Multimedia Systems 10: 245–260 (2005)
Digital Object Identifier (DOI) 10.1007/s00530-004-0155-2

Multimedia layout adaptation through grammatical specifications

Kang Zhang, Jun Kong, Meikang Qiu, Guang-Lei Song

Department of Computer Science, The University of Texas at Dallas, Richardson, Texas 75083-0688, USA

Published online: 2 February 2005 –

Abstract. Online multimedia presentations, such as news, need to be constantly updated. Increasing demands are also being made for accessing online multimedia documents from mobile devices such as PDAs. There is an urgent need for a sound but practical formalism that supports automatic adaptation to the change of media content, display environments, and user intention. This paper presents a visual language approach to the layout adaptation of multimedia objects. The underlying theory of our approach is a context-sensitive graph grammar formalism enriched with facilities for spatial representation and specification. The paper focuses on the issues and techniques for size adaptation and style adaptation in response to the change of device requirements and user interactions.

Keywords: Multimedia authoring and presentation – Graph transformation – Graph grammars – Visual languages – Adaptive presentation

1 Introduction

With rapid advances in Internet and Web technology, an increasing volume of graphs and media content is delivered over the Web. The content and the presentation structure of an online multimedia presentation may also be frequently updated. On the client side there are various kinds of viewing conditions, such as varying screen size, style preference, and device capabilities. For example, consider a diagram representing an organizational structure on the Web that may be of considerable complexity occupying a large screen space, and thus may be unsuitable for small displays [18]. Thus, if the diagram is to be viewed on the screen of a mobile device, such as a PDA (Personal Digital Assistant), the original diagram layout may not be appropriate. Another example is a news Web site, which generally needs to be constantly updated with incoming news items. Such a site may have to adapt itself frequently to the changing space and style requirements for different news categories. It would be highly desirable for the site to be able to dynamically adapt its layout.

Correspondence to: Kang Zhang (kzhang@utdallas.edu)

With current document markup languages such as HTML and WML the layout of a Web page is relatively static and fixed [4]. When a user's requirement or the device capability is changed, the layout may become unsatisfactory. The reason is that such markup languages do not provide any mechanism powerful enough for specifications to be adaptable to the changing context. Though SMIL [35] and CSS [36] provide more flexible markups for multiple alternative layouts, the markups provide absolute layout functionality rather than being adaptable to user intentions or the existing layout. Therefore, a metalevel design mechanism capable of adapting multimedia presentations in response to dynamic changes in information content is highly desirable.

To illustrate the concept of multimedia adaptation that we perceive, we use Ishizaki's schematic diagram of a process between content creation and information reception [14] as depicted in Fig. 1. The design system should be able to adapt itself to the changes in information content and in individual users' intentions. As mobile devices provide an increasing proportion of online content accesses, we argue that a multimedia authoring system should support an additional type of context changes – i.e., adaptation to the change of device capabilities. In other words, the designer of a multimedia system needs to be able to specify how the presentation would evolve based on the change of environments (e.g., from a desktop screen to a mobile display panel), user intention (e.g., zooming in or out), and information content (e.g., news update).

This paper presents a visual language approach, specifically a spatial graph grammar, for adaptive multimedia authoring and presentation. The approach is highly intuitive yet also sound in theory. The central theme of this paper is to demonstrate how to use a graph grammar formalism to spec-

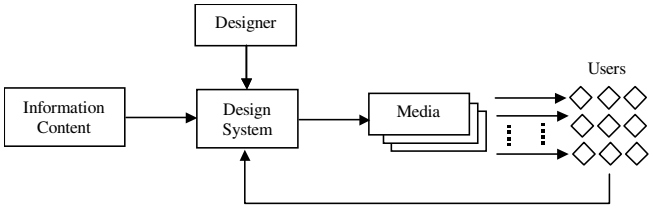


Fig. 1. Multimedia presentation design and delivery process

ify and support automatic transformation and adaptation of multimedia presentations. The approach has two major advantages due to its metatool capability: a graphical authoring tool can be automatically generated by a visual language generator, such as VisPro [38]; and the generated authoring tool can be used by novices who have no computing knowledge.

The remaining part of the paper is organized as follows. Section 2 introduces the visual language approach using graph grammars, followed by a description of spatial grammatical representation and specification in Sect. 3. Section 4 focuses on adaptation techniques to support size and style changes of multimedia presentations. Sections 5 and 6 demonstrate size and style adaptations with two real-world examples. Section 7 presents a system architecture implementing the grammatical approach. Related work is reviewed in Sect. 8. Finally, Sect. 9 concludes the paper and proposes future work.

2 A visual language and graph grammar approach

Information visualization has played an important role in facilitating easy comprehension of sophisticated systems [13]. Applying information visualization to programming, research in visual languages [6] aims at effectively improving programming productivity by applying graphical user interface technologies to support program construction. Popular visual languages include UML, automata, Petri nets [38], etc.

Designers of complex systems typically use diagramming methods as conceptual devices to organize their design space. Compared with text, graphs can represent semantic and structural information more intuitively. In most visual languages, programs are represented as graphs and thus called *visual programs*. Such graphs to be compiled/parsed are called *host graphs*. A visual programming environment includes a visual editor for graphical construction of host graphs representing visual programs and a parser for validating the syntax of host graphs. A visual language may be defined by a *graph grammar* [26], which consists of a set of rewriting rules called *productions*, as shown in Fig. 2a in the form of *Reserved Graph Grammars* (RGGs) [39]. Such a grammar is used to generate the parser for the language. Each production consists of two subgraphs, called *left graph* (with one node named "Section" in Fig. 2a) and *right graph* (two nodes interconnected by an edge). The nodes of the graphs symbolize real-world objects, and edges between nodes represent predetermined relationships. A subgraph in the host graph is called a *redex* if it is isomorphic to the left or right graph.

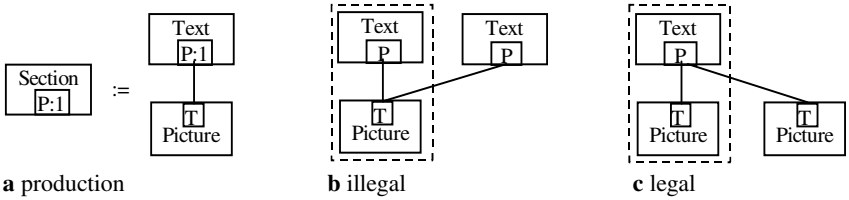
A graph transformation process is a sequence of applications of productions. Applications can be L-applications or R-applications. An L-application (or R-application) is to find a redex that matches the left (or right) graph and replace it with the right (or left) graph of a production. One of the most difficult problems with graph transformation systems is deciding which applications are allowed and which are disallowed. Even for the most restricted classes of graph grammars the membership problem is NP-hard [25].

Another obstacle restricting applications of graph grammars is that most proposed parsing algorithms [11,15,34] are based on context-free graph grammars. Many interesting graphs, however, cannot be specified by pure context-free grammars. Additional control mechanisms are necessary for context sensitivity. In many applications, the logic structure of a graph is too complex to be defined by a context-free grammar.

As the underlying formalism for automatic generation of visual languages [38], Zhang et al. proposed a context-sensitive graph grammar called RGG (Reserved Graph Grammar) [39]. It is context-sensitive since its right and left graphs can have an arbitrary number of nodes and edges. In the RGG, a node has a two-level structure: the node itself and the small rectangles embedded in the node called *vertices*. An edge is uniquely determined by two vertices in the participating nodes. All vertices in a node should be uniquely labeled. The RGG introduces context information with a simple embedding rule and is thus sufficiently expressive to handle complicated programs. In order to identify any graph elements that should be reserved during the transformation process, we *mark* each isomorphic vertex in a production graph by suffixing its label with an integer unique in the node. The purpose of marking a vertex is to preserve the context. We introduce the following embedding rule:

> *If a vertex in the right graph of the production is unmarked and has an isomorphic vertex v in the redex of the host graph, then all edges connected to v should be completely inside the redex.*

The theoretical treatment of the marking mechanism is provided elsewhere [39]. Consider the example production in Fig. 2a; vertex P in node *Text* is marked, while T in *Picture* is not. Assume that a *Text* node may connect to multiple *Picture* nodes, while a *Picture* node is allowed to connect to only one *Text* node, as a story (*Text*) may be illustrated by several photos (*Pictures*) but not vice versa. Such a restriction is easily expressed by marking P and leaving T unmarked in the definition of the production (as in Fig. 2a). According to the embedding rule, the isomorphic graph in Fig. 2b is not a redex because vertex T in node *Picture* has an edge that is not inside the redex (isomorphic graph in the dashed box) while its isomorphic vertex in the right graph is unmarked. Therefore, the graph in Fig. 2b is invalid. On the other hand, the graph in Fig. 2c is valid according to the embedding rule. There is a redex (in the dashed box) in the graph because the vertex of *Text* connecting to *Picture* has its isomorphic vertex marked in the right graph of the production, even though it has an edge connected outside the redex. It is this structure that makes an RGG effective in specifying a wide range of visual languages



a production **b** illegal **c** legal

Fig. 2. A production in Reserved Graph Grammar (**a**) and its application in host graphs (**b**,**c**)

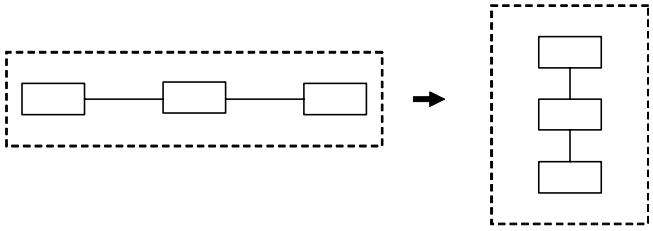


Fig. 3. Horizontal layout (**a**) transformed to vertical layout (**b**)

and efficient in parsing a certain class of visual languages. An RGG is equipped with a deterministic parsing algorithm, called *selection-free parsing algorithm* (SFPA), that only tries one parsing path. Zhang et al. prove that the time complexity of SFPA is polynomial [39].

