IMPROVISE: Automated Generation of Animated Graphics for Coordinated Multimedia Presentations

Michelle X. Zhou[1] and Steven K. Feiner[2]

[1] IBM T.J. Watson Research Center, Hawthorne, NY, USA
mzhou@watson.ibm.com
[2] Department of Computer Science
Columbia University, New York
feiner@cs.columbia.edu

Abstract. In this chapter, we describe a graphics generation system, IMPROVISE (Illustrative Metaphor PROduction in VISual Environments), focusing on how it can be used to design coordinated multimedia presentations. IMPROVISE is a knowledge-based system that can automatically create sequences of animated graphical illustrations to convey a wide variety of data. Our emphasis here is on describing how four important features of IMPROVISE facilitate coordinated multimedia presentation design. These four features are: a knowledge-rich representation of input data, a fine-grained temporal model of visual techniques, an action-based inference engine, and a portable visual realizer.

1 Introduction

An automated visual presentation system should be able to communicate a set of data entities (e.g., database tables) to a specific user in a particular context (e.g., a user with a known set of skills, using a particular display device) in a way that fulfils a presentation intent (e.g., to *summarize* certain aspects of the data). To automatically create proper visual presentations, automated visual presentation systems are usually built using a knowledge-based approach (e.g., Seligmann and Feiner 1991; Roth and Mattis 1991). In other words, these systems have knowledge about the underlying data, users and their intents, and visual design. They are also powered by an inference engine to infer the proper visual design on the fly. Thus, all knowledge-based visual presentation systems must contain three main components: a knowledge base that stores various information about data to be presented and visual design, an inference engine that performs reasoning, and a visual realizer that transforms design specifications into human-perceivable pictures.

Following this view, we have developed IMPROVISE, a knowledge-based system that can automatically create visual presentations for a wide variety of data. IMPROVISE can be used stand-alone to create purely visual presentations or cooperate with other media generators (e.g., a spoken language generator) to create

H. Bunt and R.-J. Beun (Eds.): CMC'98, LNAI 2155, pp. 43–63, 2001.

coordinated multimedia presentations. Unlike the work on automated generation of individual visual presentations (e.g., Mackinlay 1986; Roth and Mattis 1991) or sequences of discrete visual presentations (e.g., Seligmann and Feiner 1991; Andre and Rist 1993), IMPROVISE focuses on automatically creating coherent visual discourse. We use the term *visual discourse* to refer to an animated visual narrative expressed in the form of sequences of temporally-ordered, animated visual actions (Zhou and Feiner 1998a). For example, a narrative generated by IMPROVISE may start by displaying a set of objects, and then animate the highlighting of one object, followed by the generation of a cutaway view to reveal its internal structure.

To create a coherent, animated visual presentation and cooperate with other media generators, IMPROVISE has a well-formulated knowledge base, a sophisticated and efficient inference engine, and a portable visual realizer. IMPROVISE uses a knowledge-rich representation to express all input data, and has a fine-grained model for describing its visual design knowledge. These two features not only provide a foundation for automated graphics generation, but also make possible cooperative multimedia presentation design.

IMPROVISE uses a top-town, hierarchical-decomposition, partial-order planning-based inference engine to compose sequences of animated visual actions (Zhou and Feiner 1998b). As we describe later, the flexibility of this action-based approach greatly facilitates cooperative multimedia presentation design. Once the inference engine creates the design specifications, IMPROVISE's visual realizer interprets the meaning of the design while obeying the constraints specified in the design (e.g., temporal duration). Using a platform-independent multimedia authoring language and a precise action-execution scheduler, the visual realizer can render the design specifications on various platforms and faithfully maintain the specified temporal constraints among different actions.

Since we have described various features of IMPROVISE elsewhere (Zhou and Feiner 1998a, 1998b), here we focus on illustrating how four features of IMPROVISE facilitate coordinated multimedia presentation design: the representation of input data, the temporal model of visual techniques, the action-based planning engine, and the portable visualizer. After a brief discussion of related work, we give an overview of IMPROVISE's architecture and analyze two examples that are created by IMPROVISE. We then explain each of the four features in the context of designing coordinated multimedia presentations. Finally, we present our conclusions and indicate some future research directions.

2 Related Work

Unlike other automated graphics generation systems such as IBIS (Seligmann and Feiner 1991) and SAGE (Roth and Mattis 1991), IMPROVISE can generate sequences of coherent, animated graphical actions with fine-grained temporal constraints. In addition, IMPROVISE is capable of adjusting its constraints so that it can cooperate with other media components to produce a coordinated presentation.

IMPROVISE has been combined with a language generation system to produce an automated multimedia presentation system, called MAGIC (Multimedia Abstract Generation for Intensive Care) (Dalal et al. 1996). MAGIC can automatically generate multimedia briefings using two temporal media: speech and animated graphics. Compared to other multimedia presentation systems, MAGIC differs in two aspects. First, earlier multimedia systems (e.g., Feiner and McKeown 1991; Wahlster et al. 1993; Mittal et al. 1998) produce discrete multimedia presentations that do not involve temporal media, such as continuous speech or animated graphics. In these systems, media coordination tasks are relatively straightforward; for example, coordinating each picture with a text caption. In contrast, MAGIC deals with temporal media and requires that the order and duration of actions that occur in different temporal media be coordinated (Dalal et al. 1996).

Second, more recent multimedia presentation systems (e.g., Towns, Callaway, and Lester 1998; Andre, Rist, and Müller 1998; Noma, Zhao, and Badler 2000) that employ temporal media, including video clips, animated graphics, and speech, use preconstructed graphics objects, speech, or scripts to compose their presentations. In contrast, MAGIC's media-specific actions, such as speech and animation, are all generated dynamically with temporal constraints, and the coordination tasks are implicit. In other words, the decisions about coordinating the content and form of all media must be made at run-time. In addition, our media actions are specified at a more detailed level of representation, and require much finer-grained coordination. For example, temporal durations specified for words and phrases in speech must be coordinated with the durations of corresponding actions in graphics, such as displaying and highlighting objects.

