    public myApplet {
        b= Browser.getBrowser(this);
        ...
    }
}
```


VRML 97
ISO/IEC 14772-1:1997
Reference to the Browser
Netscape JavaScript
VRML-Browser
Java-Applet
b = Browser.getBrowser(this);
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 113 Ver. 13-Jul-98


# The Browser Class (I)

Access to the Applet:

- static public Browser getBrowser (Applet applet);
- static public Browser getBrowser (Applet applet, String frameName, int index);


# The Browser Class (II)

Information about the current browser status:

- public String getName();
- public String getVersion();
- public float getCurrentSpeed();
- public float getCurrentFrameRate();
- public String getWorldUrl();


# The Browser Class (III)

Modifying the scene:

- replaceWorld (Node[ ] nodes);
- loadURL (String[ ] url, String[ ] parameter);
- createVrmlFromString (String vrmlsyntax);
- createVrmlFromURL (String[ ] url, Node node, String event);

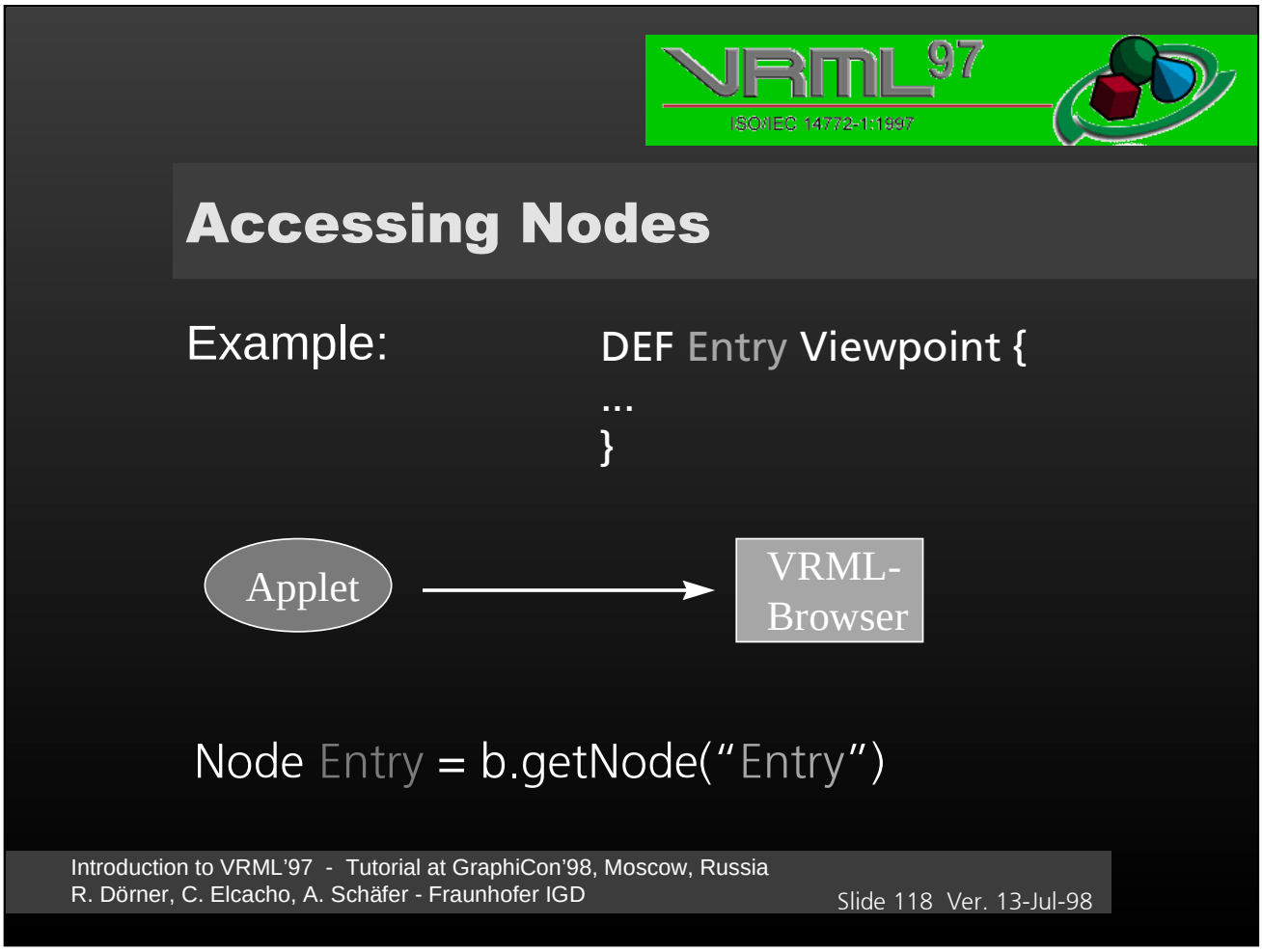

# The Browser Class (IV)

Modifiying the scene(II):

- addRoute();
- deleteRoute();

Obtaining references of a named node:

- public Node getNode(String name);


VRML 97
ISO/IEC 14772-1:1997
Accessing Nodes
Example:
DEF Entry Viewpoint {
...
}
Applet
VRML-
Browser
Node Entry = b.getNode("Entry")
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 118 Ver. 13-Jul-98


# The Node Class

- Accessing information about nodes
  - public String getType();
  - public EventIn getEventIn();
  - public EventOut getEventOut();


# Applet - VRML Communication

- Direct Manipulation of Nodes
  - Reading & writing of fields
- EventOutObserver for monitoring events in the VRML scene


# Reading Access: Example

VRML-Szene:

```
DEF move_me transform{ ... } ...
```

Java-Applet:

```
Browser b= Browser.getBrowser(this);
...
Node move_me = b.getNode("move_me");
EventOutSFVec3f field_translation =
    (EventOutSFVec3f)move_me.getEventOut("translation");
```


# Writing Access: Example

Java Applet:

```
Browser b= Browser.getBrowser(this);
...
Node move_me = b.getNode("move_me");
EventInSFVec3f field_translation =
    (EventInSFVec3f) move_me.getEventIn("set_translation");
float value[3] = new float[3];
value[0] = 14 ; value[1] = 7; value[2] = 89;

field_translation.setValue(value);
```


# Receiving Events (I)

- Routing mechanism cannot be used with Applets
- VRML scene and applet are independent

➡ Event Handling Mechanism:
EventOutObserver Class


# Receiving Events (II)

- Applet implemts the EventOutObserver interface
- Events are registered at the EventOutObserver using the advise method
- Reaction to the event is specified in the callback method


# EventOutObserver Interface (I)

```
public class MyObserver implements EventOutObserver{
    ...
    public void callback (EventOut value,
                          double timeStamp,
                          Object userData)
    {
        // Casting and Evaluation of the Out Event
    }
}
```


# EventOutObserver Interface (II)

```
class myApplicationClass{
    myApplicationClass() {
        Browser b = Browser.getBrowser(this);
        Node time = b.getNode ("time");
            // Creating EventOutObserverObject
        MyEventObserver ob =
                new MyEventObserver();
            // Registering callbacks
        time.getEventOut("fraction_changed").advise(ob,null);
    }
}
```


# EventOutObserver Interface (III)