To apply the marking technique to multimedia adaptation, consider a simple example in graphical presentation: a vertical layout provides a different visual perception and requires a different screen (usually smaller) estate from a horizontal layout, as shown in Fig. 3. This is one of the most common issues in graphical design and can be effectively applied to transforming Web graphics to suit small-screen mobile devices. Figure 4 depicts the rewriting rule (production) for this required transformation.

Since there may be multiple nodes chained in the same direction, we mark the vertices on both ends of the two nodes by attaching unique integers to the vertex labels (i.e., "N:1" and "S:2"). This means that during transformation, the edges connected to both ends will be reserved. The direction change from horizontal to vertical is reflected in the positions of the vertices, as explained in Sect. 3.1. The edges between the nodes will be shortened after transformation, as specified by "-S" and "-N". The next section will provide more details on the visual notations.

The original RGG defines the logical relations among constructs of a graph. To support graphic layout, it may be extended to take constraint rules [19] represented by (x, y) coordinates, as in our earlier work [41]. Doing so, however, sacrifices the intuitiveness and visualization potential of the RGG. In this paper, we follow the general principle of context sensitivity while increasing expressiveness and propose a set of visual notations to enhance the RGG with the capability of spatial specifications. The enhanced RGG formalism is consistent with the original RGG but more powerful for applications that require spatial knowledge, such as Web page design, graph layout, multimedia authoring, graphical user interface design, and PCB design. Furthermore, parsing spatially extended graph grammars can be faster than parsing the original RGGs due to the additional spatial information [16].

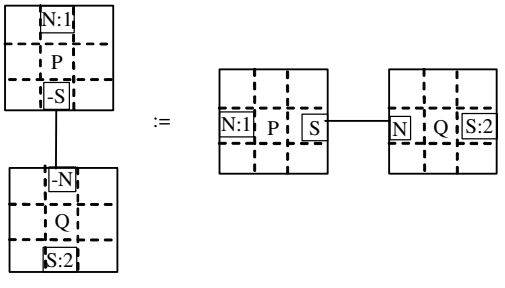


Fig. 4. A production for the transformation in Fig. 3

3 Spatial relationships and representations

When considering adding spatial notations to the RGG formalism, we generally aim at

- Retaining the original RGG syntax and semantics and
- Introducing minimal additional notations that are intuitive for spatial specifications.

We propose five categories of spatial relationships between any two given objects: *direction*, *distance*, *topology*, *alignment*, and *size*. When used to specify desired layout rearrangement, the relationships can be used to specify changes to be applied during graph transformation. The spatial relationships are currently restricted to a two-dimensional multimedia design space. It is entirely feasible that such relationships and notations could be extended to a three-dimensional space for specifying a virtual reality environment.

All the spatial relationships are defined between two objects, one referred to as the *primary object* and the other as the *reference object*. When discussed in the context of a spatial grammar, media objects are represented and referred to as *nodes* in grammar productions, as termed in Sect. 2.

3.1 Direction

To represent the relative direction between two nodes, each node is arranged as a 3×3 grid in dotted lines inside the node, as shown in Fig. 5. The central region (marked C) represents the node itself. Surrounding the central region, the eight grid regions represent eight directions: N (north), NE (northeast), E (east), SE (southeast), S (south), SW (southwest), W (west), and NW (northwest), in clockwise order. Each of these directions indicates the relative position of the reference object connected to the current object (primary object). The boundary of the area occupied by the reference object is outlined by dotted lines surrounding the primary object.

Each of the eight direction regions may include more than one vertex. The objects that are connected to a primary object

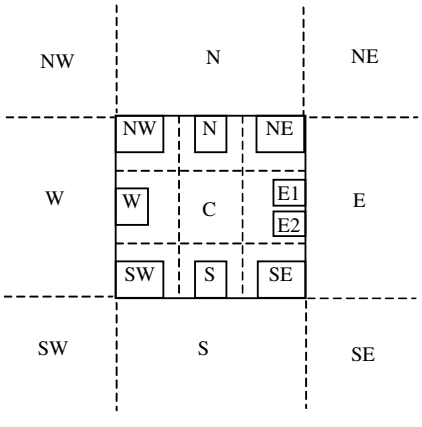


Fig. 5. Notation for direction relationships

through the vertices of the same region are in the same direction. For instance, the east region of the node in Fig. 5 has two vertices, E1 and E2, if there are two nodes connected to E1 and E2 from the right side of the present node.

3.2 Distance

The distance between two objects' centers measures an important class of spatial relationships. To specify the distance relationship, we prefix a "+" to the vertex label to indicate a long (or increased) distance to the object that the vertex connects to, "−" to indicate a short (decreased) distance, and blank to represent a distance not emphasized (or not changed). Four special cases of distances are treated separately as topological relationships due to their importance in spatial reasoning.

3.3 Topology

We can generally define four topological relationships between two nodes: *nonoverlapping, overlapping, touching*, and *containing*. Assume that D_x is the set of all points on an object x, and B_x ($\subseteq D_x$) is the boundary point set of x. Considering a primary object a and a reference object b and $D_a \cap D_b = R$, four topological relationships are defined as follows:

- a is *nonoverlapping* with b iff $R = \Phi$;
- a is *overlapping* with b iff $R \neq \Phi$, and further:
  - a is *touching* with b iff $R \subseteq (B_a \cap B_b)$; or
  - a is *containing* b iff $D_b \subseteq D_a$.

Using a rectangle to represent an object, Fig. 6 shows the four types of topological relationships. *Nonoverlapping* indicates that there is no common point on both involved objects. *Overlapping* means that there are common points between the two objects. It is represented by dotted lines on the boundary of the overlapped area. We define *touching* and *containing* as two special cases of overlapping. If common points exist only on the boundaries of two objects, the objects are *touching* each other. The touched part is represented by a dotted line. *Containing* means that all the points on one object belong to the other. In Fig. 6, the boundary of an object is totally dotted, indicating that the object is contained in the other object.

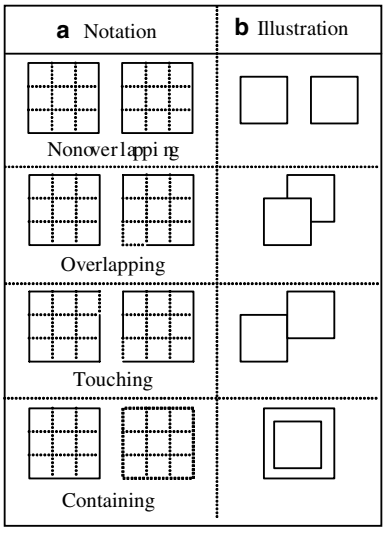


Fig. 6. Topological relations

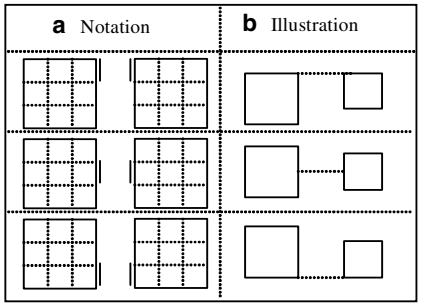


Fig. 7. Alignment relations in horizontal direction

3.4 Alignment

Two objects may be aligned vertically or horizontally, along the directions of N, S, W, or E. In the horizontal direction, we define three different horizontal alignment cases for each object, i.e., top alignment, bottom alignment, and center alignment, giving a total of 9 different alignment relationships between any two objects. The alignment relationships in the vertical direction are similarly defined. The boundary of a node is divided into 12 segments according to the 3×3 grid. A bold segment is used to indicate the alignment relationship. Figure 7 illustrates the three most common alignment relationships.

3.5 Size

Transforming a multimedia document from a desktop Web page to a PDA display may involve size changes of various media objects. To represent the changes, we mark the node's center box with a "+" to indicate that the object is large in size (or zoomed in to become larger), "−" to indicate small in size (or zoomed out to become smaller), and blank to indicate a size not emphasized (or unchanged).

3.6 Event driven

As discussed above, the spatial relationships can be used to specify a static layout structure or some predetermined structural changes. Similarly, the SWITCH construct in SMIL [35] allows different layouts to be applied upon different predetermined conditions, which are defined through test attributes. Many media players, however, do not provide sufficient support for the specification and run-time evaluation of application-dependent test attributes. On the other hand, the grammatical approach allows great flexibility to associate domain-specific triggering conditions with productions without relying on the media player's capability. There are increasing demands for providing users a sense of focusing, realized by interactively changing the details of certain parts of a multimedia document during viewing. Such a mechanism is called *interactive semantic zooming* [18].

To address the dynamic issues, we classify graph productions into conditional and unconditional ones. A transformation is performed on an unconditional production when a redex

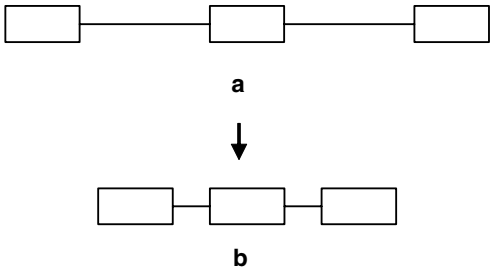


Fig. 8a,b. An example of differential scaling

is found in the host graph that matches the right graph of the production. A conditional production can only be triggered by a specific event, such as the change in the device's capability, the user's interaction, etc. Since graph transformations can be performed according to dynamic events, such as user inputs, the appearance of a multimedia document may be adjusted by triggering such conditional productions when the layout structure, user interaction method, or display environment needs to be changed. More flexible than SMIL SWITCH, conditional productions are able to handle user interactions and individually produce local effects.