3 System Architecture

IMPROVISE's cycle of presentation design, shown in Fig. 1, starts with a set of *presentation intents*, which usually describe domain-specific communicative goals. The *task analyzer* is responsible for formulating and translating presentation intents (e.g., to examine a network link structure) into corresponding visual tasks (e.g., Focus on the link and Expose its internals). To accomplish these visual tasks, the *visual presentation planner* starts the design process. In IMPROVISE, design is an interleaved process involving two submodules: the visual content planner and the visual designer. These two submodules work cooperatively, using the *inference engine* to access the *knowledge base* and infer a visual design. The *visual content planner* selects and organizes the content that needs to be presented, while the *visual designer* makes decisions about what visual cues to use and how to combine various visual elements into a coherent whole. Once the design is finished, it is written out in an intermediate presentation authoring language, and eventually converted to the target graphics language to be realized. IMPROVISE also has a simple *interaction handler* that processes *user events* and formulates new communicative goals (presentation intents), and these new goals are passed to the task analyzer where a new design cycle begins.

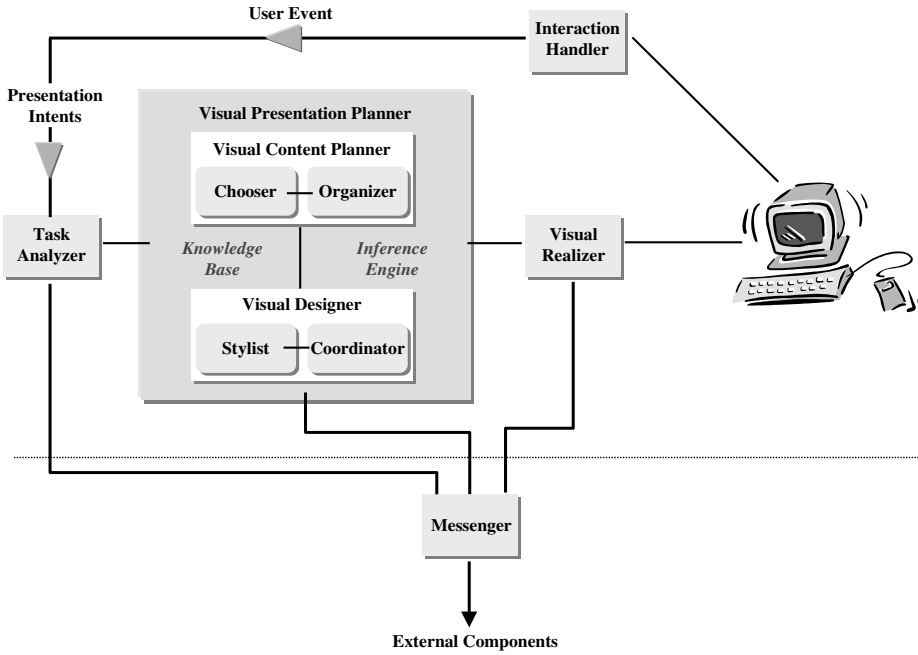


Fig. 1. IMPROVISE System Architecture.

When embedded in a multimedia system, such as MAGIC, IMPROVISE has an additional communication module, the *messenger*, which is responsible for exchanging information between other internal components of IMPROVISE and external components (e.g., a *media coordinator*, not shown here). IMPROVISE uses a task-based communication strategy for its internal components to exchange messages with external components, and to let the messenger deal with all low-level format transformation (e.g., converting the data to a format that can be understood by another system) and communication issues (e.g., establishing socket connections).

Currently, IMPROVISE supports four types of communication tasks: get-goals, get-action-orders, get-action-durations, and get-start-time. Using the first task get-goals, IMPROVISE's task analyzer can obtain the annotated communicative goals from MAGIC's media allocator. IMPROVISE's visual presentation planner employs the other three tasks to negotiate with the media coordinator for its visual actions' temporal orders, durations, and starting times.

4 Examples

Here we use two very different examples to demonstrate how IMPROVISE creates animated visual narratives in different situations. The first example shows how IMPROVISE generates a visual narrative from scratch to present a hospital

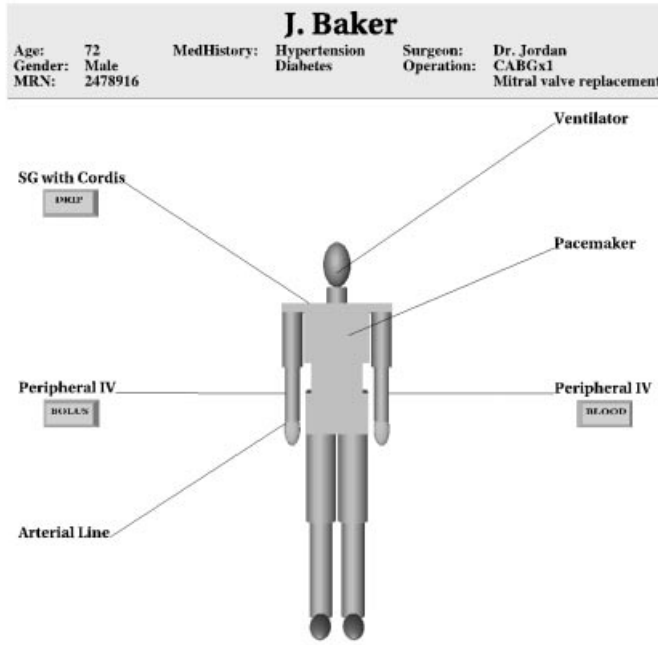
patient's information to a nurse after the patient's coronary artery bypass graft (CABG) operation. In this case, the generated visual narrative is combined with generated spoken sentences to produce a coordinated multimedia summary (Fig. 2). In the second example, IMPROVISE modifies an existing visual presentation of a computer network (Fig. 4a), using a set of animated visual actions to gradually reveal the internal structure of a user-selected network link (Fig. 4b-d).

