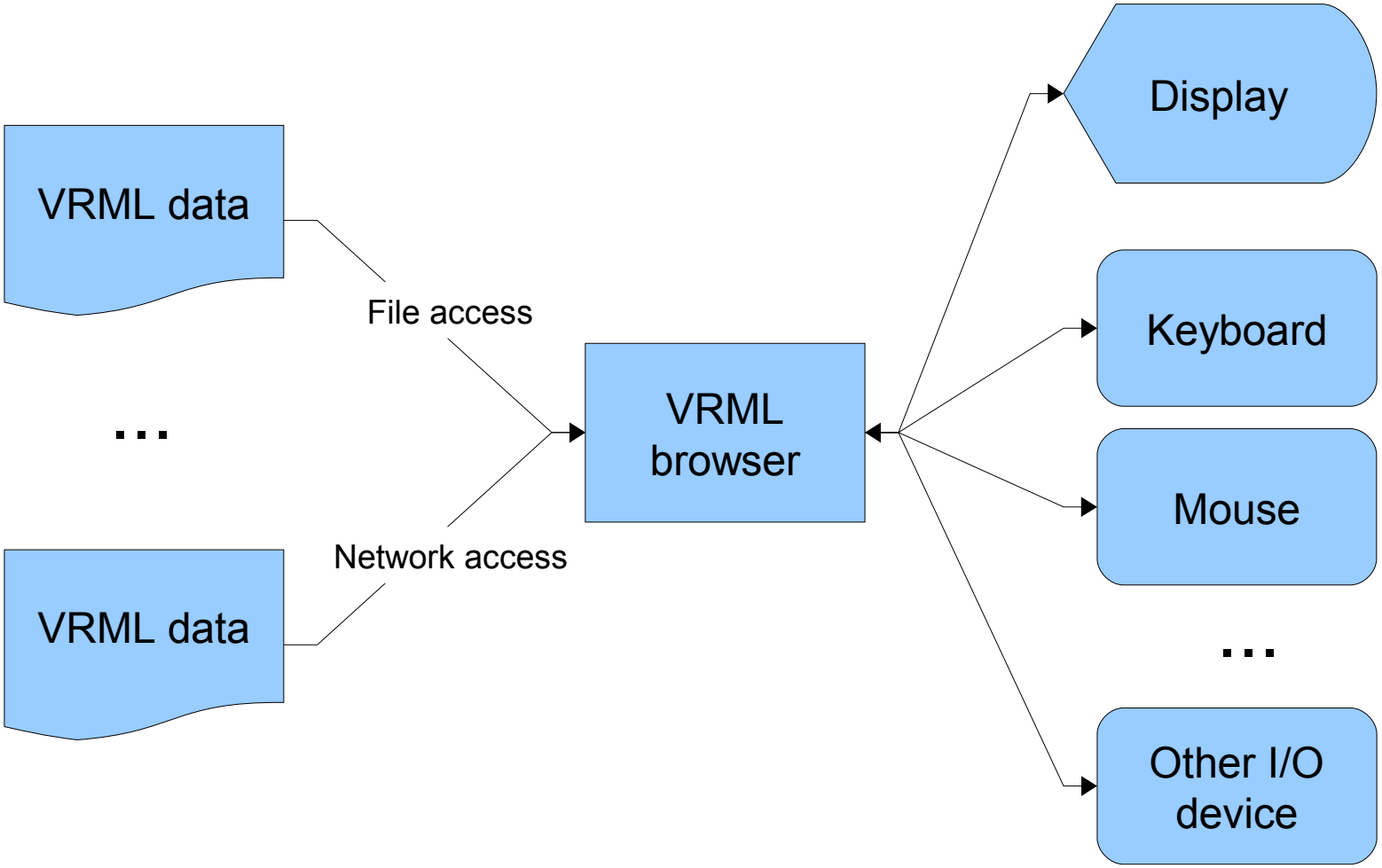
Introduction to VRML

Introduction

- *VRML*: *Virtual Reality Modeling Language*
- Previously known as *Virtual Reality Markup Language*
- What it is:
  - ✓ A language for the description of virtual worlds
  - ✓ Implementation-independent
  - ✓ An ISO standard
  - ✓ Superceded by X3D
- What is is not:
  - ✓ A language for the description of computer graphics
  - ✓ A programming language for computer graphics
- Created by Mark Pesce and Tony Parisi, introduced 1994
- Originally Web-oriented