3.7 Syntax-directed computations

The RGG supports syntax-directed computations by associating data and operators to nodes in productions in terms of *attributes* and *actions*. An attribute expresses a piece of data related to the object represented by a node and can be retrieved and evaluated in the process of parsing. Different actions can be performed on different attributes of the redex of a production to achieve the desired execution effects. Writing an action code is like writing a standard exception handler in Java by treating each attribute as an object. Attributes can supplement graphical and qualitative specifications by providing precise quantitative values. For example, to shorten the distance between two connected nodes by half as in Fig. 8, we can attach the following action code to the production in Fig. 9:

```
Action(AAMGraph g) {
   int OriginalDistance = Q.left - P.right;
   P.right = P.right + OriginalDistance/4;
   Q.left = Q.left - OriginalDistance/4;
}
```

In summary, a visual representation defines an approximation of layout while attributes and action codes supplement qualitative specifications with precise quantitative information and associated computation.

4 Adaptation to context changes

As discussed at the beginning of the paper, context changes may be due to a change in information content such as a traffic monitoring system, device capabilities such as from a desktop screen to a PDA panel, or viewer intention. Our graph-grammar-based approach is able to adjust the appearance to different displaying environments.

This section outlines automatic adaptation of the size and style of a multimedia presentation in response to any of these changes, though other aspects of adaptation may also be supported by the graph grammar approach. Detailed examples of adaptive presentations will be discussed in Sects. 5 and 6.

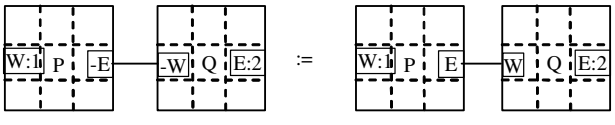


Fig. 9. Grammar specification of differential scaling in Fig. 8

4.1 Size adaptation

The most typical application of size adaptation is for Web display layout to be reduced to suit mobile devices. The simplest solution to the problem of the limited screen size is *linear scaling* (or *normal zooming*), but this is often not the best way. A more elaborate technique is *differential scaling*, in which different components of a document are scaled differently. *Differential scaling* is effective in compressing white spaces. For example, rather than performing a linear scaling, each white space is compressed while the box sizes are maintained [18], as illustrated in the simple example in Fig. 8. To specify such a transformation, we can use distance relationships as shown in Fig. 9.

To represent the change of a node size, we use "+" in the node's center box to indicate that the node will become larger (or zoomed in as discussed below) in the transformation, "−" for smaller (zoomed out), and blank for unchanged size.

4.2 Style adaptation

To suit different display spaces and devices, the layout of individual media objects and that of the entire document may need to be adapted. One such adaptation techniques is known as *alternative layout*. Figure 10 illustrates a typical example of alternative layout. Originally object B is on the right of object A. After transformation, as in Figs. 10b and 11b, object B is at the bottom of object A, and thus the connecting direction is changed.

Another type of multimedia style adaptation is called *semantic zooming* [18]. For varying interest in detail, an adapted

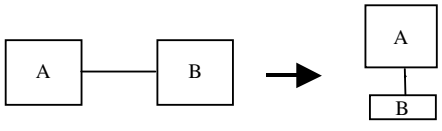


Fig. 10. Semantic zooming with an alternative layout

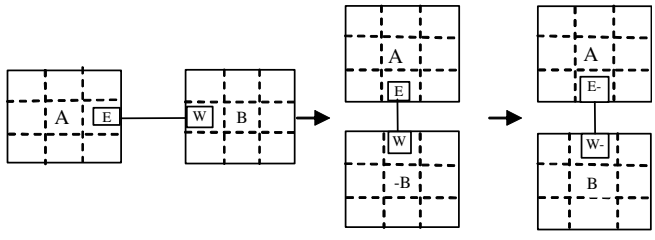


Fig. 11. Application of semantic zooming and distance rules to achieve the effect in Fig. 10

layout may initially show one level of detail. It allows the viewer to zoom in hierarchically while adapting the layout level of each individual component or group of components to the available screen size or to the viewer's preference. For example, we may need to enlarge one part, in which the user is particularly interested, while compressing unrelated parts, as illustrated in Fig. 10. We need to look into the detail of object A first, so we may view the details of A and B separately. Figure 10 illustrates the combined effects of alternative layout and semantic zooming. Figure 11 depicts the snapshots of using grammatical rules to achieve the style transformation from **a** to **b**, including the reduced size of B, and shortened distance between A and B in **c**.

In some systems [20], the above viewing technique is called *fisheye* view. The more commonly accepted concept of fisheye views refers to the geometric distortion technique when highlighting a focused area of a large display [27]. Geometric distortion enlarges the focused area while proportionally reducing other areas depending on their distances to the focused area. Hyperbolic trees offer another similar viewing technique and are widely used for Web browsing.

5 Example 1: Adapting sizes for PDA displays

This and next sections focus on two detailed examples of size and style adaptations through grammatical specifications and graph transformations. This section describes how to transform a desktop Web page to several small pieces for mobile Web browsers – an example of size adaptation.

5.1 Original Web and resulting PDA presentations

Figure 12 shows the popular NASA home page, whose size and layout may be adapted to suit small-screen Web browsers. We will transform this page into the WML format to be displayed on PDAs. The XML description for the above Web page is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<page>
   <section1>
      <block1>
         <Logo>
            <pic>
               <id> nasa </id>
               <source> ./images/nasa.bmp </source>
            </pic>
         </Logo>
         <text>
            02.01.03 Building Planets in Cyberspace
         </text>
      </block1>
      <block2>
         <theme>
            <pic>
               <id> shuttle </id>
               <source> ./images/shuttle.bmp </source>
            </pic>
         </theme>
         <button>
            <link>
               <pic>
                  <id> missions </id>
                  <source> ./images/missions.gif </source>
               </pic>
               <href>
                  http://www.nasa.gov/missions/current/
               </href>
            </link>
            ...
         </button>
      </block2>
   </section1>
   <section2>
      <pic>
         <id> improve life </id>
         <source> ./images/improvelife.bmp </source>
      </pic>
      ...
   </section2>
</page>
```

Assume the desirable outcome as illustrated in Fig. 13. We divide the original Web page into four small pages based on the four images and copy the top-left heading information and top-right hyperlinks to all the small pages. As a result, each small page contains three parts: the top part contains date and

Fig. 12. The original NASA homepage

Fig. 13. Resulting presentation as four pages on a PDA

title (tagged "Text") and NASA logo ("Logo"), the middle part is an image ("Theme" or "Picture"), and the bottom part contains three hyperlinks ("Link").

The output tree structure is translated into a WML document. Each page, or a single interaction between a user agent and a user, is known as a *card*. One advantage of this arrangement is that multiple screens can be downloaded to a client in a single retrieval and vice versa. Our task is simply to transform the XML description into several cards, each to be displayed as a PDA page. The following is part of the WML document for the PDA presentation in Fig. 13, where "card" represents a separate page:

```
<wml>
   <card id="section1" Title="nasa">
      <p>
         <img src="./images/nasa.jpg" alt="earth"/>
         02.01.03 Building Planets in Cyberspace
      </p>
      <p>
         <img src="./images/shuttle.bmp" alt="shuttle"/>
      </p>
      <p>
         <img src="./images/missions.bmp" alt="shuttle"/>
         <img src="./images/multimedia.bmp"
alt="shuttle"/>
         <img src="./images/events.bmp" alt="shuttle"/>
      </p>
   </card>
   <card id="improve" Title="improve life">
      <p>
         <img src="./images/nasa.jpg" alt="earth"/>
         02.01.03 Building Planets in Cyberspace
      </p>
      <p>
         <img src="./images/improvelife.bmp" alt="improve
life"/>
      </p>
      <p>
         <img src="./images/missions.bmp" alt="shuttle"/>
         <img src="./images/multimedia.bmp"
alt="shuttle"/>
         <img src="./images/events.bmp" alt="shuttle"/>
      </p>
   <card>
   ...
</wml>
```

5.2 Structural transformation

Each Web page is a multimedia document that has a layout constructed by many media objects. We start by analyzing the logical structure (automatically generated as a tree) and desired layout and adaptive properties of the given Web page. In the spatial graph grammar, each object is presented by a node. A Web page in XML is a tree structure whose elements can be grouped hierarchically, as shown in Fig. 14 for the given example. To convert the tree to a more structured arrangement suitable for transformation, we need to introduce the concepts of logical nodes and grouping. The tree contains several logical nodes (LNs) such as *Page, Section1, Section2*, etc. As the root, *Page* contains two *Section* nodes. *Section1* contains two *Block* nodes. *Block1* contains *Logo* and *Text*, and *Block2* contains *Theme* and *LN Button*. *LN Button* includes a number of *Link* nodes. *Section2* has a number of child nodes, called *Pictures*. Such hierarchical relationships can be automatically derived from the XML document and used to generate the data structure in Fig. 15.

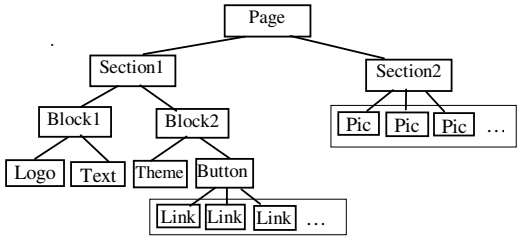


Fig. 14. Tree structure of the Web page

We use an abstract node to head a group that has many objects of a single type. Such a group header has a generic set of attributes applicable to the whole group. Each group mem-

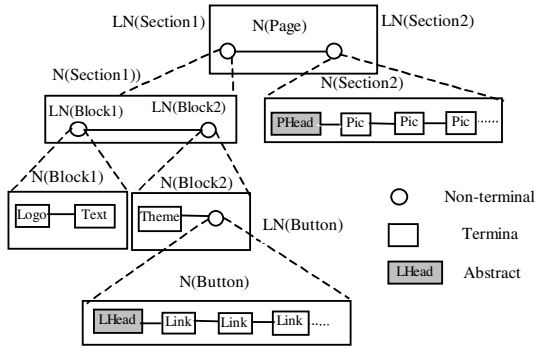


Fig. 15. A hierarchial data structure

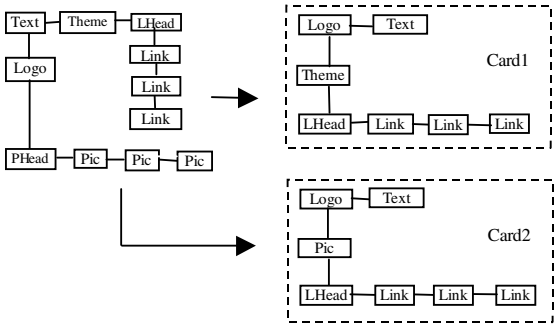


Fig. 16. a Host graph of original structure. **b** Resulting layout structure

ber inherits from its parents attributes such as vertices with spatial information. This arrangement improves the presentation efficiency. Using the concepts of groups and *LNs*, we only need to consider spatial relations of a node with its parent, child, and sibling nodes (i.e., direct relatives). For example, we will consider the relationship between the siblings *LN Block1* and *LN Block2* but not the relationships between the children of *N(Block1)* (i.e., *Logo, Text*) and those of *N(Block2)* (i.e., *Theme*, *N(Button)*). Combining the spatial information from Fig. 12 and the above logical and hierarchical information from Fig. 15, the host graph in Fig. 16a can be automatically generated to be processed by the spatial graph grammar. The application of the SGG generates the new layout structure in Fig. 16b for PDA presentations, as explained in the next subsection.

5.3 Grammatical specification

In order to perform the desired transformation, we define a set of productions as illustrated in Fig. 17. There are two right graphs for some productions. The right graph not enclosed in a dashed box participates in syntactical parsing and, together with the left graph, will be called a *syntax production* or simply S in the following description. The right graph enclosed in a dashed box is used for the layout transformation and, together with the left graph, will be called a *layout production* or simply L. A set of L productions generates a new layout either from an existing layout or from logical relationships between media objects.

Syntax productions

Syntax production <1> (or simply S<1>) expresses the initial state to start with a *Page*. If a parsing eventually reaches the state λ (initial state), it is regarded as successful [39].

S<2> illustrates that a *Page* consists of *Card* and *PHead,* and *Card* is on top of *PHead*. S<3> abstracts a *Card* from *Section1*.

S<4> specifies that *Section1* contains two blocks, and *Block1* is next to *Block2*. The gray vertex in a node means that it is marked and will be reserved during parsing. For example, the vertex labeled P is marked and will stay unchanged after parsing.

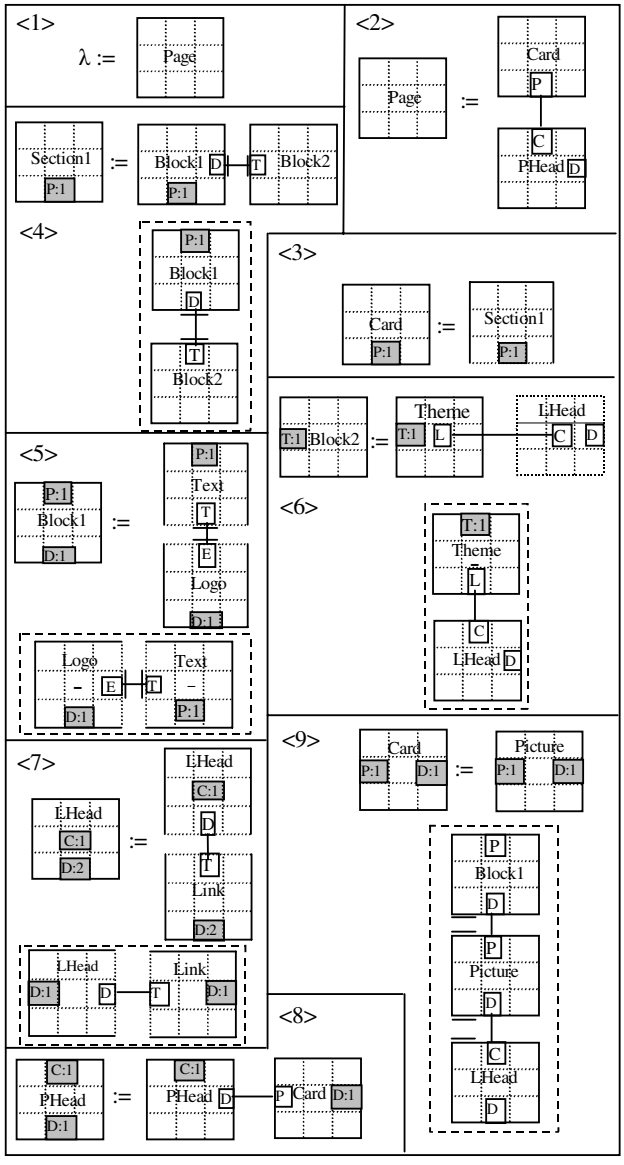


Fig. 17. Productions for the transformation from the presentation in Fig. 12 to the one in Fig. 13

S<5> specifies that *Block1* consists of *Text* and *Logo*, and *Text* is directly on top of *Logo*. The vertices labeled P and D are marked.

S<6> indicates that *Block2* includes *Theme* and *LHead*. *LHead* is a *Group Header* in the *Link* structure and is used to inherit the attributes from its parents. If the *Link* structure contains many members, using *LHead* will significantly improve the efficiency of the graphical presentation. To represent the containing relationship between *Theme* and *LHead*, we use a dotted boundary in *LHead* and connect the two nodes' central grids.

S<7> specifies that the *Link* structure consists of several terminal nodes of *Link* stacked on top of each other.

S<8> and S<9> indicate that *Section2* includes several *Pictures*. In S<8>, *PHead* and *Card* can be reduced to *PHead*. *Card* is an intermediate node and can be abstracted from *Pic-*

ture (*Pic* for short) by using S<9>. We can apply S<9> continuously until no terminal node exists.

The R-application in the SGG is a parsing process that in general consists of: selecting a production from the grammar and applying an R-application of the production to the host graph; the process continues until no productions can be applied. If the host graph is transformed into an initial graph λ, the parsing process is successful and the host graph belongs to the language defined by the graph grammar. We first use S<9> and S<8> to reduce the *Picture* structure to *PHead*. S<7> is used to reduce the *Link* structure to *LHead*. S<6> is then used to reduce *LHead* and *Theme* to *Block2*, and S<5> is used to reduce *Logo* and *Text* to *Block1*. Then we use S<4> to obtain *Section1*. Finally, S<2> reduces *Card* and *PHead* to *Page* and S<1> to λ, and thus the parsing process is successful.

Layout productions

Based on the above syntax productions for parsing the host graph, we add several extended productions enclosed in dotted boxes called layout productions for transforming the presentation in Fig. 12 to the one in Fig. 13. The layout productions are thus an additive set to the syntax productions. Combining these two sets of productions we can generate the desired layout.

Layout production <4> (or simply L<4>) transforms *Block1* and *Block2* from a horizontal relationship to a vertical relationship with *Block1* on top of *Block2*.

L<5> transforms *Text* and *Logo* from a vertically touching relationship to a horizontally touching relationship.