4.1 Presenting Information to a Nurse

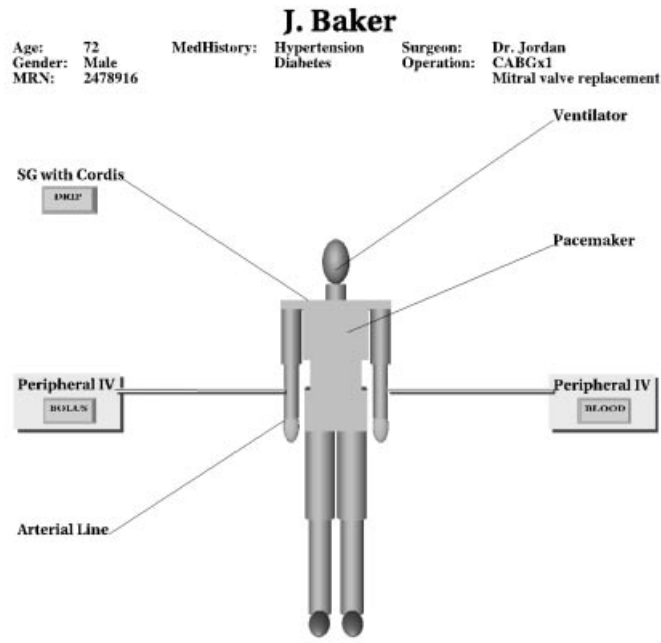
In this task, IMPROVISE must accomplish two goals: creating an overview of patient information, and elaborating the patient information details based on that overview. To achieve the first goal, IMPROVISE plans to construct a *structure diagram* that organizes various information (e.g., IV lines) around a core component (the patient's body). This decision is made based on the fact that in this domain, the nurses with whom we worked prefer to see this information arranged relative to the patient's body. In a top-down design manner, IMPROVISE first creates an 'empty' structure diagram. This empty diagram is then refined through its individual components by recursively partitioning and encoding the patient information into different groups. For example, the patient's demographics, including name, age, and gender, are encoded as the *heading* of the diagram (the highlighted block at the top of Fig. 2a); a representation of the patient's physical body serves as the *core*, and the rest of the information is arranged around the core as diagram *elements*. To express the partial designs and their refinement, variables and constraints are used to represent the progressively refined diagram at different levels of detail. In addition, spatial constraints are formulated to help determine the sizes and locations of various diagram components (e.g., the length of various lines).

To accomplish the second goal, IMPROVISE plans a series of visual actions to allow certain information to be reinforced or revealed, based on the overview. For example, the pictures in Fig. 2(a-b) are created to reinforce the patient's demographics information and IV lines by using the visual action Highlight, while the pictures in Fig. 2(c-d) are planned to reveal the drip (intravenously administered drug) and lab report details.

The order and duration of these visual actions must also be coordinated with the order and duration of corresponding spoken references to produce a coherent multimedia narrative (Dalal et al. 1996). Fig. 3 shows a segment of coordinated media actions at the beginning of the presentation, where the patient's demographics are emphasized. Here, IMPROVISE highlights the leftmost portion of the patient's demographics first to coordinate with the first part of the spoken sentence. It then highlights the remainder of the demographics to synchronize with the rest of the spoken references. As we explain in Section 5, in this case IMPROVISE must adjust its own graphics constraints to cooperate with the speech generator.

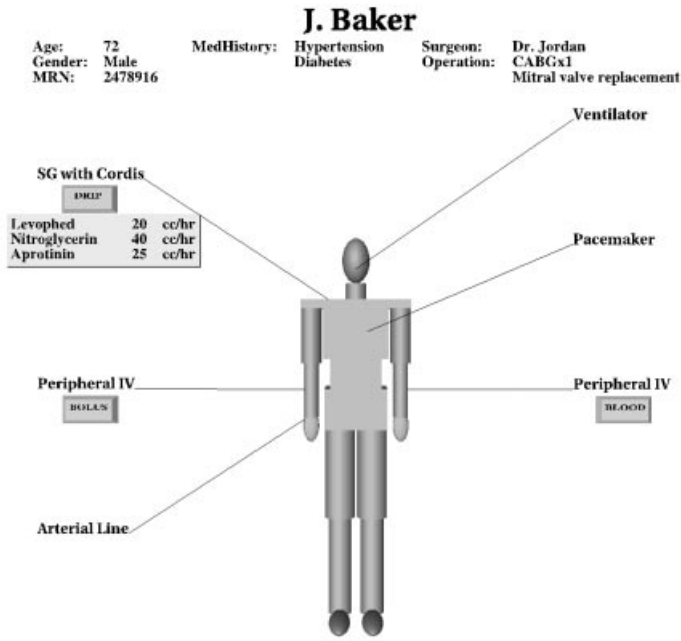


(a)

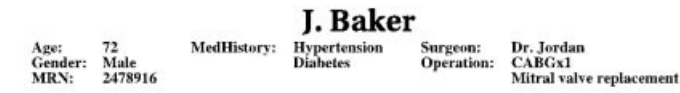


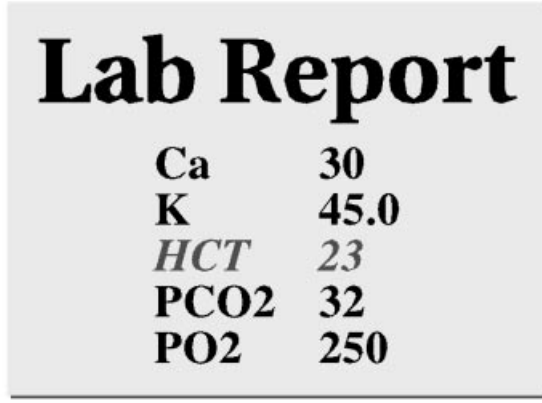
(b)

Fig. 2. Selected Keyframes Generated by IMPROVISE to Present a Patient's Information to a Nurse.



(c)





(d)

Fig. 2. (*continued*)

Speech: Mr. Baker is a seventy-two-year-old, hypertensive, diabetic male patient . . .

Speech: ... of Dr. Jordan undergoing CABG with mitral valve replacement

Fig. 3. Generated Coordinated Speech and Graphics.

4.2 Exploring a Computer Network

In this example, we show how IMPROVISE designs a new visual presentation by using sequences of animated visual transformations. Although these animations are currently used alone, the underlying design mechanisms and representation formalisms can be used to facilitate the design of a coordinated multimedia presentation (see Section 5). This example tackles the task of revealing the attributes of a user-selected network link. In a network management application, users often want to explore the internal structures of interesting network entities, such as links and nodes. For example, a link may contain a set of *virtual path segments* that have attributes such as capacity and utilization. Suppose that the user has selected link23, the link between Austin (node1), and Tucson (node5) in the network representation of Fig. 4(a).