The VRML flowchart



VRML browser

- A program that renders a virtual world according to VRML data in a file or network resource
- Available as a standalone applications or web-browser plugins (mostly due to VRML's original orientation towards the Web)
- Usually not strictly standards-compliant, to accomodate exports from 3D design software and non-critical source code errors

VRML development

- Traditional methods: coding using text editors
- VRML-specific design tools: accurate WYSIWYG design of VRML worlds, not really the hottest option around
- Generic design tools: must be able to export VRML files, design elements may be lost during export
- In-between solutions: text-based VRML editors, with VRML-specific features such as:
  - ✓ Code completion, highlighting and folding
  - ✓ Graphical tree and node view
  - ✓ Scene and node preview
  - ✓ Automatic code generation, cleanup and syntax checking
- Different methods for different circumstances

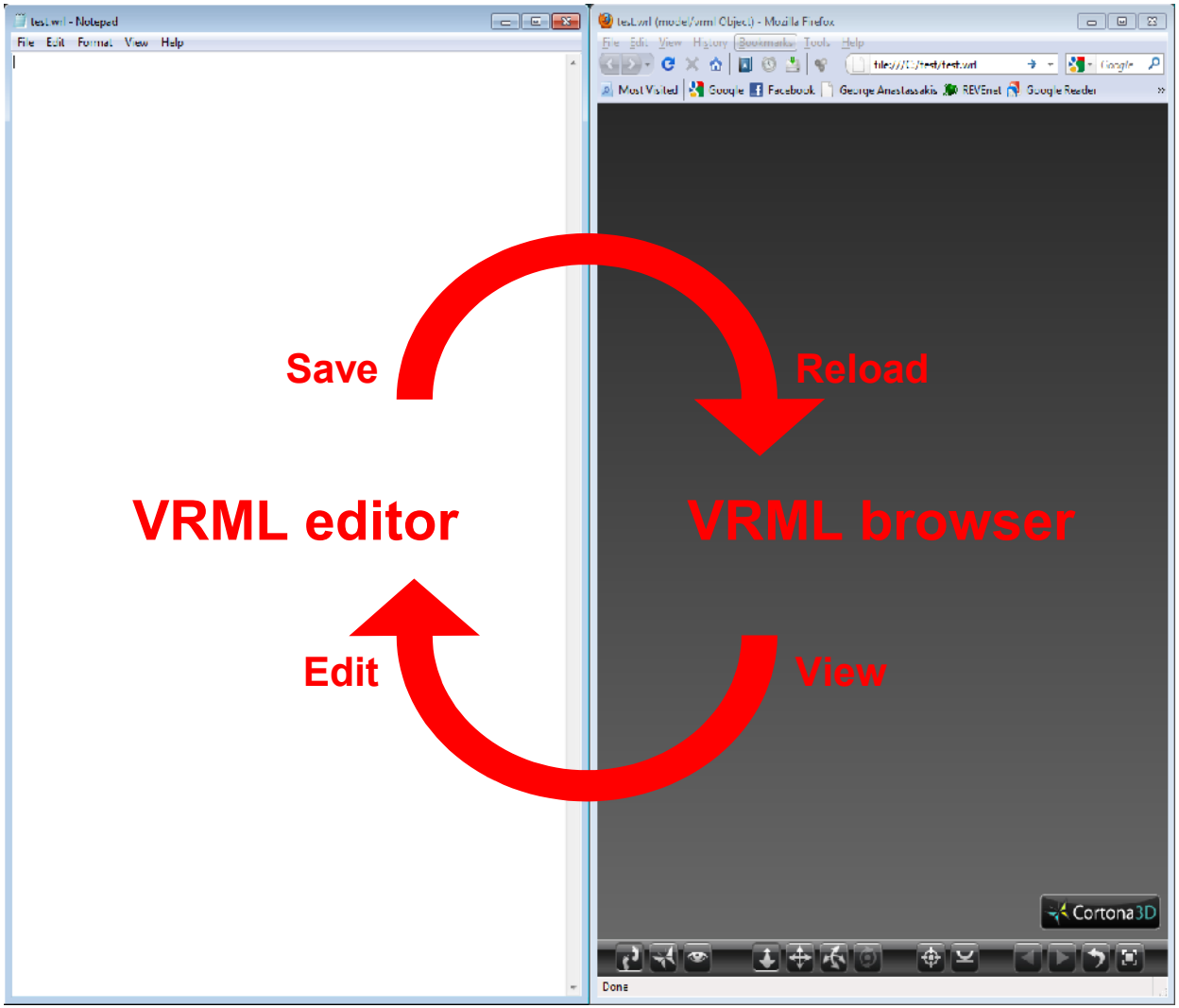
Suggested workflow (overview)