```
public class MyApplet extends Applet implements
    EventOutObserver{
    ...
    public void init(){
        Browser b = Browser.getBrowser(this); ...
        Node time = b.getNode ("time");
                // Registring callback
     time.getEventOut("fraction_changed").advise(this,null);
    }
    public void callback (EventOut value, double timeStamp,
                Object data) { ...}
}
```


# JavaScript and VRMLScript

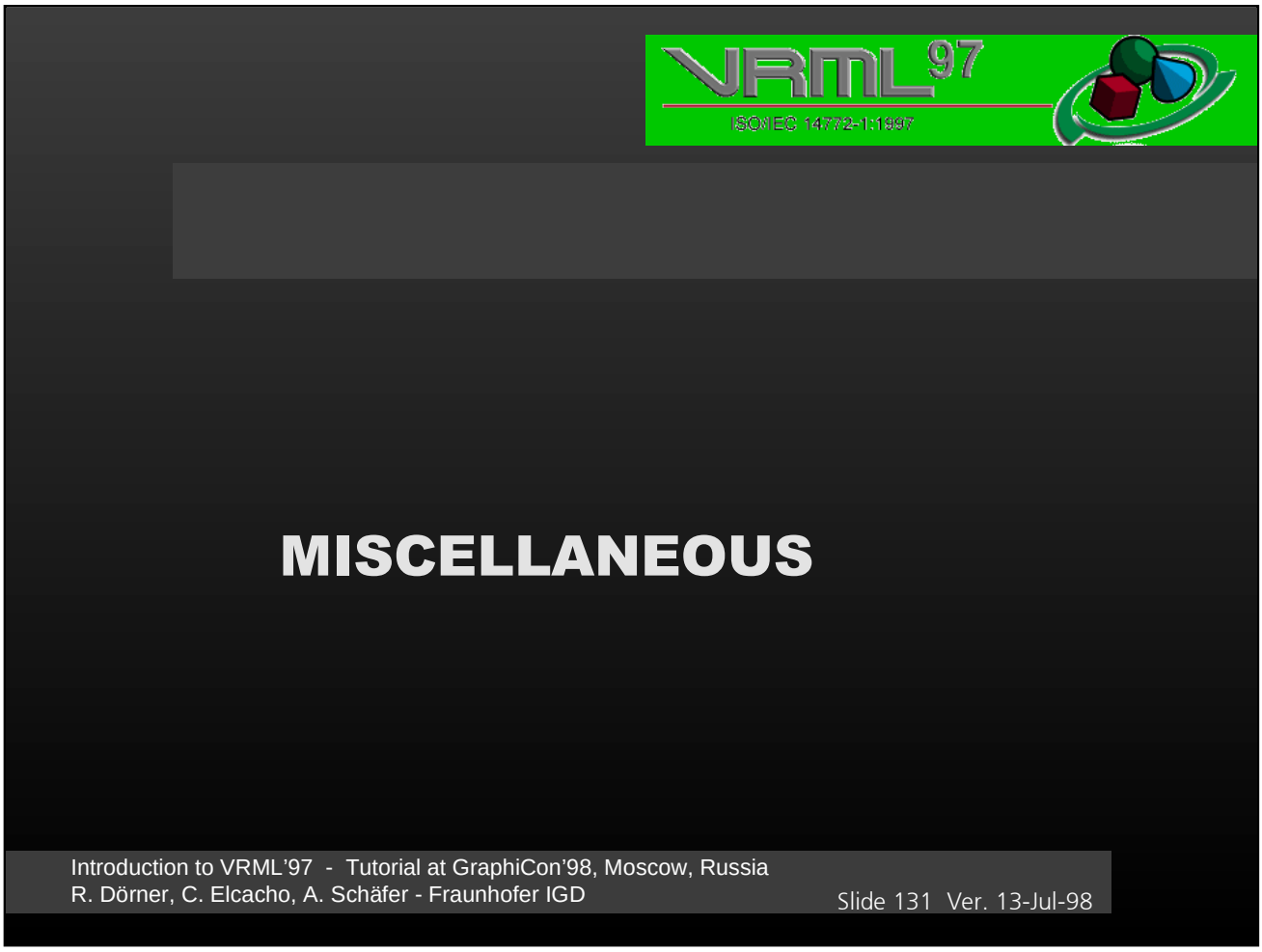
- Programs may be written in the VRML file
- JavaScript developed by Netscape
- JavaScript standardized by ECMA: ECMAScript
- VRMLScript developed by SGI
- VRMLScript is special subset of JavaScript


# JAVAScript and VRMLScript

