Non-commercial Object-Base Scene Description

S.R. Gulliver, G. Ghinea, and K. Kaur

School of Information System, Computing and Mathematics
Brunel University, United Kingdom
{Stephen.Gulliver,George.Ghinea,Kulveer.Kaur}@brunel.ac.uk

Abstract. This paper investigates methods of describing two-dimensional and three-dimensional scenes using eXtensible Mark-up Language (XML). It also investigates the initial development of a non-commercial object-based scene description language, for the effective capture of two- and three-dimensional scenes. Design science research techniques were applied to determine the ideal factors involved in effective scene capture. The solution facilitates the description of 2D and 3D environments via interpretation of object relationships; the implementation of the inheritance, functionality, interactions and behaviour.

Keywords: XML, Scene description, hierarchy, functionality, behaviour.

1 Introduction

The process of creating virtual environments is diverse, but is traditionally achieved by either describing object properties (coding / scripting), or by positioning predefined objects into a blank virtual space ('drag-and-drop', a metaphor that allows the user to pick up an object via the mouse pointer and move it to the desired location). There have been a number of languages developed to create 2D and 3D scenes and virtual environments. Sadly the languages possess problems of interoperability, reliability and performance; the inability to incorporate effectively dynamic or interactive scenes (Walczak, 2003); as well as limited or conflicting commercial support of rendering within web browsers (Macdonald, *et al., 2005*). Modifications to previous standards have been made, yet many of these problems still exist.

1.1 Scene Description Languages

Scene Description Languages aim to capture the physical world in a readable and convenient way. The advent of the XML based technologies has delivered a number of scene description languages with a range of advantages and disadvantages.

In 1994, first International Conference of the World Wide Web (WWW) was held in Geneva (Carson, *et, al*., 1999). At this conference Mark Pesce, Tony Parisi, Tim Berners-Lee, and David Ragget presented the concept of VRML. In the same year a series of requirements was developed by the WWW and an open request for the proposal was issued. As a result Silicon Graphics introduced the Open Inventor format, which has been widely accepted. Silicon graphics engineer Gavin Bell developed the working document for a VRML 1.0 specification and the first draft was presented in October 1994 at the Second International World Wide Web Conference. The early VRML just was static world, allowing for no interaction between the objects. The extended version VRML 2.0 enhanced this with the support of JAVA, JavaScript, sound, animation. The main technique used by VRML files for describing 3D world is that of the hierarchical scene graph. Scene graph techniques were made popular by the OpenInventor programmer toolkit from silicon graphics (Nadeau, 1999). As its name suggests, a scene graph is a hierarchy of groups and shapes arranged in a tree structure. Each of the parents and children within a scene graph are called nodes. These nodes are used to construct 3D shapes, position the user's viewpoint, aim a light source, define animation paths, group shapes together and so forth. Scene graph parents manage groups of children that my have children of their own. To create a complex scene hierarchy, the children can be members of other scene graphs. The child node inherits the properties of the parent node, known as simple inheritance.

The Extensible 3D language (X3D), was proposed by the Web3D consortium in 2001. X3D defines a three-dimensional run time environment and a mechanism to deliver 3D content on the web; thus extending VRML with some new graphics features (Bullard, 2003). New features included: Non Uniform Rational B-splines (NURBs), Humanoid Animation, Multi-texturing, triangle primitives, 2D shapes inside 3D shapes, advanced application programming interfaces, additional data encoding formats, and a modular approach to implementation. Although adding graphical realism, such improvements do not consider either the issues of dynamic modelling and scene behaviour.

Walczak and Wojciech (2003) developed a high level language called X-VRML, an XML based language that adds dynamic modelling capabilities to virtual scene description standards such as VRML, X3D, MPEG-4. This language overcomes the limitations of passive virtual reality systems by providing access to databases, parameterisation, object-orientation and imperative programming techniques. Parameterised virtual scene models are used to build database driven Virtual Reality (VR) applications. The applications are able to generate virtual scene instances according to the models, user queries, data retrieved from database, user preferences and the current state of the application. X-VRML therefore allows a user to make changes in the models structure at runtime, which facilitates scene behaviour and interaction. The X-VRML language also permits designers to extend the language with new features without affecting the previous designed application.