- Only applies to manual source coding, not to methods involving design tools
- Resources required:
  - ✓ Text editor or VRML editor
  - ✓ VRML browser (with reloading ability)
  - ✓ VRML specification (locally or online)
- Ideal for:
  - ✓ Learning and experimenting with VRML
  - ✓ Creating visually- and structurally-simple objects
- Not really ideal for:
  - ✓ Actual development of applications involving VRML
  - ✓ Design of visually- or structurally-sophisticated scenes

Suggested workflow step 1: environment setup

- Start the editor (yes, it will be *that* detailed)
- Start the VRML browser
- Arrange them vertically (if possible)
- Create an empty file with a ".wrl" extension (will be referred to as the "VRML file") on the editor
- Load the VRML file on the VRML browser (ignore complaints about missing header)
- You can now edit VRML source code, save it and reload the scene on the VRML browser without having to bring windows in and out of focus or rearrange them
- Following slide shows Notepad and Cortona3D Viewer in Mozilla Firefox on Microsoft Windows 2007

Suggested workflow step 1: environment setup



Suggested workflow step 2: VRML header

- In order for VRML source code to be valid, it must contain at least the VRML header:

```
#VRML V2.0 utf8
```

- Add the following at the top of the VRML file
- Save, reload

Suggested workflow step 3: add a node

- Add an empty node named <node_name>, for example, a Transform node:

```
DEF <node_name> Transform {
}
```

- Locate the node type (section “Node reference”):

...

6.52 Transform

```
Transform {
  eventIn      MFNode      addChildren
  eventIn      MFNode      removeChildren
  exposedField SFVec3f     center           0 0 0    # (-∞,∞)
  exposedField MFNode      children         []
  exposedField SFRotation  rotation         0 0 1 0  # [-1,1],(-∞,∞)
  exposedField SFVec3f     scale            1 1 1    # (0,∞)
  exposedField SFRotation  scaleOrientation 0 0 1 0  # [-1,1],(-∞,∞)
  exposedField SFVec3f     translation      0 0 0    # (-∞,∞)
  field        SFVec3f     bboxCenter       0 0 0    # (-∞,∞)
  field        SFVec3f     bboxSize         -1 -1 -1 # (0,∞) or -1,-1,-1
}
```

The Transform node is a grouping node that defines a coordinate system for its children that is relative to the coordinate systems of its ancestors. See 4.4.4, Transformation hierarchy, and 4.4.5, Standard units and

...

Suggested workflow step 4: learn about the node

- Locate information related to the node type (section "Concepts", use links in the node type specification):

...

The Transform node is a grouping node that defines a coordinate system for its children that is relative to the coordinate systems of its ancestors. See 4.4.4, Transformation hierarchy, and 4.4.5, Standard units and coordinate system, for a description of coordinate systems and transformations.

...

...

4.4.4 Transformation hierarchy

The transformation hierarchy includes all of the root nodes and root node descendants that are considered to have one or more particular locations in the virtual world. VRML includes the notion of *local coordinate systems*, defined in terms of transformations from ancestor coordinate systems (using Transform or Billboard nodes). The coordinate system in which the root nodes are displayed is called the *world coordinate system*.

A VRML browser's task is to present a VRML file to the user; it does this by presenting the transformation hierarchy to the user. The transformation hierarchy describes the directly perceptible parts of the virtual world.

The following node types are in the scene graph but not affected by the transformation hierarchy: ColorInterpolator, CoordinateInterpolator, NavigationInfo, NormalInterpolator, OrientationInterpolator, PositionInterpolator, Script, ScalarInterpolator, TimeSensor, and WorldInfo. Of these, only Script nodes may have descendants. A descendant of a Script node is not part of the transformation hierarchy unless it is also the descendant of another node that is part of the transformation hierarchy or is a root node.

...