Presentation Dynamism in XML

Patrick Schmitz, Ludicrum Enterprises; Simon Thompson, University of Kent; Peter King, University of Manitoba

We have defined three extensions to XML-based languages

- Parameterised templates
- Attribute values as calculated expressions
- Author defined events

and illustrated them in an example using XHTML, SMIL animation and SVG graphics.

Here we present the extensions through an example. Full details of the extensions, their implementation and further case studies are available at

http://www.cs.kent.ac.uk/~sjt/PDXML/PDXML.pdf

Fundamental to the declarative approach in XML-based languages is the idea that authors describe *what* a presentation should be, rather than *how* that effect should be achieved. A language like SVG or SMIL is a domain specific language (DSL) describing graphic or multimedia artefacts.

A DSL has limitations: what should we do when we want to do something that the DSL does not support?

- We could *embed* the DSL in a programming language; Haskell and other functional languages are used to do this. This requires the author to become a programmer.
- We can write definitions in *script*. This completely breaks the declarative model of authoring.
- **We can use our proposal: *extend* the capabilities of the declarative, XML-based, language.**

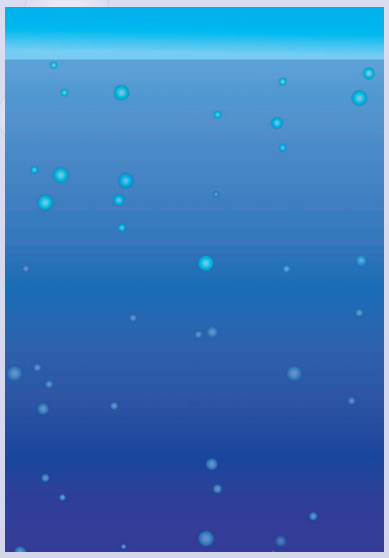
Event use

We use a template to define a bubble. Instances define the size and initial position.

The duration of the animation of a bubble is a function of its size. This requires calculation of the dur attribute.

Bubbles burst when they reach the surface; we use an event to specify this condition which ends the animation.

The size and xOrigin are given random values, using a random number generator in a calculated expression.



Template definition

```
<template id = "bubble">
  <param name = "xOrigin" />
  <param name = "size" />
  <circle cx = "$xOrigin" r = "$size" ... >
    <animate attributeName="cy" from="100%" to="10%"
          dur="calc(10-($size/2)),atStart)" />
  </circle>
</template>
```

Template instance and calculation

```
<instance xlink:href="#bubble">
  <param name = "xOrigin" value="calc(rand(0, 100),atStart)" />
  <param name = "size"    value="calc(rand(5, 15),atStart)" />
</instance>
```

Event definition

```
<event target = "bubble"  type= "atSurface"
   predicate = "bubble.cy <= 10%" />
```

Event use

```
<audio src="pop.mp3"
  begin="bubble.atSurface" />
```

Template definition

Templates permit the creation of an arbitrary number of similar objects. The template has a name which appears in instantiations via #name. The template may have an arbitrary number of parameters. Each parameter is defined by a <param> element, which may contain a default value for the parameter. Within the template element, a formal parameter is referenced by prefacing its name by $, as in $size.

Template instance

An instance is given by an <instance> element. Instances refer to the template using the template ID. In the SVG example we have used the xlink:href attribute for this purpose.

This is more powerful than the SVG <defs> and <use> mechanism for two reasons:

- Parameters can, in principle, be arbitrary, whereas in SVG it is only pre-exisiting attributes which can be assigned on instantiation.
- Since each instance creates a new object – rather than a 'shadow' – instances can be the target of animation and so forth.

Calculated expressions

We allow expressions rather than literal values to be used to give values to attributes. Expressions to be evaluated are called out using the keyword calc.

An expression is re-evaluated whenever a referenced property changes its value, unless explicitly specified otherwise.

Event definition

Events signal a user-defined condition such as a collision or a limit condition. An event is characterised by a Boolean-valued expression, or predicate. The event occurs when the value of the predicate turns from false to true. In the example, the predicate signals the arrival of the bubble at the surface of the water.

Event use

A user-defined event is used in exactly the same way as language-defined events such as mouse clicks. In particular, an event may be used to begin or end a timed element (e.g., an animation).

Prototype Implementation in Internet Explorer 6

Templates: For each instance, we copy the content of the template, replacing the parameter references by their values. To remove the template itself from the display graphs, we use a CSS stylesheet to set the CSS display property to none.

Expressions: IE behaviours locate the calc expressions, whose values are then calculated. Using a dependency graph, expression values are recalculated (only) when the values of constituent parts change.

Events: An IE behavior creates and fires an event when the value of the predicate expression – calculated as above – changes from false to true.

Future Developments

The work raises a number of questions:

- We can calculate attributes such as from, to, by, etc. How limited is this in practice; what other items would it be useful or feasible to animate?
- Can we calculate timing attributes so that timing is computed?
- How should scoping of identifiers ('local id-spaces') be added to XML languages?
- Can structured types be accommodated within the XML approach?
- What are the analogues of classes and inheritance for templates?

More generally, we want to explore the implications of our proposed extensions for XML-based languages in general.