L<6> specifies how to transform two objects from a containing relationship to a vertical relationship. Before the transformation, *Theme* contains *LHead*. After the transformation, *Theme* is on top of *LHead*.

L<7> transforms a sequence of *Links* from vertically touching relationships to horizontally touching relationships; this transformation is repeatedly applied.

In L<9>, when *Picture* with left and right vertices finds a match, it is converted to a *Block1-Picture-LHead* structure, whose three nodes are vertically aligned along the left edges.

We first parse the host graph in Fig. 16a to λ. During parsing, a stack is used to record the sequence of the productions being used. Then, from λ, the original parsing tree is retrieved. At each step, the corresponding layout productions are popped from the stack to perform layout transformations. For example, when *Card* with its southern vertex is matched, S<3> is used to generate *Section1*. Then, we use L<4> to obtain a new layout in which *Block1* and *Block2* hold a vertical relationship. For *Block1*, L<5> is used to derive a horizontal relationship between *Logo* and *Text*. Using L<6>, *Theme* is moved to the top of *LPHead*. L<7> is used to obtain the horizontal *Link* structure. Now we obtain the first PDA page, represented as *Card1* in Fig. 16b. L<9> is used to expand *Card* to the *Block1-Picture-LHead* structure. *Logo* and *Text* are then generated using L<5> and the *Link* structure generated using L<7>. We therefore obtain the second PDA page (marked *Card2* in Fig. 16b). The third and fourth pages, also of the *Card2* structure, are generated in the same fashion. The layout in Fig. 16b can be automatically transformed to the final layout illustrated in Fig. 13.

6 Example 2: Adapting presentation styles

This section provides another example, this one on the adaptation of presentation styles.

6.1 A presentation style

As an example of style adaptation, consider an art museum that organizes its multimedia documents in a predetermined logical structure, as shown in Fig. 18. In one exhibition season, the museum would like to display the documents on the Web as displayed in Fig. 19 with the following presentation organization. The page consists of a menu bar of various hyperlinks on the left side for the whole museum, hyperlinks to all the curatorial departments at the top, and the collection highlights occupying the main page area. The highlights of each museum department consist of a number of well-known artworks (i.e., pictures). Assuming the pictures need to be displayed with 3 in each row, six pictures of the selected "Painting and Sculpture Highlights" are displayed in two rows by three columns in the main area.

6.2 Grammatical specification

The logical structure of Fig. 18 is regarded as a host graph, which is used to dictate the presentation layout according to

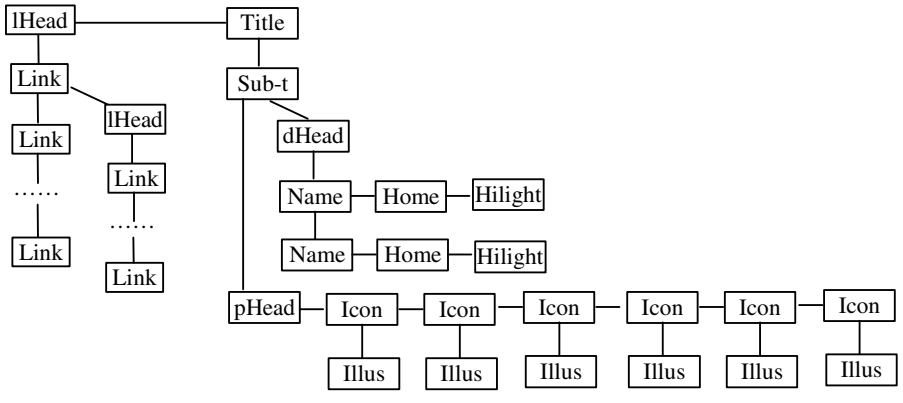


Fig. 18. Host graph of the Museum multimedia document

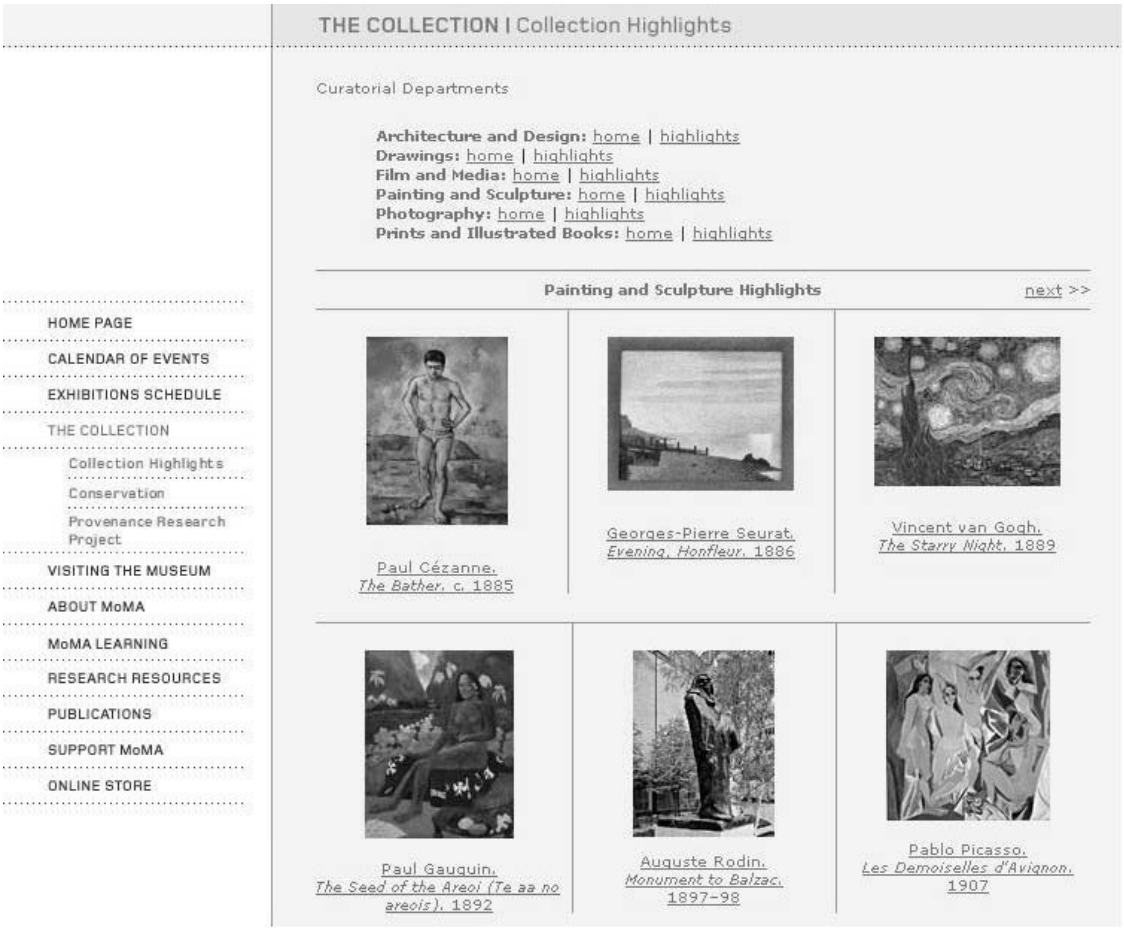


Fig. 19. A museum multimedia presentation

a grammar specification. The complete set of graph grammar production rules that meet the requirements of the presentation style of Fig. 19 is listed in Fig. 20. The document consists mainly of two composite objects, *Sections* and *Content*, which participate in production <2> in Fig. 20.

The *Sections* object consists of a number of hyperlinks that enable the user to navigate other museum documents from the same page. The hyperlinks are organized hierarchically. A link at level i may include several links at level $(i + 1)$. The links at the same level are aligned to the left, and level $(i + 1)$ links are indented from its level i links. Production <3> abstracts a terminal node called *Link* to a nonterminal node *Section*. Production <4> dictates how to reduce two links (represented by two *Sections*) while establishing their spatial relations – vertically aligned and touching each other. Production <5> is for reducing the last *Section* node, which is characterized by an unmarked N vertex in *Section*. Production <6> demonstrates how to attach hyperlinks (upgraded from *Link* to *Section*) at level $(i + 1)$ to a hyperlink at level i with right indentation (realized by a partially touched relationship as shown in the right graph of the production).

The right side of *Context* consists of three objects, *Title*, *Depts*, and *Pics*, as specified in production <7>. The *Title* object presents the title of the document and is placed above the other two composite objects. The *Depts* object, representing a list of departments in the museum, consists of multiple *Dept* objects to be aligned to the left and vertically touching each other. As shown in production <9>, each *Dept* object consists of three primitive objects, *Name*, *Home*, and *Hilight*, which are to be touching each other and aligned horizontally. Production <8> reduces two *Dept* objects into one, and production <10> provides the syntax of a *Depts* object.

A *Pic* object represents a picture displayed in the main area and has *Icon* and *Illus* objects as specified in Production <11>. The *Icon* object is placed above the *Illus* object, and both are center aligned. Representing the highlights of a department to be displayed in the main area, a *Pics* object is abstracted from a *pHead* object in production <12> and consists of multiple *Pic* objects to be laid out according to productions <13> and <14>. By applying production <13> repeatedly to reduce two *Pic* objects into one *Pic* object, a sequence of *Pic* objects is generated.

Since three *Pic* objects are required to be displayed in each row, we need to use *action* codes to specify the constraint. An action code associated with a production is a Java exception-handling method [39] used to specify the semantics of the production and to provide additional control information to the parser. We introduce a global variable, called *NumOfCol*, to record the current number of the *Pic* objects in the current row. Initially, *NumOfCol* is set to 1 in production <12>. Every successful application of production <13> increases *NumOfCol* by 1 until its value reaches 3. Production <14> is applied

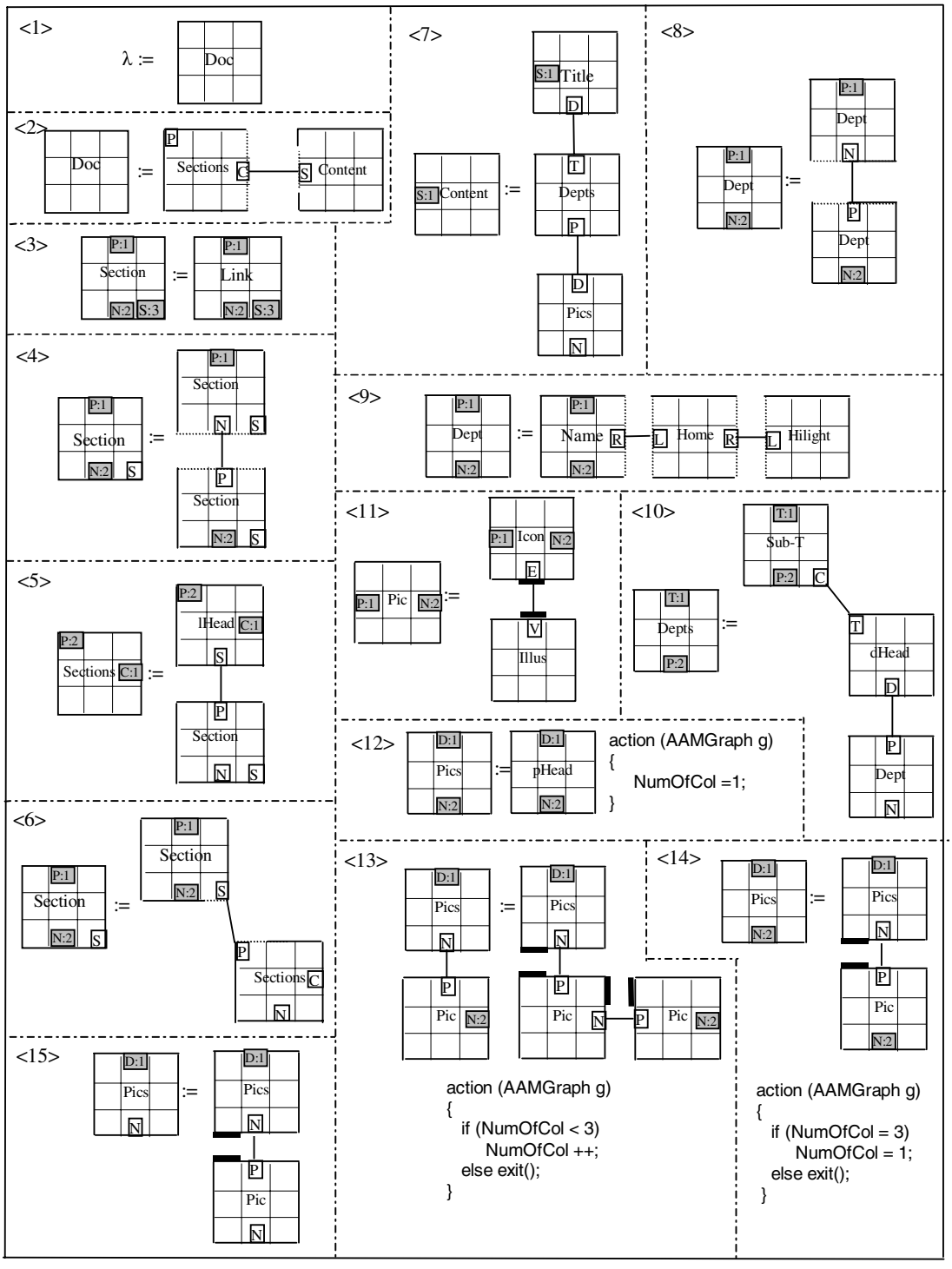


Fig. 20. Graph grammar definition of both the document structure in Fig. 18 and presentation in Fig. 19

only when *NumOfCol* is equal to 3, indicating that there are already three *Pic* objects in a row. However, the last row may contain less than three *Pic* objects. Production <15> handles such a special case. Apart from the action code (production <15> has no action code), the only difference between productions <14> and <15> is that the N vertex is marked in <14> and unmarked in <15>.

6.3 Adapting to an alternative style

Assume in another exhibition season the museum homepage will be presented in an alternative layout, as illustrated in Fig. 21, where the main area has a different arrangement. It would be a time-consuming and error-prone process to manually adjust the layout of each page for a large number of similar pages. Fortunately, the spatial graph grammar provides an

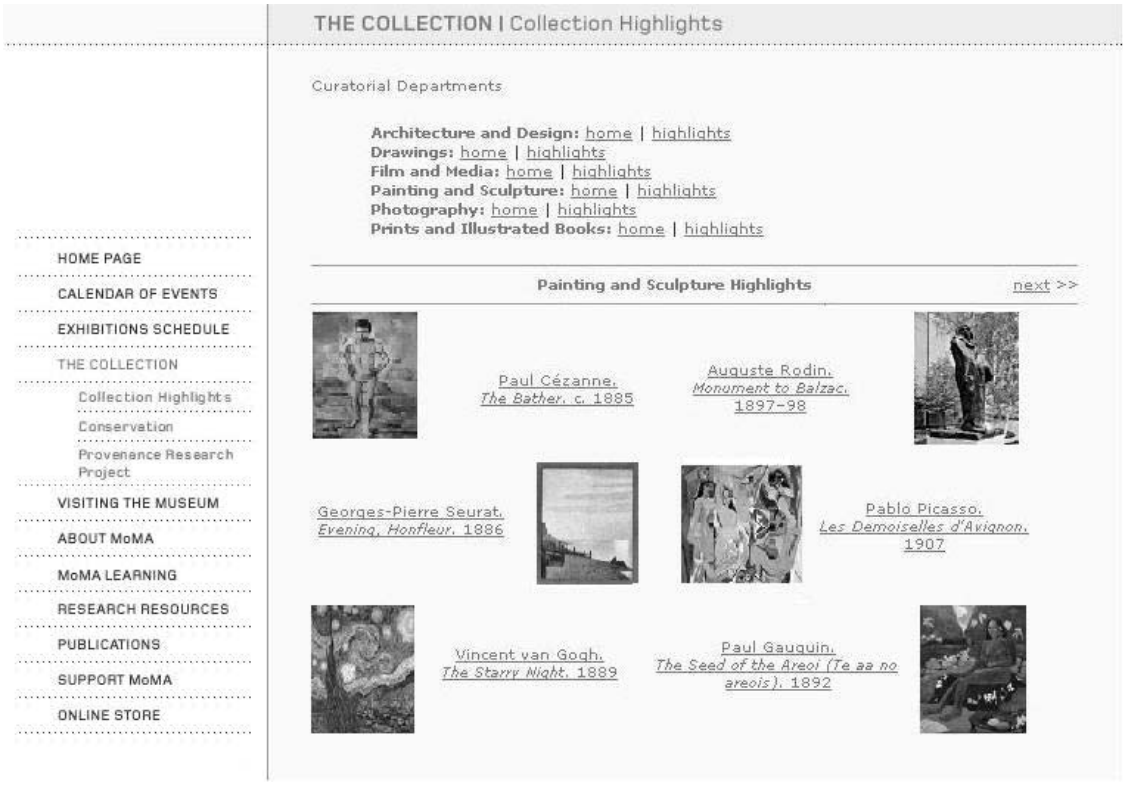


Fig. 21. An alternative presentation style