Hulquist *et al.* (2006) proposed an interesting and effective technique for generating virtual environments *(VEs)* and describing scenes. Similar in nature to many computer game world simulators, this technique describes scenes by allowing the user to control the scene environment, for example: wet, sparse, tropical, cloudy, light, mountains and undulating. The main benefit of this technique, although not allowing description of object parameter, is that it allows large and complex virtual environments matching certain user requirements to be created relatively quickly.

1.2 Adding Behaviour to the Scene

In our opinion, the success of 3D applications on the web depends upon the degree of object behaviour and interactivity. Object behaviour could be the function, weight, gravity, kinematics, etc. For a number of decades, the use of visual representation in the process of building software has been studied and a number of high-level languages have been developed. Virtual environments are used for a number of applications including educational, industrial process, behavioural modifications and games. At present, many tools have been developed in order to generate static environments but a lot more could be done with respect to dynamic worlds.

Commonly behaviour is applied through short programs generally written in a script language and then connected to the 3D objects in the visual file. For example, Meyer and Conner (1995) have suggested language extensions that allow specifying new types of VRML nodes as well as behaviour descriptions, which can be reused and composite. They proposed a separator called prototyping nodes, which could allow defining something without actually using it. Another *behaviour system* has been described by Nadeau and Moreland (1995) that uses Perl scripts integrated with VRML worlds. In this system behaviour is defined using another set of objects. Example of this could be teapot, which includes a number of objects such as heating its content like tea, etc.

An intelligent behaviour system was also developed by Del Bimbo, *et al.* (1994), whereby an agent responds to events in a virtual environment using intelligent decision-making. Similarly, Arjomandy and Smedley (2004) described a method of adding behaviour to an object by using touch sensors and a scripting language. Messages are passed between objects in order to create behaviour. An example of this kind of behaviour was used in recent the film 'Lord of the Rings' to generate soldiers used in battle scenes.

In work closer to ours, Dachselt and Rukzio (2002) introduced an XML based framework for modelling 3D objects and developed a language called Behaviour3D. Furthermore, Burrows and England (2005) suggested a language called BDL (Behaviour Description Language). BDL proposed a view of behaviour and developed software architecture, which provides the basis for implementing behaviour at run time. The behaviour description language (BDL) described is based on VRML and adds a number of features to support behavioural specification for using it with their behaviour engine.

1.3 Interaction Added to the Objects in a Scene

Interactions are used to describe the dynamic operations of the user, objects and the environment. Interactions define how a user interacts with the environment, with the objects and shows how objects interact with each other.

Hendricks, *et al.* (2003) produced a Meta authoring tool, which allows both beginner and advanced users to create virtual reality applications. This allows the migration from non-programmer to experienced programmer. According to Hendricks, *et al.* (2003) there are two ways of implementing behaviour and interaction in the virtual application:

- *Behaviour Based Interaction*: The behaviour-based interactions of objects occur according to attributes of the objects themselves, providing them a level of autonomous behaviour. For example, the behaviour of a water toy is to float on the water. The interaction of this toy with any other liquid will be the same as it is with the water due to its floating nature.

- *Event-Based Interaction:* The event-based interaction takes place when the occurrence of an event is caused by user intervention or any change in the environment. For example, turning the light bulb on and off by pressing the switch. A user interacts with the object switch and it interacts with other object light bulb.

In addition, Walczak and Wojciechowski (2005) have suggested a method for creating interactive presentation dynamically for virtual environments. A non-programmer can create objects and their actual presentation employing a user-friendly content management application. The process of creating a presentation consists of (a) creating a presentation space, (b) selecting objects, (c) selecting a presentation template and (d) setting values of object and template parameters. The same presentation space can be presented differently by the use of multiple presentation template instances associated with different presentation domains.