IMPROVISE first formulates a communicative goal: Elaborate<link23>. Using a set of elaboration strategies and relevant data properties (e.g., that there are multiple links shown and this link has an internal structure), this communicative goal is then refined using two visual tasks (abstract visual actions): Focus<link23> (bringing the selected link into the center of focus) and Expose<link23> (revealing the link's internal structure). These two abstract visual actions are ordered to ensure the design's effectiveness. In particular, IMPROVISE will Focus on the selected object (link23) first and then Expose the object's internal structure.

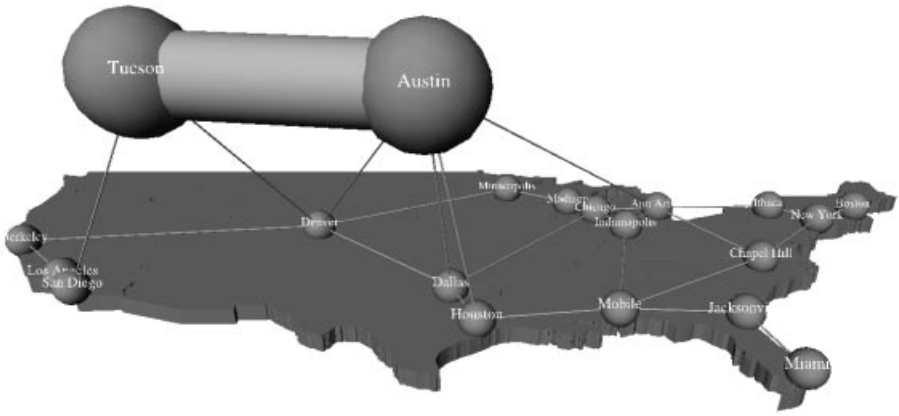
Since Focus is a composite action, it is associated with a set of decomposition strategies. These strategies state that Focus may be achieved using one of three actions: Separate, Enlarge, and Highlight. In this case, IMPROVISE chooses to refine Focus with Separate (separating link23 from the rest of the network). The rationale behind this decision is that a link is likely to intersect with other objects and Separate pulls intersected objects away to prevent any potential intersection while achieving focusing. In contrast, focusing by Enlarge increases the intersection possibility, and focusing by Highlight does not fix or prevent any intersection. Similarly, another composite action, Expose, is replaced by the

Fig. 4. Keyframes Generated by IMPROVISE to Reveal a Link's Internal Structure.

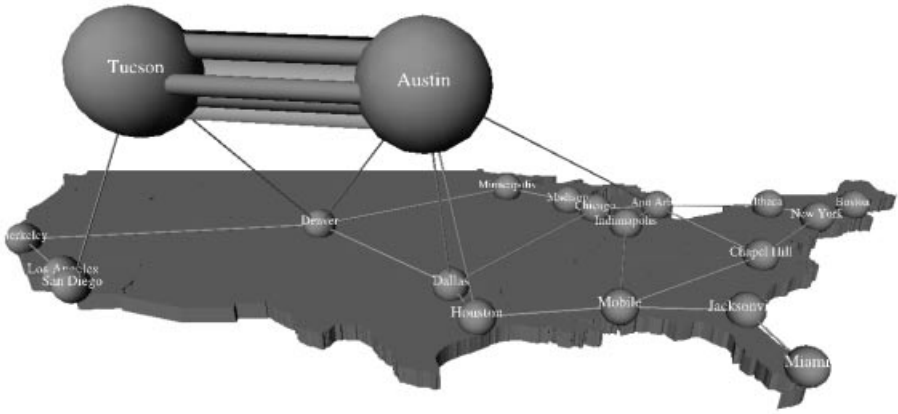
action Open (open link23), as the selected link is represented as a cylinder, which is a graphical object that IMPROVISE knows how to 'open'.

At this level of design, IMPROVISE needs to take care of the precondition of the action Expose. Expose requires that the visual representations of the internals must exist. Since the visual representation of the current link's virtual path segments does not yet exist, IMPROVISE needs to create one. As in the previous section's example, IMPROVISE generates the graphical representation for the virtual path segments.

When IMPROVISE advances to the next level of design, Separate, now the only remaining non-primitive action, needs to be decomposed. In this case, Separate is decomposed into two actions: Align and Move. Align rotates the whole network



(c)



(d)

Fig. 4. (*continued*)

so it is laid down (Fig. 4b), while Move moves the link and its two end nodes away from the rest of the network (Fig. 4c). A partial order is also specified: executing Align before Move to ensure a coherent transformation. The action Open is already a primitive action. However, its precondition requires that the object be large enough so that the opening effect is recognizable. To establish this precondition for Open, IMPROVISE chooses the action Scale. This also means that Scale must be executed before Open to achieve its precondition first.

When the plan is complete, we have a global order among actions (the attached numbers represent an animation's starting and ending times in seconds):

Align(0, 3) → Move, Scale, . . . (3, 6) → Open(6, 9)

Note that such a global order is not necessarily a linear order; for example, actions Move and Scale are executed simultaneously here. Rendered images of several keyframes from the resulting design are shown in Fig. 4(b-d).

5 Designing Multimedia Presentations

Although the previous two examples are quite different, IMPROVISE uses the same set of design strategies and inference mechanism with two different domain models: one for hospital patients and one for computer networks.

Based on these two examples, next we explain in detail how four important features of IMPROVISE play key roles in the design of coordinated multimedia presentations. We first show how IMPROVISE's data representation formalism and temporal model of visual techniques help provide useful information to other media components in the course of designing a coordinated multimedia presentation. We then describe how IMPROVISE's action-based inference engine can cooperate with other media generators to produce desirable visual actions for a multimedia presentation. Finally, we discuss how IMPROVISE's visual realizer synchronizes the execution of multiple media actions.

5.1 Data Representation

IMPROVISE is built to create a wide range of visual presentations for conveying heterogeneous data that can include both quantitative and qualitative information. To achieve this goal, IMPROVISE has a rich data semantic model. In other words, IMPROVISE has a representation of the underlying meaning of its input data, and uses this information to create graphical representations that communicate the data.