adaptive approach to document presentation since the system can select an appropriate set of productions and automatically generate a desired layout when the context is changed. To support the above alternative presentation style, what is needed is simply a subset of new productions that will replace productions <11> to <15>, as listed in Fig. 22. By applying the alternative productions, the document not only displays two *Pic* objects in each row, but also interleaves the *Icon* and *Illus* objects. Other presentation styles could also be easily adapted by modifying the relevant part of the grammar. A typical example is when the width of the viewing device is too narrow to fit three *Pic* objects, so that the layout can be adjusted to two *Pic* objects or even one in each row.

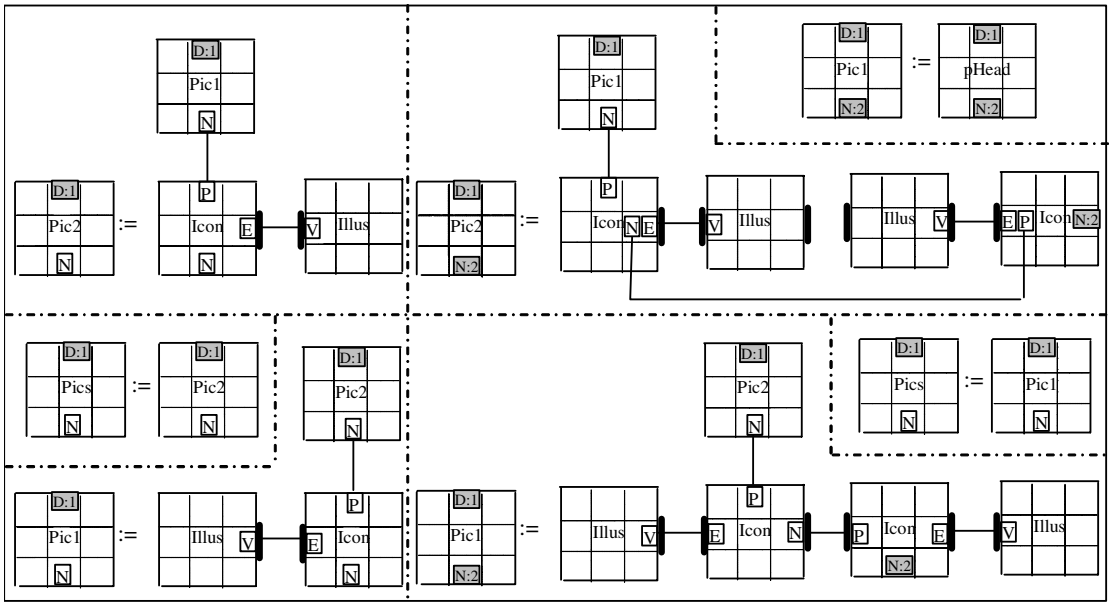


Fig. 22. Revised subset of productions (replacing productions 11–14 in Fig. 20) for generating the alternative presentation style of Fig. 21

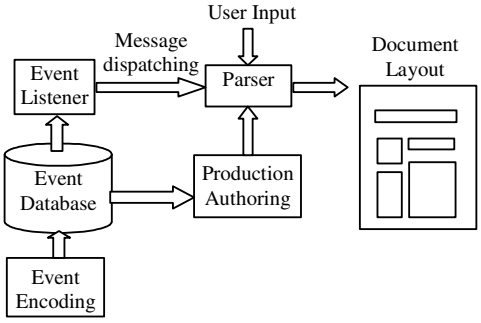


Fig. 23. System architecture

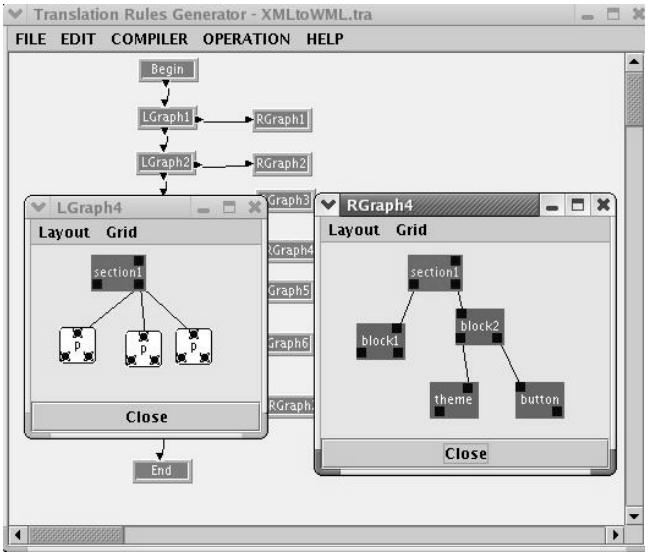


Fig. 24. Specifying productions using Rule Generator

7 System architecture and implementation status

At the system level, the grammatical approach described above is realized by four modules, as shown in Fig. 23: event encoding, event listener, production authoring, and parser. The event-encoding module lets the user describe the events to which the grammar should be sensitive. The event listener dynamically monitors the system to see if any changes have occurred due to content update or user interactions. When the event listener retrieves user inputs and messages from the event database, the parser performs the corresponding graph transformation according to the predefined graph grammar.

The production-authoring module provides a tool to define a graph grammar according to the desired document layout and its dynamic behavior. The grammar dictates how to construct a multimedia document layout through various types of media objects, as described in the previous sections. A production specifies not only how to construct composite objects but also what the constructs look like and how they adapt to dynamic changes.

The parser validates the structure of a host graph and automatically generates a parsing tree that reflects the hierarchical structure. Also, the layout is adjusted according to the spatial specifications, which are integrated with the structural specifications. For example, when a user modifies the font sizes or device characteristics, a message is dispatched to the parser, and a conditional (event-driven) production may be triggered to perform a graph transformation. The positions and styles of objects are adjusted according to the spatial specifications in the grammar. During the process of graph transformation, some objects may collectively construct a composite object, which is treated as one entity whose position change in the later layout process will not affect the spatial relationships among its internal objects.

When defining grammar productions for graph layout where edges represent only geometric relations, we allow only one relation between any pair of nodes. Such relationships can be efficiently handled by the original RGG formalisms. Our graph grammar formalism with spatial specification mechanisms is sufficiently expressive in specifying multiple connectivity and complex presentation structures.

The whole system except the layout component has so far been implemented based on a visual language generation (VPL) framework. A VPL framework is essentially a metatool for the automatic generation of visual specification tools [38] with which different multimedia authoring and presentation languages can be automatically generated according to varied requirements specified through spatial graph grammars. For the example in Sect. 5, we can specify the original document structure through syntax productions as shown in a snapshot of Fig. 24 (i.e., production-authoring module in Fig. 23; events are not used in this example). The production without spatial specification is slightly different from the full version of Fig. 17. The metatool then automatically generates the language environment with a graphical editor and a parser. A user can then use the graphical editor to draw an application document structure and provide desired texts and attributes, as illustrated in the snapshot in Fig. 25. When the compiler is triggered from the menu on top of the editor, the resulting document structure is visualized (Fig. 26) and the WML document listed in Sect. 5.1 is generated.

8 Related work

There have been a number of systems and approaches for the authoring and presentation of multimedia systems [24]. The Synchronized Multimedia Integration Language (SMIL) [5,35] allows control over which media elements and where and when those media elements are to appear in a multimedia presentation. Though the SMIL is flexible enough to support multiple alternative layouts, there are several fundamental differences between the SMIL and our graph-grammar-based approach. First, rather than providing absolute layout positioning as in the SMIL, a graph grammar defines the desirable layout adaptive to the existing layout or user intention. Second, when a media element is deleted or inserted, satisfying predefined structural constraints, an updated representation can be automatically generated in the grammatical approach through parsing, but this is not possible with the SMIL. Third, the position of a media element in the SMIL is defined relative to the size of the element's parent geometry. The RGG can not only define a representation in the same way through attributes and action codes, but it can also specify the position of one element relative to another through graphical notations.

Cascading Style Sheets (CSS) [36] define how to display Web documents, including specification of fonts, background,

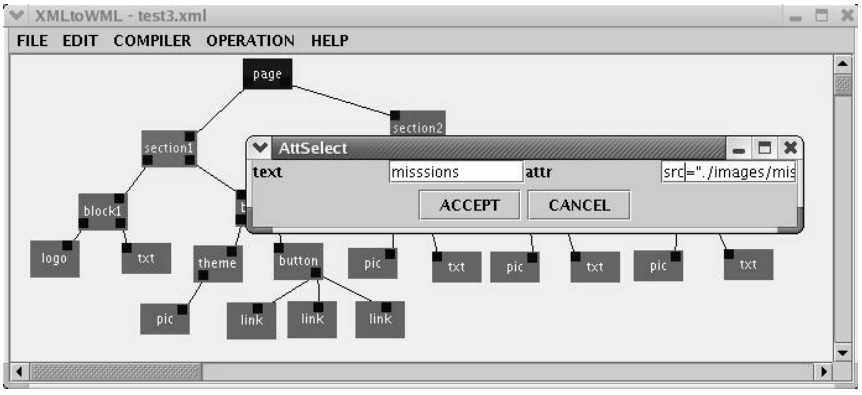


Fig. 25. Specifying original document structure using the generated graphical editor

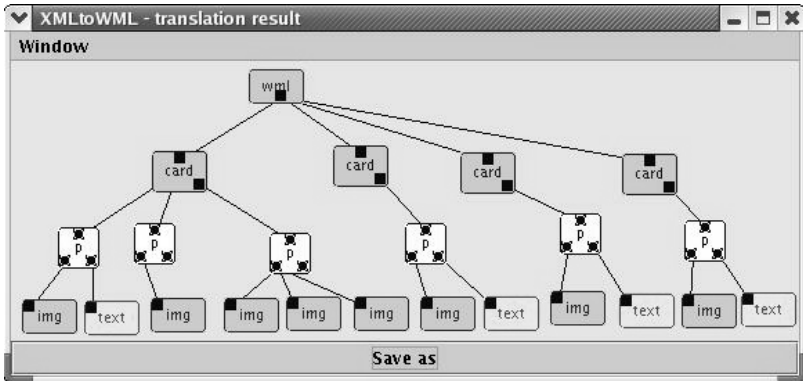


Fig. 26. Automatically translated document structure

foreground, and so on. They allow both the author and reader to provide rules that specify various attributes of a Web document. Multiple style definitions will cascade into one according to some conflict-resolving rules. The layout mechanism of both the CSS and SMIL works on a predefined specification. Only through a transformation language, such as XSLT [37], may CSS and SMIL allow the layout mechanism to work conditionally on a previous layout or a spatial property. The graph grammar approach, however, allows the new layout to be generated based on the previous layout as well as on a specification.

Among knowledge-based approaches, Comet [10] and WIP [2] employ some forms of rule-based mechanisms to represent the graphical design knowledge. The rules control the search of all possible solutions and determine an appropriate solution. One of the most difficult issues in these systems is how to specify the control mechanism, which could be more easily addressed by a parsing algorithm for graph grammars.

