

# InTml: A Description Language for VR Applications

Pablo Figueroa  
Dept. of Computing Science,  
University of Alberta,  
Edmonton, Canada  
pfiguero@cs.ualberta.ca

Mark Green  
School of Creative Media,  
City University of Hong Kong,  
Hong Kong, China  
smmark@cityu.edu.hk

H. James Hoover  
Dept. of Computing Science,  
University of Alberta,  
Edmonton, Canada  
hoover@cs.ualberta.ca

## ABSTRACT

We present the Interaction Technique Markup Language (InTml), a profile on top of the core X3D that describes 3D interaction techniques (InTs) and hardware platforms. InTml makes 3D InTs easier to understand, compare, and integrate in complete virtual reality (VR) applications. InTml can be used as a front end for any VR toolkit, so InTml documents that plug together 3D InTs, VR objects, and devices can be fully described and executed.

## Categories and Subject Descriptors

I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—*Virtual reality*; I.3.6 [Computer Graphics]: Methodology and Techniques—*Languages*; D.2 [Software]: Software Engineering

## Keywords

Virtual Reality, VR development environments, InTml, X3D profiles

## 1. INTRODUCTION

The core and full X3D profiles concentrate on the description of geometry and the compatibility with the VRML97 specification. Such nodes allow us to create applications with the same functionality we had in VRML97, but in a much better language. However, this language still lacks the necessary abstractions to describe complete VR applications that use novel I/O devices, or complex navigation, selection, or manipulation techniques. Traditionally, novel devices have been integrated to VRML applications as scripts that hide platform-dependent code, and novel interaction techniques by scripts that overrides the default navigation techniques. However, such solutions add a performance penalty due to the restricted execution model derived from VRML97, and also precludes the portability and understanding of the overall application.

InTml<sup>1</sup> is a XML application that describes input devices, output

<sup>1</sup>The language was previously known as 3dml. We decided to change its name to avoid confusions with [12] and to emphasize its main purpose as interaction technique description language

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Web3D'02, February 24-28, 2002, Tempe, Arizona USA.  
Copyright 2002 ACM 1-58113-468-1/02/0002 ...\$5.00.

devices, interaction techniques, and applications – a combination of all previous elements plus geometry in X3D. It is a language that can be readable by programs, as well as by designers. InTml provides an instrument for comparison of 3D InTs, for the study of alternative interaction techniques in an application, and for rapid prototyping. InTml can accelerate the production of VR applications, because its concepts are higher level than the ones in languages traditionally used in this area, such as C++ and Java, without the restrictions of devices and interaction techniques in languages such as VRML. InTml also provides a way to hide unnecessary details from designers such as device configuration and interaction technique implementation, so they can concentrate on the tasks that a VR application does<sup>2</sup>.

We plan to implement InTml as a front end for several VR toolkits. In this way, VR applications described as InTml documents could be portable among installations, and web-based distribution of full-featured VR applications might be feasible. Rapid prototyping environments will provide designers with tools for browsing and choosing 3D InTs and devices, so a VR application can be created in a separate environment from its actual deployment installation.

This paper is organized as follows: Related work is presented in Section 2, followed by the basic concepts of this representation in Section 3, and the abstract execution environment in Section 4. Section 5 shows some examples of use, Section 6 shows the current state of the development, Section 7 shows the way InTml can be integrated to VR toolkits, in particular to environments based on the core and full profiles of X3D, future work is in Section 8, and some conclusions in Section 9.

## 2. RELATED WORK

Current VR toolkits define InTs in programming languages such as C++ or Java, so programming skills in such languages are necessary to develop any application. VRML[6] and the full X3D[22] can define 3D InTs as a set of SCRIPT nodes connected by ROUTEs; however it is not easy to identify particular InTs, to change the default set of devices, and to reuse such InTs in other programs.

The X3D components specification[23] defines a way to extend the functionality of X3D by using BML[4], a markup language for Java beans integration that is very powerful but exposes much of the bean complexity to the user. InTml proposes a simpler component model to the one used in BML.

Newsgroups such as 3dml[1] have been discussing a way to describe 3D InTs, either to capture their design and intention in for-

<sup>2</sup>These details are implemented by a particular VR toolkit, hidden from the designer.

malisms such as UAN[19], or to capture their main algorithm and code generation mechanisms by pseudocode or scripting languages such as Python[21]. This paper presents a markup language that can be integrated to the set of profiles on top of X3D in order to describe all design issues of a VR application, in a VR toolkit-independent way.