As data semantics may vary greatly from one application to another, it is difficult to build a data model that captures all possible individual data features. Thus, we develop a knowledge-rich, data semantic model based on a set of meta information. The meta information is organized by a *data characterization taxonomy* that abstracts a set of common visual presentation-related data properties (Zhou and Feiner 1996). Our taxonomy describes data characteristics along six dimensions:

type:	atomic vs. composite data entities
domain:	semantic categories, such as temperature or mass
attribute:	meta attributes, such as ordering, form, and importance
relation:	data connections, such as has-part and attributive relations
role:	data function, such as being an identifier or a locator
sense:	visual interpretation preference, such as symbol or label

Not only do these meta properties help us formulate portable (application-independent) design rules, but IMPROVISE can also use them to coordinate its actions with other media generators.

Different media may be used to convey different data entities or to present the same data entities from different perspectives. For example, in the medical example given above, the speech component may be instructed to summarize the received therapy, while IMPROVISE is directed to emphasize the critical components of the therapy. Thus, speech may state that the patient "has received massive cardiotonic therapy" and graphics may pinpoint the individual components involved in this therapy by highlighting the drips, the ventilator, and the pacemaker. In this case, IMPROVISE fulfils its assigned task by creating a highlighting action:

```
Highlight <drips, Ventilator, Pacemaker>
    (style OUTLINE) (startTime ...)(endTime ...)
```

This action specifies the objects to be highlighted, the highlight style, and the associated temporal constraints. Similarly, the speech component produces a speech action:

```
Speak <cardiotonic-therapy>
```

Since both the highlighting and speech actions are associated with temporal constraints, they must be coordinated to produce a coherent presentation. This is different from other automated multimedia presentation systems, which focus on generating written text with static graphical illustrations (Feiner and McKeown 1991; Mittal et al. 1998) or composing preconstructed media actions with generated actions (Towns, Callaway, and Lester 1998; Andre, Rist, and Müller 1998; Noma, Zhao, and Badler 2000). In these systems, temporal coordination is either not required or the coordination task is explicit (e.g., coordinating a generated action, such as a gesture, with a pre-existing media action, such as playing a video clip).

To coordinate different media actions, MAGIC's media coordinator must first identify corresponding actions among the media actions that it has received; for example, from a set of actions, it needs to relate the highlighting with a specific speech action. One approach for the media coordinator to relate corresponding actions is to trace all media generation processes to find out whether different actions are related to the same goal. In our example, the highlighting and the speech actions are created to accomplish the same goal (conveying the therapy). Nonetheless, it is often too costly for the media coordinator to understand and keep track of the complete generation processes of all different media generators. Thus, in our approach, IMPROVISE provides the media coordinator with additional information by exploiting the data characteristics. In this case, relying on data semantic relations, IMPROVISE can inform the media coordinator that the drips, Ventilator, and Pacemaker, are parts of the cardiotonic therapy. The media coordinator can then relate the two actions based on this information.

Based on data characteristics, IMPROVISE can selectively inform the media coordinator about the rationale behind its design decisions to facilitate coordination. Suppose that there is a quantitative ordering among a set of data entities to be presented. To create an effective illustration, IMPROVISE decides to present the data entities one by one in a sorted order (e.g., a descending order). Assume that a series of spoken sentences is also generated to explain each data entity in the set. To coordinate the visual displays with the spoken sentences, the media coordinator must come up with a temporal order that is compatible across both media. Besides offering the graphics ordering information, in our case, IMPROVISE can better inform the media coordinator by indicating that its presentation order is in fact determined by the sortable quantitative ordering attribute.

IMPROVISE may also selectively inform other media components about its intermediate actions. This information could help other media components create cross-references to material that it has generated (McKeown et al. 1992). Suppose that IMPROVISE informs the speech component that the data entities have been sorted. Here, the sorting action is an intermediate graphics action and it is output by IMPROVISE with the data ordering property. This information could then be used to generate more informative speech: the speech could first refer to the data set being presented in a particular order, and then explain individual data entities in that order. This type of content cross-reference may not be achieved in other multimedia generation systems, as other components can only utilize information about finalized graphical actions. For example, in COMET, the graphics generator IBIS can inform the language generator about the graphical actions it has decided to use (e.g., the generation of a cutaway view; see McKeown et al. 1992). But without a deep semantic model, IBIS has no way of telling which one of its intermediate steps would be important for the language generator to know.

5.2 Visual Techniques

IMPROVISE uses a visual presentation language to specify and manipulate graphical presentations (Zhou and Feiner 1998a). We characterize visual design knowledge divided into three types: visual objects, visual techniques, and visual design principles. *Visual objects* are the syntactic, semantic, and pragmatic encoding of visual patterns. *Visual techniques* are procedures that describe a graphical synthesis or manipulation process. *Visual design principles* are sets of rules that guide the proper application of visual techniques. We concentrate here on describing visual techniques, since their representations are most useful in designing coordinated multimedia presentations.

Visual techniques are used by IMPROVISE to assemble a new visual presentation or to modify an existing one. IMPROVISE has two main types of visual techniques: formational and transformational. *Formational* techniques create visual objects from scratch. For example, the formational technique DesignStructureDiagram creates a structure diagram to encode a particular set of input data entities. *Transformational* techniques modify existing visual objects. For example, the transformational technique Move modifies the location of a visual object.

Unlike IBIS and SAGE, IMPROVISE can employ visual techniques with temporal constraints (Zhou and Feiner 1998b). Compared to other planning systems that handle temporal constraints (Wilkins and Myers 1995; André and Rist 1996), IMPROVISE uses multilevel topological and metric temporal constraints to specify visual actions at a finer granularity. This unique feature enables IMPROVISE to cooperate effectively with other media generators to produce fine-grained, coordinated multimedia presentations. Specifically, IMPROVISE can specify/modify visual techniques with both temporal order and duration constraints, since these two types of constraints are the basis for coordinating animated graphics with other temporal media (Dalal et al. 1996).