The emphasis of our work is to consider both scene hierarchy, yet also the concepts of behaviour and interactivity. An office, for example, is defined by the effective placement of objects, such as a keyboard, a CPU, a mouse, a table, a desk and a chair. The table may be made up of other objects: drawers, legs, etc. This table is physically linked in some way to other objects in the environment, such as the floor, the keyboard, the monitor, etc. In addition, certain objects have definable functionality allowing interaction (both with the user, but also with other objects). Each object has a physical appearance, location and orientation, as well as possibly having considerable functional potential (often similar in nature to other instances of its type).

2 Initial Development

2.1 Examples and Assumptions

A design science approach was used in our work. Accordingly simple real world scene examples were used to establish the important factors involved in the effective capture of 2D and 3D scenes. Abduction processing was used to define the following design objectives:

1. The final solution should be able to define both 2- and 3-Dimensional scenes using a XML- based description.
2. A visual representation of hierarchy and interactivity is required to support non-technical users.
3. Each object should be able to inherit the properties of other objects.
4. The design should be able to add functionality to an object.
5. The design should be able to define behaviour on the basis of properties.
6. The concept of interaction should be clearly defined.

Assumptions that were made when undertaking this research include:

1. The physical properties of an object (such as shape, appearance, translation, etc) may be derived via the current X3D language.
2. There is a predefined, yet variable, set of relationship tags that will be used to describe the scene. For example, ‘on the top’, ‘on the right hand side’, ‘touches’, etc. Such tags could be defined by the user as required.

3. The concept of adding functionality will be described using UML flowcharts. Class diagrams are functionally implemented in object-orientated programming code.

Incorporation and collaboration of current visual object-based external standards facilitates acceptance, whilst limiting the need for technical programming skills.

2.2 Capturing Hierarchy Dependencies

In our solution, hierarchical dependencies is visually represented by relational dependencies of objects (see Figure 1). For example: the Table has two draws.

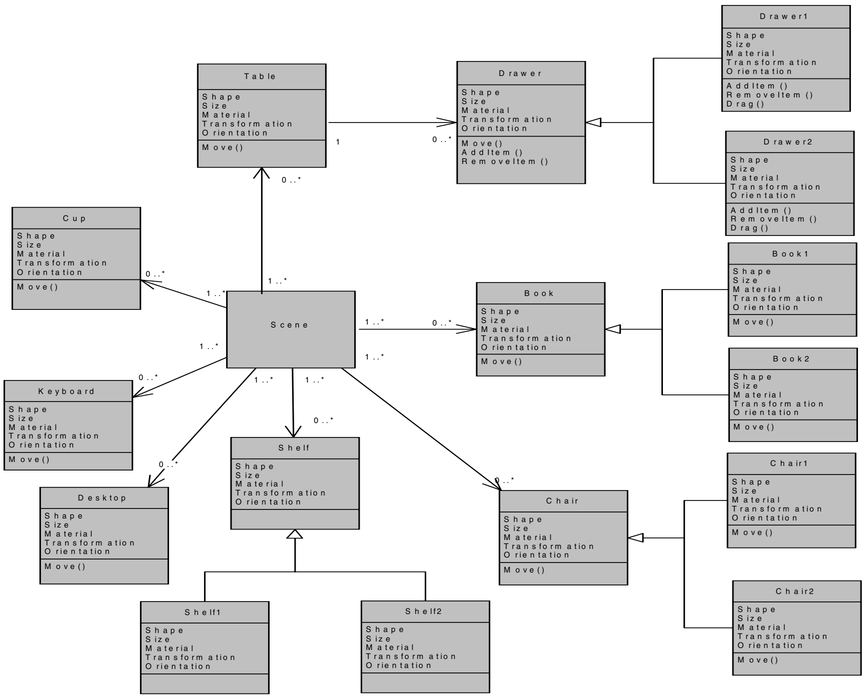


Fig. 1. Office objects defined within the scene hierarchy

2.3 Capturing Functional Dependencies

In our solution, function dependencies is visually represented by adding associated links between objects. For example: Drawer 1 On top of Drawer 2.