This paper describes details about the integration between the X3D profiles and InTml. More details about InTml as a language and its features, like complexity hiding or extensibility, can be found in[11].

### 3. BASIC CONCEPTS

InTml uses a dataflow architecture as the general structure of an application, because of its generality and because it can be implemented in a broad variety of environments. VR objects, devices, and InTs are the components of this dataflow, and there is a simple component model that defines how such elements are interconnected. The component model describes 3D InTs and devices by their interface –input and output chunks of information– plus ways to encapsulate details of complex elements.

A *filter* represents interaction techniques in the dataflow. In its simplest form, it is composed of input and output ports – which carry typed events – and state information. Figure 1 shows a way to represent a filter – `SelectByTouching` – in a diagram that shows its input ports on the left of a box and output ports on the right. Its output is a selected object from the scene. Its inputs are the VR object used as hand representation, the current position and orientation of such an object, the scene of objects to pick from, and the events that inform about added or deleted objects from the scene. More complex filters can encapsulate interconnected instances of filters and objects. In its purpose, it is similar to the node with the same name in the SVG[8] specification, with the difference that SVG filters are predefined tags and have in general only one input and only one output; whereas InTml filters can be user defined, can receive input from several sources and can send several chunks of information to other filters.

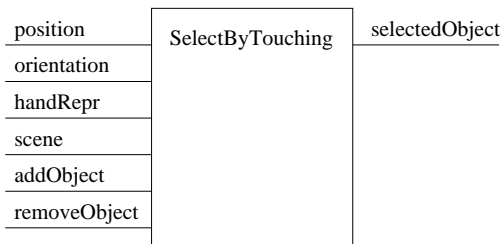


Figure 1: Select by Touching. An Example of a Filter

VR objects represent identifiable pieces of content in the virtual environment: elements that can be seen, heard, or touched by the user. An *object holder* is a special type of filter that allows changes on a VR object. An *input device* represents data originated from an input device, whereas an *output device* represents the capabilities for rendering of an output device. A *platform* is a collection of input and output devices, that represents a particular physical setup.

An *application* allows a designer to plug together all the previous elements in order to meet certain user requirements. Figure 2 shows a simple application, that allows a user to move a virtual hand with a tracker and touch virtual objects. In this example we have one filter (`SelectByTouching`), one InT (`Feedback`), one input device (`handTracker`), one output device (`console`), two VR objects (`scene` and `handRepr`), and one object holder (`handRepresentation`).

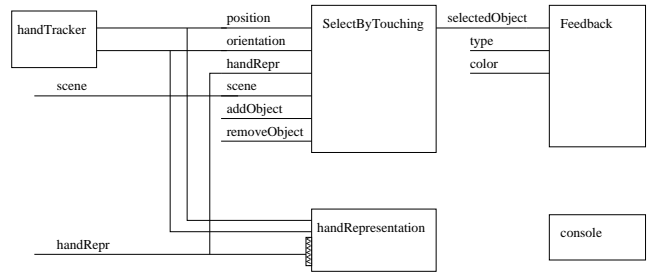


Figure 2: Simple Application. Touching Objects With a Virtual Hand.

Filters, InTs, and applications are documented in InTml by a short description, a long description, multiple index-based views<sup>3</sup>, and references to the source of information where they are defined. This information is used in the presentation scheme presented in Section 6.

### 4. INTML EXECUTION ENVIRONMENT

Given the variety of hardware installations that InTml is targeted at, its execution model for a VR application should accommodate the features and limitations of each platform. InTml defines general semantic rules that can be implemented in different hardware architectures, no matter the number of machines, processors, or I/O devices. The following characteristics define a InTml program in execution:

- A filter can receive several events from all its inputs at the same time. All the events it generates to its output ports in one execution are considered as originated from the same moment in time.
- Objects in the scene can only change as a result of events received by object holders. No secondary effects are caused from the implementation of 3D InTs

The execution of the dataflow of filters can be divided in three main stages:

- collecting time (*ct*): Input from devices is collected and pre-processed.
- execution time (*et*): Input from devices is propagated to the filters in the dataflow. The actual order of execution of filters is defined by the particular implementation of the execution model. Each filter reads information from its input ports, executes its behavior, and sends at once all the produced information by its output ports.
- propagation time (*pt*): All change events received at object holders are executed in the corresponding objects in the scene.