First, IMPROVISE can use qualitative temporal constraints (e.g., before and after) to specify the ordering between two visual techniques. For example, IMPROVISE may generate a set of highlighting actions to stress in sequence the individual drip items depicted in Fig. 2(c). These actions are ordered to produce a natural, top-to-bottom, highlighting effect. After receiving this graphical ordering information, the media coordinator can order the corresponding spoken references to produce a coordinated presentation. Note that, in this case, graphics is constrained by its spatial layout, but the speech does not have a preference as to the order in which the items should be spoken. As will be seen next, IMPROVISE can also adjust its constraints to cooperate with a speech action.

In addition to defining qualitative ordering constraints, IMPROVISE can also specify quantitative temporal durations within a visual technique (usually indicated in seconds). In particular, IMPROVISE can define up to four temporal durations within a visual technique: An absolute startTime and endTime define a total duration; animDuration is the time taken to turn on the desired visual effects (e.g., gradually changing the colour of an object to highlight it); holdingDuration is the time spent on keeping the effect on the screen (e.g., holding the highlighting effect); and animOffDuration limits the time taken to reverse the visual transformation (e.g., turning off the highlighting). These durations greatly simplify the definition of a complex animation without compromising its function. For example, we can reverse the animated effect easily without explicitly introducing an undo action (e.g., turning off highlighting without using a separate 'unhighlight' action).

More importantly, IMPROVISE's approach to temporal durations helps produce fine-grained temporal media coordination. For example, IMPROVISE does not have a rigid duration constraint as to how long the patient identification and medical history should stay highlighted (Fig. 3). Consequently, using speech's duration constraints, IMPROVISE can fine-tune its highlighting duration to ensure that the highlighting is turned on or off at the exact moment when the corresponding speech segment starts or finishes.

To facilitate temporal media coordination, IMPROVISE also uses flexible time-window constraints. For example, IMPROVISE may specify that a highlighting action needs a minimum of 1s or maximum of 2s to turn on the highlight, and another 3s to 5s to keep the highlighting. The media coordinator can use the time window to compute a time duration acceptable for both graphics and speech.

In our previous example of highlighting drip items, the media coordinator may compute an agreeable time duration to ensure that the duration of each spoken reference to these items is not too short to cause a blinking effect, and that the graphical highlighting is not too long to create an awkward silence.

5.3 Action-Based Inference Engine

To create coherent, animated graphical illustrations, IMPROVISE employs an action-based inference engine. Given a set of inputs, including the data to be conveyed, the presentation intent, and the relevant presentation context, our engine can automatically create animated illustrations using sequences of temporally-ordered visual actions (instantiated visual techniques).

The core of the engine is a top-down, hierarchical-decomposition, partial-order planner that employs visual techniques as planning operators and visual design principles as planning constraints. Using a top-down hierarchical-decomposition strategy (Young, Pollack, and Moore, 1994), IMPROVISE first sketches a visual design that is complete, but too vague to be realized; then it refines the vague parts of the design into more detailed subdesigns, until the design has been refined to sequences of visual actions that can be executed by a realizer. To facilitate this top-down hierarchical design process, we have the notion of primitive and abstract techniques. Primitive techniques, such as Move and Scale, are clearly defined by a parametrized visual procedure. In contrast, abstract techniques, such as DesignStructureDiagram, which may contain other abstract or primitive techniques, are only partially specified. Abstract techniques must eventually be decomposed into a set of primitive techniques to be carried out.

In keeping with this approach, IMPROVISE's design process includes an action-decomposition process, in which abstract visual actions are recursively replaced with a set of more specific, partially-ordered subactions. In the network example presented earlier, the Separate action is decomposed into two actions: Align and Move. In addition to action decomposition, IMPROVISE may also require object decomposition. For example, a DesignStructureDiagram action may be decomposed into a set of subactions that define individual diagram components, such as the header. Accordingly, the data input used to produce the diagram must also be decomposed into smaller units that can be used by the subactions. Similar to action decomposition, object decomposition also produces partially-ordered data subunits. Since we have described elsewhere how IMPROVISE handles action and object decomposition (Zhou1999), we focus here on how our inference engine uses action/object decomposition to produce flexible visual plans that benefit the design of a coordinated multimedia presentation.

Using a planning approach, our inference engine produces a visual plan. As the plan bears sequences of actions, IMPROVISE can communicate with the media coordinator to negotiate the ordering of these actions. In particular, visual actions in IMPROVISE are partially ordered only if there is insufficient information. Generating a partial order not only provides a negotiation ground for IMPROVISE

to cooperate with other media generators by relaxing its constraints, but also helps improve the design efficiency by reducing unnecessary backtracking.

In the example shown in Fig. 3, IMPROVISE produces the following actions with two sets of possible partial orders to accomplish the task of emphasizing the patient's demographics information:

```
Action1: Highlight <demographics>
Action2: Highlight <mrn, age, gender>
Action3: Highlight <medhistory>
Action4: Highlight <surgeon operation>

Partial orders:
1. (contains Action1
       ((before meet) Action2 Action3 Action4))
2: (contains Action1
       (* Action2 Action3 Action4))
```

Here, using Allen's (1983) temporal relations, the first set of partial orders specifies that Action1 starts before and ends after all the other actions (contains); and that among the other three actions, each action (e.g., Action2) ends either before or at the same time that the next action (e.g., Action3) in the list starts (before or meet). The second set states the same relationship between Action1 and the other three actions, but there is no particular ordering among the rest of the three actions (* relation). After receiving the negotiation requests from the media generators, the media coordinator uses MATS (Kautz 1991) to compute a compatible order specified in terms of the objects (Dalal et al. 1996); in this example, the order returned to IMPROVISE is:

```
(contains demographics
    ((before equal meet) mrn age medhistory gender
                         surgeon operation))
```

By adapting this compatible order, IMPROVISE can then refine its own partial orders to produce a complete order of graphical actions. In this case, IMPROVISE must refine the second set of partial orders because the first set is incompatible with the negotiated results:

```
Complete order: (contains Action1
                    (meet Action5 Action4))
Action5 = (merge Action2 Action3)
```

Here a new action Action5 is created by IMPROVISE to resolve the conflict between the negotiated order and the structure of current graphical actions. Specifically medhistory comes between age and gender, effectively breaking the structure of Action2. In this case, IMPROVISE is able to merge Action2 and Action3 together, because the two actions have similar goals (emphasizing part of demographics), the same type (highlighting), and the same constraints (e.g., colouring style). Thus, a new action is generated to replace Action2 and Action3:

```
Action5: Highlight <mrn age gender medhistory>
```

Improvise can replan because our action-based inference engine maintains the history and state of actions as steps are begun and completed. As shown in this example, the complete history (e.g., goal association) and state of the actions (e.g., the constraints specified within actions) provide much-needed information for replanning.