We made the assumption that there is a predefined, yet variable, set of relationship tags that will be used to describe the scene. Every relationship has a meaning that aids the description of interactivity. Although the list of link definitions is potentially unlimited, in our worked example interactions were restricted to: **On the top of** *(ObjA, ObjB)*; **Opposite to** *(ObjA, ObjB)*; **In the RHS** *(ObjA, ObjB)*; **In the LHS** *(ObjA, ObjB)*; **Next to the** *(ObjA, ObjB)*; **Touches** *(ObjA, ObjB)*. This is then visually represented using visual connections between object instances (see Figure 2).

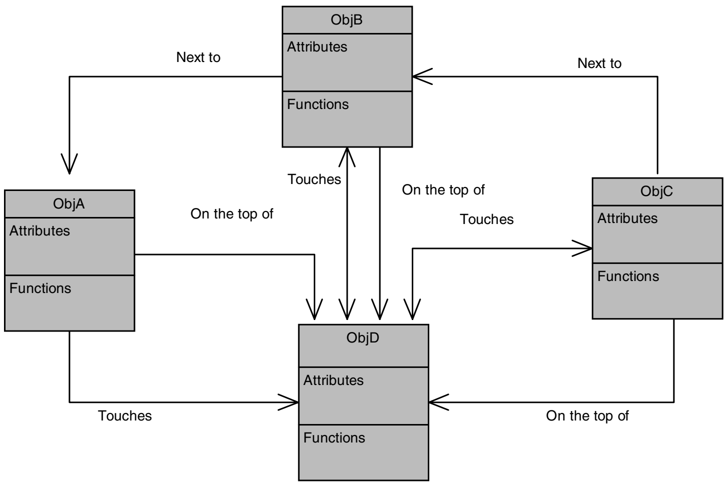


Fig. 2. Visual demonstration of object functional dependencies

Parameter information can be incorporated by manipulating the interactions. In our example, distance was added as the total length measured between two points (currently unit independent). Distance, as a parameter, can be used to calculate coordinates of the objects on the basis of the relationships that exist among objects. If

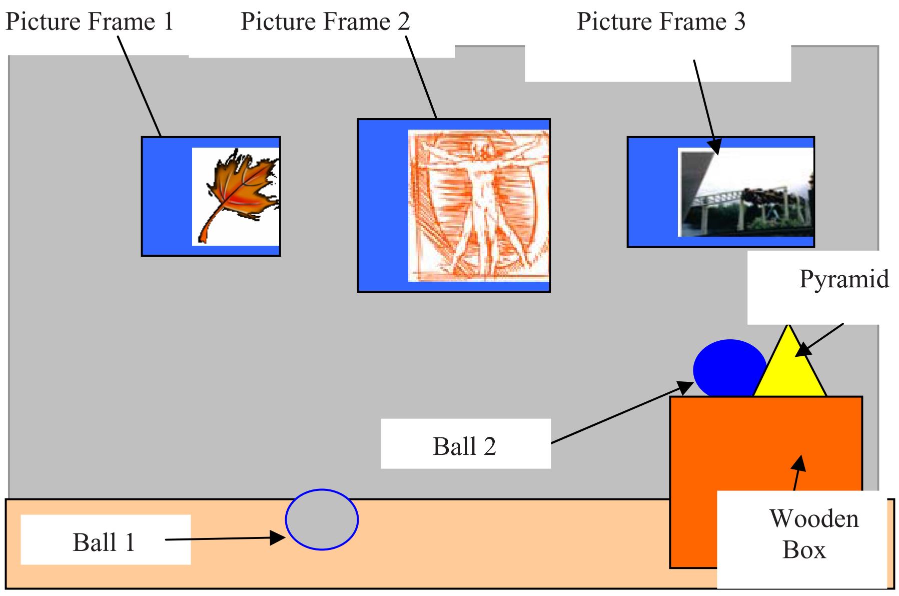


Fig. 3. Two-Dimensional Scene of a child's room

the objects used in the scene are cake (ObjA) and a party hat (ObjB). The relationships between these objects are "*ObjA may be to the RHS of ObjB*", "*ObjA is next to ObjB*" The distance between these two objects is given as 5. Interesting, as more interactions, both functional and parameter information, is defined between objects, more specific information can be determined concerning both the physical and functional dependencies in the model.