Vazirgiannis et al. propose a spatiotemporal composition model [30] and indexing schemes for efficient querying in such a spatiotemporal coordinate system [31]. The model translates spatial and temporal relationships among multimedia objects into minimal and uniform expressions and allows authors to specify an object's spatial features either as absolute coordinates or in relation to other objects. Algorithms and tools have been developed to transform relative data into absolute coordinates and to verify the integrity of spatial and temporal relationships. The model does not address its adaptability to the changing space and layout requirements. Based on the nested context model (NCM) [7], HyperProp [29] emphasizes the importance of document logical structuring. It supports event-based spatial synchronization and behavior specification but offers no explicit specification of document layout and spatial adaptation. Temporal aspects are also investigated by Guan et al. in their model of Distributed Object Composition Petri Net (DOCPN) [12] that facilitates the synchronization of multimedia presentations in a distributed computing environment. The present paper does not address the temporal issue but rather leaves it as our future work.

In a dynamic interface, the attributes of elements are defined in terms of other elements and attributes of the viewing environment: information links indicate a (semantic) connection between two pieces of information that can belong to different information domains, and an information view is a collection of correlated objects displayed together to help the user perform some activities on the objects [3]. Interactivity allows the display to be dynamically adapted to user requirements. Borning et al. present a system architecture in which both the author and the viewer can impose page layout constraints [4]. The final appearance of a Web page is thus the result of negotiation between the author and the viewer. Marriott et al. [18] extend Scalable Vector Graphics (SVG) with constraint-based specification. Such an extension supports client-side adaptation of documents to different viewing conditions. These approaches do not offer visual specifications, and their layout solutions rely on constraint solvers.

In dynamic authoring, "authoring" refers to creating the content for any kind of presentation or document [21]. Dynamic authoring advocates that capture-based systems should support flexible hypertext structures generated by linking through interactive operations [23]. Some user interface toolkits use the approach of recognition and mediation by constructing a library of reusable error correction tools, that can provide

structured support for resolving ambiguity at the input event level [17].

Our work was influenced by that of Weitzman and Wittenburg [32], who applied a graph grammar formalism – Relation Grammar [33] – to the automatic presentation of multimedia documents. The grammar governs the structure of the document. One or more parsing trees, each of which represents an independent presentation, are derived through a parser. Then, a syntax-directed translation is made on the tree. The final layout is created by a constraint solver following the translation. In this approach, relational grammar functions as a mapping from a representation of one style of multimedia documents to the forms that specify how to realize the media objects. Inspired by the work of Weitzman and Wittenburg, Cruz and Lucas developed a visual querying and presentation system called Delannay^{MM} [8], but grammars are not used in this system.

Another area of research that has influenced our work is graph drawing [9]. Six et al. [28] proposed postprocessing techniques (after some major graph layout process), called *refinement*, for effective graph drawing. The techniques can significantly improve the quality of orthogonal drawings by reducing a graph's area, bends, crossings, and total edge length [9]. In a graphical layout, maintaining a consistent view by automatically beautifying the display is desirable [19]. We apply a grammatical approach rather than an algorithmic approach to the graph layout problems addressed by Six et al. [28]. In [42], Zhang et al. present an approach to combining the RGG formalism with constraint rules to support the automatic layout of orthogonal graphs. The work reported in this paper represents a major extension that introduces spatial representation and specification.

Research has been done on graph grammar support for Web information transformations. To support the automatic layout of flowcharts, Zhang et al. [40] recently presented a visual approach to XML document design and transformation that uses RGGs [39] to define the XML syntax and to specify the transformation between different XML formats.

9 Conclusions and future work

The grammatical approach is promising in that it provides a powerful mechanism for representing layout structures graphically and for performing online validation and adaptation through an automatically generated parser. This paper has presented the concept of applying graph grammars to the transformation of multimedia presentations to achieve automatic adaptation to the change of media content, different layout requirements, and user interactions. Such transformations usually involve location change, differential scaling, and semantic zooming. To graphically represent these three types of changes, we have proposed the notation of grid nodes, and five categories of spatial relationships. We use a context-sensitive graph grammar formalism to explicitly describe the syntax of Web application layouts and transformation methods. The parsing algorithm of our spatial graph grammars has polynomial-time complexity in most cases [16], and the parser performs an automatic validation on the layout structure.

The graph transformation tool can be considered an authoring language generator, i.e., a metatool, that can generate any authoring tool environment or regenerate a modified tool whenever needed. A multimedia author without any knowledge of graph grammars or design rules will be able to use the generated authoring tool to make adaptive presentations by drawing graphical structures. Syntax check and design validation are then automatically performed by the authoring tool. A graph layout can be transformed according to the defined grammar or run-time events such as a user interaction.

We are currently implementing the layout component and will apply it to a rendering tool. More work needs to be done to enrich the grammar formalism and enhance the event-handling capability. An important issue is the scalability of graphical views. One solution is to introduce hierarchical views [22]. Another is to apply the grammar replacement process to support graph expansion and shrinking, which would, however, limit one to context-free graph grammars. Writing production rules and their action codes to handle adaptation is not an easy task, even for a design expert, since it requires a good command of the grammar formalism. It has been the authors' goal to partially automate the production authoring tool to create part of the rules when layout requirements are spelled out. This and support for scalability will be our immediate future work.

Designing dynamic mobile interfaces to support interactive communication with mobile devices is a research and design problem. In dynamic capture, access, and authoring of multimedia presentations, the attributes of some media elements may be defined in terms of those of other media elements or in relation to the attributes of the viewing environment. We will investigate attributed grammars that adapt to multiple simultaneous changes.