In addition to partial-order planning/replanning, the top-down hierarchical design strategy employed by our inference engine also facilitates media coordination. Instead of waiting for all graphical actions to be fully specified near the end of the design process, Improvise can incrementally provide other components with a set of visual actions at different levels of detail. For example, at a high level, Improvise may inform other components about its decision to use two abstract actions Focus and Expose to gradually reveal the network link's internals (Fig. 4). At a lower level, Improvise may advise other components that it chose the primitive action Open to achieve Expose.

By outputting these progressively refined visual actions at each level of planning, Improvise can allow the media coordinator and other media components to know what can or cannot be done early on. As a result, critical design conflicts can be detected early, and timely remedies can be tried to avoid costly backtracking. Once learning that at a high level Improvise must use Focus before Expose to gradually reveal the link's internals, the media coordinator could compute a compatible order at an early design stage to make sure that the order of corresponding speech actions matches the order used in graphics.

On the other hand, through the exchange of information, Improvise could also cooperate with other media generators by gradually incorporating their feedback into its planning process. For example, at a high level, and with insufficient constraints, Improvise may only know that it needs to emphasize the drip items as a whole (highlighting the entire list). Through the exchange of information, it may learn that the speech generator will enumerate the drip items in sequence. Using this information, Improvise could refine its current visual emphasis plan by adding sub-highlighting, which would highlight each item in turn as the item is spoken.

5.4 Portable Visual Realizer

Once a visual plan is complete, it needs to be realized by a *realizer*. To make our realizer portable, we separate Improvise's realization from its visual design process by employing an intermediate presentation authoring language (PAL). Similar to the node-based scene graph descriptions used in Open Inventor (Wernecke 1994), PAL is an object-oriented authoring language that represents every visual object and visual technique as a node (e.g., a material node). A complex node (e.g., a table chart node) may recursively consist of a collection of simpler nodes (e.g., text nodes). High-level design specifications, such as the temporal constraints among visual actions, and low-level graphics details, such as the geometric properties of visual objects, are all encapsulated in the nodes. Moreover,

new media actions or design patterns can be easily added as new types of nodes. In fact, we have incorporated a speech action node in PAL and these speech actions can be treated as the same as visual actions. For example, using PAL, the set of coordinated graphical and speech actions depicted in Fig. 3 can be represented as four nodes:

```
DEF Act1 Highlight {
        operands demographics
        style MARKER       // change the background colour
        startTime 0.0
        endTime 7.66
        color 1.0 1.0 0.0  // use a yellow background
}
DEF Act2 Highlight {
        operands [mrn, age, gender, medhistory]
        style COLORING     // change the text colour
        startTime 0.0
        endTime 5.25
        color 1.0 0.0 0.0  // use red text
}

DEF Act3 Highlight {
        operands [surgeon, operation]
        style COLORING
        startTime 5.25
        endTime 7.66
        color 1.0 0.0 0.0
}
DEF Act4 Speak {
        operands [name, age, medhistory, gender,
                  surgeon, operation]
        startTime 0.0
        endTime 7.66
}
```

Once these actions are expressed using PAL, the realizer parses the PAL descriptions. The challenge now is to execute this set of actions while faithfully maintaining all temporal constraints. In our approach, we have implemented a time queue to schedule various actions and ensure that their temporal constraints are met. All actions are first entered in the time queue by their starting times. The scheduler then uses a global alarm clock to invoke actions when their starting times are reached. A local timer is also maintained within each visual action to signal its termination when its finishing time approaches.

This approach works fine until the following problem arises: Two closely scheduled actions (e.g., actions A and B in Fig. 5) may overlap because the scheduler cannot guarantee a full stop in a previous action (e.g., A) when its local timer expires. This can happen because the local timer does not account for the time spent for executing various implicit finishing acts. For example,

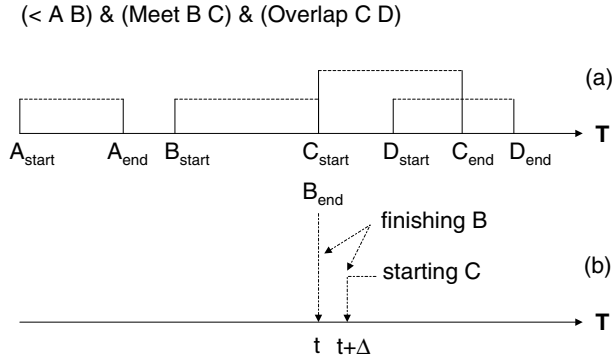


Fig. 5. Time Queue for Action Execution.

action A may call an instantaneous undo act (animOffDuration is 0.0s) when its local timer expires. Thus, there is no guarantee that A's undo act will be finished before B starts.

To fix this problem, each action is required to signal the scheduler when it is truly finished. In addition, we insert a dummy finishing act for each action in the time queue at its finishing time to ensure that the global clock will be stopped if the previous action has not finished. As shown in Fig. 5(a), when the global clock reaches the dummy act A_{end}, it will not be advanced to action B_{start} until it receives A's finishing signal.

The above approach fixes only half of our problem: it works for actions scheduled one after another (e.g., A and B in Fig. 5a), but not for actions scheduled right next to each other (e.g., B and C). In this case, the realizer is expected to execute two tasks simultaneously: finishing the previous action (B) and starting a new action (C). In our uniprocessor implementation, these tasks will be executed in a nondeterministic order. This may result in undesirable visual effects. Let B be Act2 and C be Act3 from the previous example; here B must unhighlight [mrn, age, gender, medhistory] before C starts to highlight [surgeon, operation]. Because of the nondeterministic execution order, C might start before B finishes, causing an undesired visual effect: two sets of objects highlighted at the same time instead of in sequence.