To explicate a scene, a simple layout has been chosen for description (see Figure 3). This scene is a room, which has got a number of things such as three picture frames, two balls, one pyramid and a wooden box (see Figure 3). The system requires coordinates for one of the object. User is supposed to describe the scene on the basis of relationship exists among these various objects. The scene is depicted using two dimensions of the objects

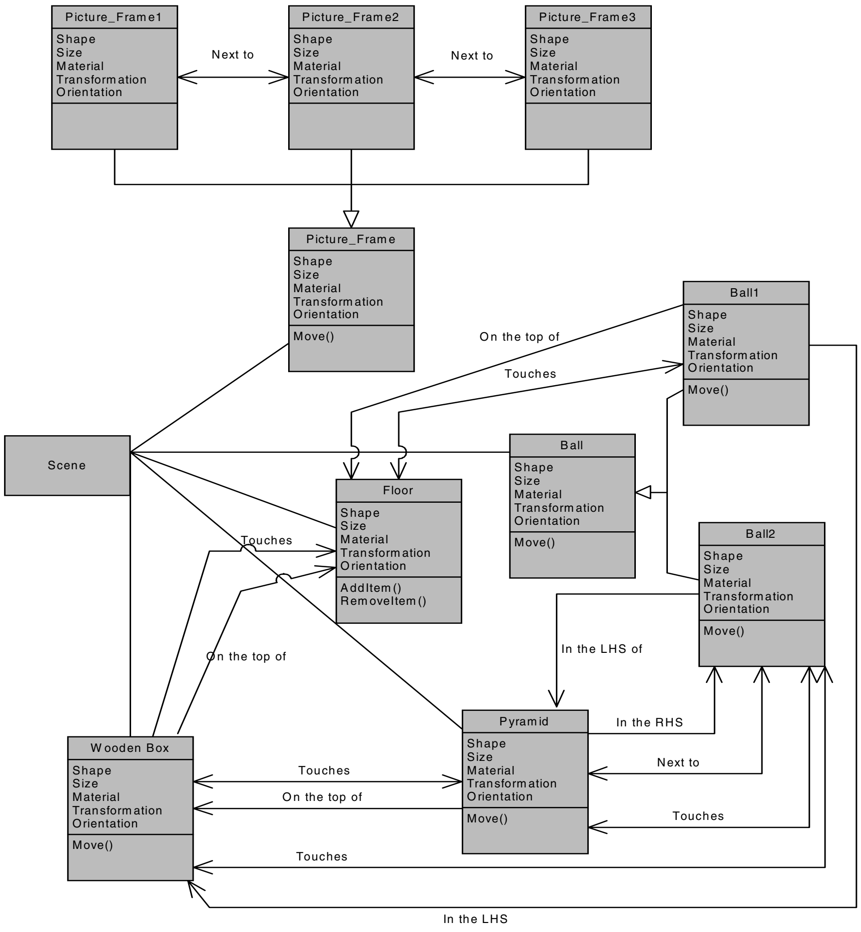


Fig. 4. Child's Room: objects and their functional relationships

In our example (Figure 3), there is a ball on the floor; it is also **touching** the floor. There is another ball, which looks similar to the ball on the floor but this ball is **on the top of** a wooden box. The bottom of ball **touches** the wooden box. A pyramid is **on the top** of box. It is placed **in the right hand side** of the second ball. The ball **touches** this pyramid. Both of the objects are **on the top of** wooden box and touch the box. There are three picture frames hanging on the wall. One of them is **in the right** corner of the room; second picture frame is **next to** it. The distance between these two frames is about 10m distances. The last picture frame is next to second picture frame and the distance is about 8m.