Another future project is to add the time dimension to the design of multimedia presentations. Temporal specifications determine the sequence of presentation. Allen presented some common temporal relations such as *during*, *before*, and *meet* relations [1], which are potentially adaptable to the grammar as conditional attributes. Extensive research has been devoted to the temporal aspects of multimedia authoring and presentations. We plan to investigate the combined use of temporal and spatial specifications and explore the full power of the grammatical approach.

Acknowledgements. The authors would like to thank anonymous reviewers for their constructive comments and suggestions, which have helped us to significantly improve the paper. The work is partially supported by the National Science Foundation under grant number IIS-0218738.

References

1. Allen JF (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843
2. Andre E, Finkler W, Graf W, Rist T, Schauder A, Wahlster W (1993) WIP: The automatic synthesis of multimodal presentations. In: MayBury M (ed) Intelligent multimedia interfaces. AAAI Press/MIT Press, Cambridge, MA, pp 75–93
3. Bjork S, Redstrom J, Ljungstrand P, Holmquist LE (2000) PowerView – using information links and information views to navigate and visualize information on small displays. In: Proc. HUC'2000, pp 46–62

4. Borning A, Lin RK, Marriott K (2000) Constraint-based document layout for the Web. Multimedia Syst 8:177–189
5. Bulterman DCA, Rutledge L (2004) SMIL 2.0 – interactive multimedia for Web and mobile devices. Springer, Berlin Heidelberg New York
6. Burnett MM (2004) Visual language research bibliography. http://www.cs.orst.edu/∼burnett/vpl.html
7. Casanova M, Tucherman L, Lima M, Rodriguez N Soares L (1991) The nested context model for hyperdocuments. In: Proc. Hypertext, San Antonio, TX, pp 193–201
8. Cruz IF, Lucas WT (1997) A visual approach to multimedia querying and presentation. In: Proc. ACM Multimedia'97, Seattle, November 1997, pp 8–14
9. Di Battista G, Eades P, Tamassia R, Tollis IG (1999) Graph drawing: algorithms for the visualization of graphs. Prentice Hall, Englewood Cliffs, NJ
10. Feiner S, McKeown K (1993) Automating the generation of coordinated multimedia explanations. In: MayBury M (ed) Intelligent multimedia interfaces. AAAI Press/MIT Press, Cambridge, MA, pp 117–138
11. Golin EJ (1991) A method for the specification and parsing of visual languages. Ph.D. thesis, Brown University, Providence, RI, May 1991
12. Guan SU, Yu H-Y, Yang J-S (1998) A prioritized Petri Net model and its application in distributed multimedia systems. IEEE Trans Comput 47(4):477–481
13. Herman I, Melancon G, Marshall MS (2000) Graph visualization and navigation in information visualization. IEEE Trans Visual Comput Graph 6(1):24–43
14. Ishizaki S (2003) Improvisational design – continuous responsive design communication. MIT Press, Cambridge, MA
15. Kaul M (1982) Parsing of graphs in linear time. In: Proc. 2nd international workshop on graph grammars and their application to computer science. Lecture notes in computer science, vol 153. Springer, Berlin Heidelberg New York, pp 206–218
16. Kong J, Zhang K (2003) Spatial graph grammars for graphical user interfaces. Technical report, UTDCS-47-03, Computer Science Department, University of Texas at Dallas, October 2003
17. Mankoff J, Abowd GD, Hudson SE (2000) OOPS: A toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces. Comput Graph 24(6):819–834
18. Marriott K, Meyer B, Tardif L (2002) Fast and efficient client-side adaptability for SVG. In: Proc. WWW 2002, Hawaii, 7–11 May 2002, pp 496–507
19. Minas M, Viehstaedt G (1993) Specification of diagram editors providing layout adjustment with minimal change. In: Proc. IEEE symposium on visual languages, pp 324–329
20. Muchaluat DC, Rodrigues RF, Soares LFG (1998) WWW fisheye-view graphical browser. In: Proc. IEEE Multimedia Modeling
21. Myers BA (1998) Authoring interactive behaviors for multimedia. In: Proc. 9th NEC research symposium, Nara, Japan, August–September 1998
22. Pietriga E, Vion-Dury J-Y, Quint V (2001) VXT: A visual approach to XML transformations. ACM symposium on document engineering, Atlanta, GA, 9–10 November 2001, pp 1–10
23. Pimental M, Abowd G, Ishiguro Y (2000) Linking by interacting: a paradigm for authoring hypertext and hypermedia. In: Proc. Hypertext 2000, Austin, TX, pp. 39–48
24. Prabhakaran B (2000) Multimedia authoring and presentation techniques, guest editor's introduction. Multimedia Syst 8:157
25. Rozenberg G, Welzl E (1986) Boundary NLC graph grammars – basic definitions, normal forms, and complexity. Inf Control 69:136–167
26. Rozenberg G (ed) (1997) Handbook on graph grammars and computing by graph transformation: foundations, vol 1. World Scientific, Singapore
27. Sarkar M, Brown MH (1994) Graphical fisheye views. Commun ACM 37(12):73–84
28. Six JM, Kakoulis KG, Tollis IG (2000) Techniques for the refinement of orthogonal graph drawings. J Graph Algor Appl 4(3):75–103
29. Soares LFG, Rodrigues RF, Saade DCM (2000) Modeling, authoring and formatting hypermedia documents in the HyperProp system. Multimedia Syst 8:118–134
30. Vazirgiannis M, Kostalas I, Sellis T (1999) Specifying and authoring multimedia scenarios. IEEE Multimedia, July–September 1999, pp 24–37
31. Vazirgiannis M, Theodoridis Y, Sellis T (1998) Spatio-temporal composition and indexing for large multimedia applications. Multimedia Syst 6:284–298
32. Weitzman L, Wittenburg K (1998) Grammar-based articulation for multimedia document design. In: Maybury MT, Wahlster W (eds) Readings in intelligent user interfaces. Morgan Kauffmann, San Francisco, CA, pp 310–327
33. Wittenburg K, Weitzman L (1996) Relational Grammars: Theory and practice in a visual language interface for process modeling. In: Proc. AVI'96, Gubbio, Italy, 27–29 May 1996
34. Wills LM (1992) Automated program recognition by graph parsing. Ph.D. thesis, MIT AI Lab, Cambridge, MA
35. W3C (2001) Synchronized multimedia integration language (SMIL 2.0). http://www.w3.org/TR/2001/REC-smil20-20010807/, August 2001
36. W3C (2004) Cascading Style Sheets (CSS). http://www.w3.org/Style/CSS/
37. W3C (1999) XSL Transformation (XSLT). http://www.w3.org/TR/xslt
38. Zhang K, Zhang DQ, Cao J (2001) Design, construction and application of a generic visual language generation environment. IEEE Trans Softw Eng 27(4):289–307
39. Zhang DQ, Zhang K, Cao J (2001) A context-sensitive graph grammar formalism for the specification of visual languages. Comput J 44(3):186–200
40. Zhang K, Zhang DQ, Deng Y (2001) Graphical transformation of multimedia XML documents. Ann Softw Eng 12:119–137
41. Zhang KB, Zhang K (1999) An incremental approach to graph layout based on grid drawing. In: Proc. 3rd workshop on software visualization (SoftVis'99), University of Technology, Sydney, 3–4 December 1999
42. Zhang KB, Zhang K, Orgun MA (2002) Grammar-based layout for a visual programming language generation system. In: Proc. 2nd international conference on the theory and application of diagrams (Diagrams'02), Atlanta, GA, 18–20 April 2002. Lecture notes in computer science, vol 2317. Springer, Berlin Heidelberg New York, pp 106–108