To ensure desired visual effects, we add sub-order temporal constraints to serialize simultaneous actions using heuristics. For example, one of IMPROVISE's heuristic rules asserts that all dummy finishing acts precede any other action scheduled at the same time. In the above example, the plan agent will process B_{end} before C_{start}, as if the time point t is expanded into a time interval [t, t+Δ] (Fig. 5b). This ensures that all objects in action B are unhighlighted before any object in action C is highlighted.

6 Conclusions and Future Work

In this chapter, we have presented IMPROVISE, an automated graphics generation system that can automatically create sequences of coherent, animated graphi-

cal illustrations to convey a wide variety of data. We explained how IMPROVISE facilitates the design of coordinated multimedia presentations, and examined IMPROVISE's knowledge-rich data representation model and its role in providing useful information for other components to make informative and cooperative design decisions. We introduced IMPROVISE's visual techniques and their temporal constraints, and explained how these temporal constraints facilitate temporal media negotiation. We also described IMPROVISE's action-based inference engine and demonstrated how its partial-order and hierarchical design approaches make it possible for IMPROVISE to negotiate incrementally with other media generators. Finally, we presented IMPROVISE's visual realizer, emphasizing its action-execution scheduler.

We are currently working on ways to improve IMPROVISE so that it can be used to create more sophisticated multimedia presentations. One approach that we are taking is to extend IMPROVISE to generate interactive graphical illustrations whose illustrative objects are also actionable. This would allow users to ask for new information (e.g., relevant details) or to manipulate the information already presented (e.g., to compare two data objects). By producing interactive illustrations, IMPROVISE could cooperate with other media components to create coordinated, interactive multimedia presentations.

Building upon current visual action manipulation strategies (e.g., the action merge strategy), another direction is to investigate additional strategies to augment IMPROVISE's replanning capabilities for situations when the compatible order returned by the media coordinator does not agree with IMPROVISE's graphical action structure. For example, the media coordinator may inform IMPROVISE that the devices and treatments (e.g., IV lines) illustrated in the patient example need to be presented sequentially, instead of together as they are now. In this case, a separation strategy is needed to effectively break the single display action into several display actions that handle one item or several relevant items at a time (e.g., to ensure that left and right IVs always appear simultaneously).

Acknowledgments

We thank Rahamad Dawood for implementing the scheduler, and Blaine Bell for porting the entire system from IRIX to Windows. This research was supported in part by DARPA Contract DAAL01-94-K-0119, the Columbia University Center for Advanced Technology in High Performance Computing and Communications in Healthcare (funded by the New York State Science and Technology Foundation), the Columbia Center for Telecommunications Research under NSF Grant ECD-88-11111, and ONR Contract N00014-97-1-0838.

References

Allen, J. (1983) Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

André, E. and T. Rist (1993) The design of illustrated documents as a planning task. In M. Maybury, editor, *Intelligent Multimedia Interfaces*, Menlo Park, CA: AAAI Press/The MIT Press, 94–116.

André, E. and T. Rist (1996) Coping with temporal constraints in multimedia presentation planning. In *Proceedings AAAI '96*.

André, E., T. Rist, and J. Müller (1998) Webpersona: A life-like presentation agent for the world wide web. *Knowledge-Based Systems*, 11(1):25–36.

Dalal, M., S. Feiner, K. McKeown, S. Pan, M. Zhou, T. Höllerer, J. Shaw, Y. Feng, and J. Fromer (1996) Negotiation for automated generation of temporal multimedia presentations. In *Proceedings ACM Multimedia'96*, Boston, MA, November 18-22, 55–64.

Feiner, S. and K. McKeown (1991) Automating the generation of coordinated multimedia. *IEEE Computer*, 24(10):33–41.

Kautz, H. (1991) *MATS (Metric/Allen Time System) Documentation*. AT&T Bell Laboratories.

Mackinlay, J. (1986) Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141.

McKeown, K., S. Feiner, J. Robin, D. Seligmann, and M. Tanenblatt (1992) Generating cross-references for multimedia explanation. In *Proceedings AAAI'92*, 12–17.

Mittal, V., J. Moore, G. Carenini, and S. Roth (1998) Describing complex charts in natural language: A caption generation system. *Computational Linguistics*, 24(3):431–467.

Noma, T., L. Zhao, and N. Badler (2000) Design of a virtual human presenter. *IEEE Computer Graphics and Applications*, 20(4):79–85.

Roth, S. F. and J. Mattis (1991) Automating the presentation of information. In *Proceedings IEEE Conference on AI Applications*, 90–97.

Seligmann, D.D. and S. Feiner (1991) Automated generation of intent-based 3D illustrations. *Computer Graphics*, 25(4):123–132.

Towns, S., C. Callaway, and J. Lester (1998) Generating coordinated natural language and 3D animations for complex spatial explanations. In *Proceedings AAAI '98*, 112–119.

Wahlster, W., E. André, H. Finkler, J. Profitlich, and T. Rist (1993) Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63(12):387–427.

Wernecke, J. (1994) *The Inventor Mentor: Programming Object-Oriented 3D graphics with Open Inventor*. Reading, MA: Addison Wesley.

Wilkins, D. and K. Myers (1995) A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation*, 5:731–761.

Young, R.M., M.E. Pollack, and J.D. Moore (1994) Decomposition and causality in partial-order planning. In *2nd Intern. Conference on AI Planning Systems: AIPS-94*, Chicago, IL, June 1994, 188–193.

Zhou, M. (1999) Visual planning: A practical approach to automated visual presentation. In *Proceedings IJCAI'99*, August 1999, 634–641.

Zhou, M. and S. Feiner (1996) Data characterization for automatically visualizing heterogeneous information. In *Proceedings IEEE InfoVis'96*, San Francisco, CA, October 1996, 13–20.

Zhou, M. and S. Feiner (1998) Automated visual presentation: From heterogeneous information to coherent visual discourse. *Journal of Intelligent Information Systems*, 11:205–234.

Zhou, M. and S. Feiner (1998) Efficiently planning coherent visual discourse. *Journal of Knowledge-Based Systems*, 10(5):275–286.