Visual description of functional relationships between objects allows a user to describe relationships in a scene in a natural form. Moreover, the user is not concerned with the co-ordinates of the object to describe the scene. The scene that is described above in Figure 16 can be explained using simple sentences i.e. in natural way. For instance the scene is described here in simple language as:

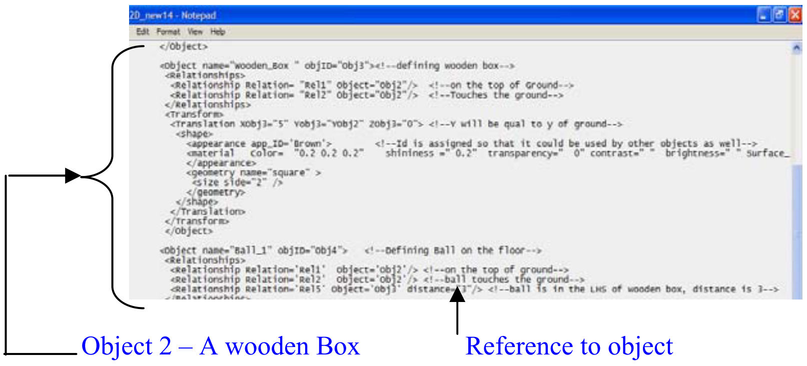
The relationship used for describing the scene is differentiated using bold fonts. The objects used in the scene are listed below: *Floor, Ball1, Ball2, Pyramid, Wooden Box, Picture Frame1, Picture Frame2, Picture Frame3*. The relationship include: **Next to** *(Picture1, Picture2)*; **Next to** *(Picture3, Picture2)*; **On the top of** *(Ball1, Floor)*; **Touches** *(Ball1, Floor)*; **On the top of** *(Pyramid, Wooden Box)*; **On the top of** *(Ball2, Wooden Box)*; **Touches** *(Ball2, Wooden Box)*; **Touches** *(Ball2, Pyramid)* **Part=** *"Right side";* **Next to** *(Ball2, Pyramid)*; **In the RHS** *(Pyramid, Ball2)*; **In the LHS** *(Ball2, Pyramid).*

All the above relationships amongst objects was then represented in a visual way. Figure 4 clearly presents all the objects and the relationships that exist among these objects. All objects have a number of relationships with other objects. As shown in the diagram, classes for all three frames; thus *pictureframe1*, *pictureframe2* and *pictureframe3* are subclasses of the super class picture frame. According to the nature of child class, these three child classes inherit the properties and function of the parent class named picture frame. *Ball1* and *Ball2* also inherit the features of the parent class named *Ball.* This will help to reduce the complexity of the program as only one class will be needed to define and child classes will use inheritance methods in order to appear and perform the same functions. The next section will discuss the coding part for this scene.

3 XML File Structure

Each relationship is given an identity so that it can be used by each of the objects (see Figure 4). All information about relationships and locations of objects was described. A file link was used to describe the image appearance.