```
DEF Blow Script {
   eventIn SFBool touch
   eventOut SFInt32 whichChoice
   url [ "javascript:
       function initialize () { whichChoice=1;}
       function touch (eventValue){
              if (whichChoice == 2)
                whichChoice = 0;
              else whichChoice ++;
       }"
   "choice.class"]
}
```


# VRML Programming: Conclusion

- Different languages:
  - Java
  - ECMAScript
- Different API
  - VRML Scripting using Script node
  - External authoring interface (EAI)


VRML 97
ISO/IEC 14772-1:1997
MISCELLANEOUS
Introduction to VRML'97 - Tutorial at GraphiCon'98, Moscow, Russia
R. Dörner, C. Elcacho, A. Schäfer - Fraunhofer IGD
Slide 131 Ver. 13-Jul-98


# VRML Authoring

- Integrative nature of VRML requires the use of many different tools in the authoring process:
  - Geometry modellers
  - Scene composition tools
  - Programming tools (e.g. Java compiler/IDE)
  - Browsers
  - Data Converters
  - Texture editors
  - Video/sound tools


# Performance / Scene complexity

- No. of Polygons: about 4000 triangles max. without 3D hardware (especially for distribution on the WWW)
- Too many textures may also be a bottleneck
- Limit number of Java threads


# Performance / Scene complexity

- Use interpolators und ROUTEs sparingly
- Use LOD
- Many Light sources have a great impact on performance
- Limit Size of .wrl file (when distributed on the Net)

⇒ use “gzip” compression


# Tools: VRML Browsers

- Typically free of charge
- Most often installed as “plugin” or “ActiveX Control” (Netscape/MS Internet Explorer)
- Installation of more than one Browser results typically in problems


# Tools: VRML Browsers

- COSMO Player 2.0
  - Windows
  - EAI, Java, JavaScript, VRMLScript
- Worldview 2.1
  - Windows
  - EAI, Java, JavaScript


# Tools: VRML Browsers

- Sony Community Place PRD2
  - Windows
  - Java
- CASUS Presenter
  - Windows, Sun, SGI, Linux
  - EAI, Java


# Tools: 3D World Builders

- ac3d
  free modeller, only for creation of geometry
- Caligari TrueSpace 3
  Animation system
- COSMO Worlds
  very powerful, SGI and PC


# Tools: 3D World Builders

- V-Realm Builder 2.1
  Animation system specially for VRML
- Internet 3D Space Builder aka
  Cosmo Home Space Designer
  creation of static worlds


## Other tools

- VRML Generators (e.g. automatic visualization of directory structure)
- LOD Generators
- Converters (e.g. for dxf, obj format)
- VRML Parsers
- VRML Syntax Checkers
- Test Suites


# Application examples

- Entertainment, e.g. virtual Lego-blocks
- Edutainment, e.g. space exploration
- Visualization of scientific data
- Biology, e.g. “visible human”
- Physics, e.g. visualization of electric fields


# Application examples

- Chemistry, e.g. molecule editing
- Navigation aids for WWW, e.g. 3D Hyperlinks
- Catalogue systems (virtual warehouse)
- Virtual exhibition booth
- Visualization of production processes in industry


# Application examples

- Arts, e.g. Choreography
- Virtual Museum
- Architecture, Urban development
- Marketing, e.g. “Banner ads”
- Database visualization


# Current developments

VRML Working Groups:

- MPEG-4 Integration
- DHTML Integration
- DBWork (Database connectivity)
- Compressed Binary Format
- Humanoid Animation
- Multi user support


# Current developments

- Browser Conformance
- DIS (Distributed Interactive Simulation)
- GeoVRML
- User Input (e.g. keyboard input)
- Objekt oriented VRML
- VRML Data Streaming

⇒ New version of standard expected in 1999


# Java 3D

- 3D API for Java
- Does not specify a file format
- Based on a scene graph model
- Specification 1997
- First implementation march 1998


# References: WWW (I)

- VRML Repository
  *http://www.sdsc.edu/vrml/*
  comprehensive list of links around VRML
- VRML Consortium home page
  *http://www.vrml.org/*
  The consortium has the goal of promoting and furthering the development of VRML


# References: WWW (II)

- COSMO Home page
  *http://cosmosoftware.com/*
  VRML Tools and links to interesting VRML worlds
- Javasoft Home page
  *http://www.javasoft.com/*
  Everything around Java


# References: WWW (III)

- The Mining Company
  *http://vrml.miningco.com/*
  Another repository of VRML related information


# References: mailing lists

- General VRML mailing list:
  *majordomo@vrml.org*
  *subscribe www-vrml*
- Several working group mailing lists


# References: books (I)

- VRML 2.0 Source Book
  Ames, Nadeau, Moreland
  Wiley, 1996
- Annotated VRML Reference Manual
  Bell, Carey
  Addison-Wesley, 1997
- Teach Yourself VRML in 21 Days
  Marrin, Campbell
  SAMS Net, 1997


# References: books (II)

- The VRML 2.0 Handbook
  Hartman, Wernecke
  Addison-Wesley, 1996
- Late Night VRML 2.0 with Java
  Roehl, Couch et al.
  ZD Press, 1997
- JAVA for 3D and VRML Worlds
  Lea, Matsuda, Miyashita
  New Riders, 1996


# Questions & Comments

? ? !