The separation between *et* and *pt* ensures that all filters see the same state of the scene while they are executing, at any given time. This avoids complex interdependencies in the execution of interaction techniques.

### 5. MAIN FEATURES AND EXAMPLES

Listings 1, and 2 are InTml documents that represent the examples in Figures 1, and 2, respectively <sup>4</sup>.

<sup>3</sup>Index-based views are explained in Section 6

<sup>4</sup>Only important elements are presented here. For more details look at the InTml website.

---

**Listing 1** Select by Touching. XML Code

---

```
1 <FilterClass id="SelectByTouching">
  <Description>
    Implements details of selection by
    collision detection.
5 </Description>
  <Indexes>
    <Index id="first"
      value="InTml.selection.details"/>
    <Index id="papers"
10      value="_hidden"/>
  </Indexes>
  <IPort id="p" type="Pos3D">
    <ShortDesc>Change of position
    </ShortDesc> </IPort>
15 <IPort id="q" type="Quaternion">
    <ShortDesc>Change of rotation
    </ShortDesc> </IPort>
  <IPort id="handRepr" type="VRObject">
    <ShortDesc>Users' hand
20 </ShortDesc> </IPort>
  <IPort id="scene" type="Scene">
    <ShortDesc>Selectable objects
    </ShortDesc> </IPort>
  <IPort id="addObject" type="VRObject">
25 <ShortDesc>Dynamically added objects
    </ShortDesc> </IPort>
  <IPort id="removeObject" type="VRObject">
    <ShortDesc>Dynamically removed objects
    </ShortDesc> </IPort>
30 <OPort id="object" type="VRObject">
    <ShortDesc>Selected object</ShortDesc>
  </OPort>
</FilterClass>
```

---

---

**Listing 2** Simple Application. XML Code

---

```
1 <App>
  <!-- Platform-dependent section -->
  <ODevice id="display"
    type="GenericDisplay"></ODevice>
5 <IDevice id="handTracker"
  type="Generic6DOFTracker"></IDevice>
  <!-- Load the scene -->
  <Object id="scene" type="Scene"
    filename="scene.wrl"></Object>
10 <Object id="handRepr" type="VRObject"
    filename="hand.wrl"></Object>
  <!-- InT instances -->
  <ObjectHolder id="handRepresentation"/>
  <Filter id="select"
15   type="SelectByTouching"></Filter>
  <Filter id="feedback"
    type="FeedbackOneIT"></Filter>

  <Binding iE="_self" iP="handRepr"
20   oE="select" oP="handRepr" />
  <Binding iE="_self" iP="handRepr"
    oE="handRepresentation" oP="object"/>
  <Binding iE="handTracker" iP="pos"
    oE="select" oP="pos" />
25 <Binding iE="handTracker" iP="q"
    oE="select" oP="q" />
  <Binding iE="_self" iP="scene"
    oE="select" oP="scene" />
  <Binding iE="select" iP="object"
30   oE="feedback" oP="object" />
</App>
```

---

The element `FilterClass` defines the interface of new interaction techniques, in terms of input and output ports. Each input and output port has a name and a type, and an input port can have a default value. For example, `<OPort id="object" type="VRObject">` defines an output port named `object` of type `VRObject`. Instances of such classes can be used in any other filter or application declaration.

The element `App` is used to create applications. Listing 2 shows how instances of objects, devices, filters, and InTs can be defined and connected together in order to show selection by touching. Note that geometry might be separated from the InTml description, so details about geometry are hidden.

## 6. CURRENT STATUS

The development around InTml has been concentrated in the following areas: The design of the DTD, editor support, a documentation scheme, a library of interaction techniques, and a set of applications as examples.

The DTD contains the 23 tags that describe the language: `App`, `Binding`, `Constant`, `Description`, `Deviceclass`, `Filter`, `FilterClass`, `IDevice`, `Implements`, `Import`, `Index`, `Indexes`, `IPort`, `Object`, `ObjectClass`, `ObjectHolder`, `ODevice`, `OPort`, `Package`, `PaperRef`, `Platform`, `PlatformClass`, and `Shortdesc`. A simple HTML documentation is presented in the website by using perlSGML[14]. We are considering a further refinement of integration with X3D, in terms of style and in terms of node names.