Shape / appearance of objects are described using current modelling languages; size / scale, orientation and object material / texture was dynamically manipulated in object parameters. Documentation of functionality and inheritance was achieved by launching a java-based programming environment and using java functional inheritance properties.



```
</Object>

<Object name="wooden_Box " objID="Obj3"><!--defining wooden box-->
 <Relationships>
  <Relationship Relation= "Rel1" Object="Obj2"/>  <!--on the top of Ground-->
  <Relationship Relation= "Rel2" Object="Obj2"/>  <!--Touches the ground-->
 </Relationships>
 <Transform>
  <Translation XObj3="5" YObj3="YObj2" ZObj3="0"> <!--Y will be qual to y of ground-->
   <Shape>
     <appearance app_ID='Brown'>        <!--Id is assigned so that it could be used by other objects as well-->
     <material   Color=  "0.2 0.2 0.2"    shininess =" 0.2"  transparency="  0" contrast=" "  brightness=" " Surface_
     </appearance>
     <geometry name="square" >
      <size side="2" />
     </geometry>
   </Shape>
  </Translation>
 </Transform>
 </Object>

<Object name="Ball_1" objID="Obj4">   <!--Defining Ball on the floor-->
 <Relationships>
  <Relationship Relation='Rel1'  object='obj2'/> <!--on the top of ground-->
  <Relationship Relation='Rel2'  object='Obj2'/> <!--ball touches the ground-->
  <Relationship Relation='Rel5' Object='Obj3' distance="3"/> <!--ball is in the LHS of wooden box, distance is 3-->
```

Fig. 5. 'A child's bedroom' - automatically generated from visual interactivity relationships

4 Conclusion

A XML- based description is described, that supports: a visual representation of hierarchy and interactivity; inheritance of properties from other objects; addition of object functionality and behaviour on the basis of properties. Previous scene description standards have been developed, yet they do not cover the multi-dimensional requirements that appear key to the capture of a dynamic scene. By incorporating multiple standards, external programming tools, we have taken positive steps towards a more eclectic implementation solution, for use in scene description.

References

1. Arjomandy, S., Smedley, J.T.: Visual Specification of Behaviours in VRML Worlds. In: Proceedings of the ninth international conference on 3D Web technology, Monterey, California, USA, pp. 127–133 (2004)
2. Del Bimbo, A., Pala, P., Santini, S.: Visual image retrieval by elastic deformation of object sketches. In: Proceedings of IEEE Symposium on Visual Languages, pp. 216–223 (1994)
3. Carson, S.G., Pulk, F.R., Carey, R.: Developing the VRML 97 International Standard. IEEE Computer Graphics and Applications 19(2), 52–58 (1999)
4. Dachselt, R., Rukzio, E.: Behaviour3D: An XML based Framework for 3D Graphics Behaviour. In: Proceeding of the Eighth international Conference on 3D Web Technology, Saint Malo, France, 101–ff (2003)
5. Hendricks, Z., Marsden, G., Blake, E.: A Meta-Authoring Tool for Specifying Interactions in Virtual Reality Environments. In: Proceedings of the 2nd international Conference on Computer Graphics, Virtual Reality, Visualisation and interaction in Africa, Cape Town, South Africa, pp. 171–180 (2003)
6. Hultquist, C., Gain, J., Cairns, D.: Affective Scene Generation. In: Proceedings of the 4th international Conference on Computer Graphics, Virtual Reality, Visualisation and interaction in Africa, Cape Town, South Africa, pp. 59–63 (2006)
7. Macdonald, J.A., Brailsford, F.D., Bagley, R.S.: Encapsulating and manipulating Component object graphics (COGs) using SVG. In: Proceedings of the 2005 ACM Symposium on Document Engineering, Bristol, UK, pp. 61–63 (2005)

8. Meyer, T., Conner, B.: Adding Behaviour to VRML. In: Proceedings of the First Symposium on Virtual Reality Modeling Language, San Diego, California, USA, pp. 45–51 (1995)
9. Nadeau, R.D.: Building Virtual Worlds with VRML. IEEE Computer Graphics and Applications 19(2), 18–29 (1999)
10. Nadeau, R.D., Moreland, L.J.: The Virtual Behaviour System. A behaviour Language Protocol for VRML. In: Proceedings of the First Symposium on Virtual Reality Modeling Language, San Diego, California, USA, pp. 53–61 (1995)
11. Walczak, K., Cellary, W.: Building database Applications of Virtual Reality with X-VRML, Tempe, Arizona, USA, pp. 111–120 (2002)
12. Walczak, K., Wojceichowski, R.: Dynamic Creation of interactive mixed reality presentations. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, Monterey, California, USA, pp. 167–176 (2005)