Xeena[9], an extendible and generic editor for XML documents, has been adapted as an editor for InTml documents. Despite the basic support – it just guides the user in the creation of valid InTml documents – we think is the first step in a more robust and guided way to create InTml applications.

We have developed in our website [10] a presentation scheme

for InTml documents, that is shown in Figure 3. These webpages present InTml documents in a more readable way than in plain XML, for comparison and understanding purposes. The main elements in this scheme are:

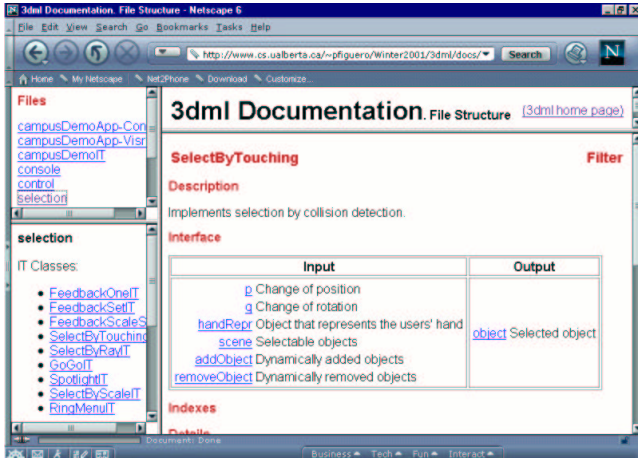


Figure 3: File view of InTml.

- File view (Figure 3). Applications and InTs are classified by the file they are contained in. It is the basic view for the library contents and the simpler way to browse the information.
- Category views (Figure 4). It is possible to create several hierarchical index-based views for InTs and applications. In this way InTs can be classified by several criteria, which can help designers to understand easier the library and choose the best InTs for a particular application. For example, all InTs we have described in InTml are indexed by the paper in which they appear, plus the following categories: travel, selection, manipulation, control, and feedback<sup>5</sup>.
- Filter details. It presents a general description of the filter, its interface – input and output ports – its position in other categories, its implementation – object holders, filter, and InT instances – details about its ports, and bibliography.
- Application details. It presents a summary of the tasks – InTs and filters – objects, and devices that the application uses, with a detailed information of connections and purposes of each element.

<sup>5</sup>Inspired by Bowman's [5] and Barrileaux's [2] work

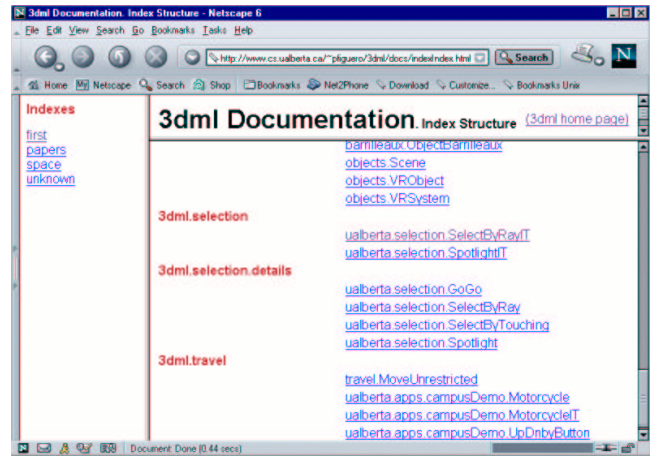


Figure 4: View by categories of InTml.

These HTML pages have been generated from the InTml documents using XSLT[24], and can be used to show any document that follows the InTml definition rules.

The specification currently contains the most important 3D InTs presented in papers in the last 10 years: selection by raycasting, selection by touching, aperture-based selection[13], techniques based on proprioception[16], image-plane based techniques[17], and walking techniques[20].

## 7. IMPLEMENTATION OF INTML

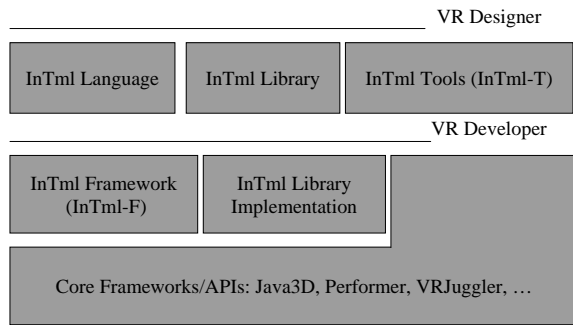
InTml is a profile for description of VR applications and 3D interaction techniques that can be implemented in several toolkits and environments, such as the full X3D profile.

Figure 5 shows the proposed architecture for an implementation of InTml. It is designed to support two different roles in the development of any VR application: VR designers, who define new applications by plugging-in components from the library, and VR developers, who write the required code<sup>6</sup> for new components. VR designers create new applications with the provided tools by composing components from the InTml library. Such tools are: the library browser, the semantic checker for new component definitions, the compiler that translates applications written in InTml to the framework language, and a visual programming environment<sup>7</sup>. VR developers use, extend, and maintain a white box framework<sup>8</sup>, that implements the InTml concepts on top of a core VR framework, and the implemented components of the InTml library.

<sup>6</sup>The required code might be geometry and appearance details for objects, code in the core framework for new interaction techniques and animations, or code for the integration of new devices.

<sup>7</sup>This programming environment is not yet available

<sup>8</sup>A white box framework is the first stage in the evolution of a framework as is defined in [18], and we plan to continue this evolution pattern.



**Figure 5: Architecture of an implementation of InTml.**

A new implementation of InTml, in terms of Figure 5, consists of creating the appropriate compiler, InTml Framework, and InTml Library Implementation for a particular Core Framework. In particular, the implementation of InTml on top of the full X3D profile is pretty straightforward. The InTml Framework can be defined as a set of rules of how prototypes for interaction techniques should be defined, the InTml Library Implementation can be a set of prototypes that implement the interfaces defined in the InTml Library, and the compiler can be embedded in a browser that reads both InTml and X3D files, or a stand-alone module that outputs X3D files from InTml descriptions.

In the case of a InTml-capable browser, geometry of VRObjects can be described in the core X3D, in a different document as in Listing 2 or embedded in the same document by the use of XML namespaces. The browser will load the InTml description of the application – interaction techniques, object identifiers, devices, and connections – and the necessary X3D tags for each object’s geometry.

In the case of a stand-alone compiler, the environment should have PROTOs for all interaction techniques and devices. Each prototype might include scripts or calls to external Java classes, in order to implement the particular functionality of the InT.

However, any implementation on top of X3D has the following limitations:

- Due the execution model of X3D, only one event can be received in a filter, at any time. This limits the general execution model of InTml.
- Most of the navigational techniques should be implemented in scripts, that override the default behavior, so there is a penalty in performance.
- The behavior of object holders is not guaranteed.
- PC-based applications can be easily represented. However, other platforms require an implementation of several scripts that read such devices, or a special implementation of the X3D browser.

Listing 3 shows one possible translation of the program in Listing 2 to the full X3D profile. Scene and object elements are included in the description as inline elements, filter, device, and object holder instances are created as ProtoInstances, given the external, previously-defined, and platform-dependent proto definitions

**Listing 3 Simple Application. X3D code**

```

1  <X3D>
    <Scene>
        <NavigationInfo type="NONE"/>
        <ProtoInstance DEF="handTracker"
5     name="handTracker"/>
        <Inline DEF="scene" url="scene.wrl"/>
        <Inline DEF="handRepr"
            url="hand.wrl"/>
        <ProtoInstance DEF="handRepresentation"
10     name="ObjectHolder">
            <fieldValue name="object"
                value="handRepr"/> </ProtoInstance>
        <ProtoInstance DEF="select"
15     name="SelectByTouching">
            <fieldValue name="handRepr"
                value="handRepr"/>
            <fieldValue name="scene"
                value="scene"/> </ProtoInstance>
        <ProtoInstance DEF="feedback"
20     name="FeedbackOneIT"/>
        <ROUTE fromField="pos"
            fromNode="handTracker"
            toField="pos"
            toNode="select"/>
25     <ROUTE fromField="object"
            fromNode="select"
            toField="object"
            toNode="feedback"/>
        </Scene>
30 </X3D>
  
```

in the library. Finally, connections between filters are modeled either by fieldValues inside the proto instances or by routes. Note that the default navigation technique is disabled and the output device is now changed to the possibilities of X3D.

## 8. FUTURE WORK

This is the first stage of our project, in which we define the language and the documentation mechanisms of 3D InTs and VR applications. It is possible to design tools to view 3D InTs in more readable formats –i.e. automatically generated diagrams from the description, or visual programming environments for designers. We are working in the first implementations of InTml as front end of VR toolkits, and as a proof-of-concept we are using VR-Juggler[3], suitable for our CAVE-like installation, and Java3D[15] with additional frameworks such as Xj3D[7], suitable for any PC.

We have started a process of debugging the InTml documents that describe our library of 3D InTs. In the future we plan to incorporate more InTs, to offer the designer better tools to handle this representation, and to add more documentation and usability data.

## 9. CONCLUSIONS

InTml defines a profile on top of the core X3D for description of full-featured virtual reality applications. InTml defines a uniform way to represent 3D InTs that is high-level, toolkit-independent, component-based, reusable, and extensible.

Designers of VR applications can understand and compare several 3D InTs described in the same language, and with an uniform documentation paradigm. This language also allows designers to represent VR applications at a high level of abstraction, allowing them to easily use 3D InTs.

## 10. REFERENCES

- [1] 3d user interfaces newsgroup. <http://www.mic.atr.co.jp/~poup/3dui.html>.
- [2] J. Barrileaux. *3D User Interfaces With Java 3D*. Manning Publications, August 2000.
- [3] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. Vr juggler: A virtual platform for virtual reality application development. In *Proceedings of IEEE Virtual Reality*, pages 89–96, 2001.
- [4] BML: Bean Markup Language. <http://www.alphaworks.ibm.com/tech/bml>.
- [5] D. A. Bowman and L. F. Hodges. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *The Journal of Visual Languages and Computing*, 10(1):37–53, February 1999.
- [6] R. Carey and G. Bell. *The Annotated VrmL 2.0 Reference Manual*. Addison-Wesley, 1997.
- [7] W. Consortium. Java3d x3d/vrml loader. <http://www.web3d.org/>.
- [8] W. W. W. Consortium. Scalable vector graphics specification. <http://www.w3.org/TR/SVG/>.
- [9] I. Corporation. Xeena. xml editing environment. <http://www.alphaworks.ibm.com/tech/xeena>.
- [10] P. Figueroa. InTml. vr applications for non-programmers. <http://www.cs.ualberta.ca/pfigueroa/InTml/>.
- [11] P. Figueroa, M. Green, and H. J. Hoover. 3dml: A language for 3d interaction techniques specification. In *Eurographics 2001. Short Presentations*, pages 273–280, 2001.
- [12] Flatland. 3dml. <http://www.3dml.org/>.
- [13] A. Forsberg, K. Herndon, and R. Zeleznik. Aperture based selection for immersive virtual environments. In ACM, editor, *UIST*, pages 95–96, 1996.
- [14] E. Hood. perlsgml. <http://www.nacs.uci.edu/indiv/ehood/perlSGML.html>.
- [15] S. Microsystems. Java 3d home page. <http://java.sun.com/products/java-media/3D/index.html>, 1997.
- [16] M. R. Mine, F. P. B. Jr., and C. H. Sequin. Moving objects in space : Exploiting proprioception in virtual-environment interaction. In ACM, editor, *SIGGRAPH*, pages 19–26, 1997.
- [17] J. S. Pierce, A. Forsberg, M. J. Conway, S. Hong, R. Zelenik, and M. R. Mine. Image plane interaction techniques in 3d immersive environments. In ACM, editor, *Symposium on Interactive 3D Graphics*, pages 39–43, 1997.
- [18] D. Roberts and R. Johnson. Patterns for evolving frameworks. In D. R. Robert Martin and F. Buschmann, editors, *Pattern Languages of Program Design 3*. Addison-Wesley, 1998.
- [19] A. C. Siochi and H. R. Hartson. Task-oriented representation of asynchronous user interfaces. In ACM, editor, *Proceedings of the SIGCHI conference on Wings for the mind*, pages 183–188, 1989.
- [20] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and J. Frederick P. Brooks. Walking > walking-in-place > flying, in virtual environments. In *Proceedings of the SIGGRAPH 1999 annual conference on Computer graphics*, pages 359–364. ACM, 1999.
- [21] G. van Rossum. Python language website. <http://www.python.org/>.
- [22] Web 3d consortium. <http://www.web3d.org>.
- [23] X3d components specification. <http://www.web3d.org/TaskGroups/x3d/lucidActual/X3DComponents/X3DComponents.html>.
- [24] XSLT: XSL Transformations. <http://www.w3.org/TR/xslt11/>.