Automated Visual Presentation: From Heterogeneous Information to Coherent Visual Discourse

Michelle X. Zhou **Steven K. Feiner**
Department of Computer Science
Columbia University
500 West 120th St., 450 CS Building
New York, NY 10027
{zhou,feiner}@cs.columbia.edu

Abstract

Automated visual presentation systems should be able to design effective presentations for heterogeneous (quantitative and qualitative) information. They should also be able to work in static or interactive environments and capable of employing a wide range of visual media and visual techniques. In this paper, we focus on three tasks in building visual production systems: establishing a thorough understanding of the presentation-related characteristics of domain-specific information; classifying several types of visual information and capturing their distinct syntactic, semantic, and pragmatic features; and formulating a set of design principles.

We define a data-analysis taxonomy to characterize heterogeneous information. In addition, we have modeled presentation context information such as audience identity to produce user-centered visual design. To utilize and manipulate visual information, we have classified it into visual objects and visual tools based upon its role in the visual production process. To guide the visual design process, we have formulated a set of design principles that ensure the expressiveness and effectiveness of a design. To test and evaluate our work, we have developed a prototype system called IMPROVISE based on the research results. We use examples generated by IMPROVISE to illustrate how it constructs visual presentations.

Keywords: Automated graphics generation, heterogeneous data characterization, visual information classification, coherent visual discourse, visual design principles.

1 INTRODUCTION

For many years, visual presentations have been used to make information more understandable by humans. However, as the complexity and volume of information increase, the tasks of data manipulation, exploration, and inference quickly begin to exceed human capacity. To cope with large quantities of information in a timely, customized manner, researchers are developing automated visual presentation systems. These systems promise to simplify an application programmer's design tasks by providing a set of design rules that choose appropriate visual forms for different information (Feiner, 1985; Mackinlay, 1986; Roth and Mattis, 1991).

Automated visual presentation systems may be viewed as automated production systems, whose input is the information to be presented, and whose output is the desired visual presentation. From this perspective, previous automated visual production systems (e.g., Mackinlay, 1986; Roth and Mattis, 1991; Feiner, 1985; Seligmann, 1993) have confined their production capabilities by either limiting the types of their input (e.g., dealing with only quantitative or qualitative information) or restricting the forms of their output (e.g., producing only individual presentations). To produce more effective visual presentations for a wider range of information types, our work focuses on automatically producing coherent visual discourse for heterogeneous information. Here, *heterogeneous information* includes both quantitative and qualitative, and dynamic and static information. *Quantitative* information refers to discrete or continuous numerical scales or ranges, and *qualitative* information abstractly describes various aspects of domain objects, such as categorical and relational information. Whereas *static* information remains unchanged, *dynamic* information varies with time. *Visual discourse* is a series of connected visual displays, of which a single display or a series of static displays may be considered as special cases. To remain *coherent*, a visual discourse must ensure smooth transitions between displays, maintain consistent designs within and among displays, and achieve effective visual unifications among various components.

The information to be presented is the input of a visual production system. This information could vary a great deal from one domain to another. To formulate general rules that help map the information to visual discourse, we must understand presentation-related properties that are common to all information. This task is called *data characterization* (Roth and Mattis, 1990). To characterize heterogeneous data systematically and comprehensively, we have developed a six-dimensional data-analysis taxonomy. Based on this taxonomy, we discuss how domain data is organized and represented to capture various presentation-related properties and aid the design process. In addition to characterizing data, we have also modeled presentation context information, such as expected users, situations, or equipments, to produce user-centered visual design. To facilitate data manipulation and inferencing, we represent both domain and context information in an object-oriented paradigm.

In our approach, the visual production process utilizes a set of visual tools to either assemble new visual objects from scratch, or transform existing visual objects into new ones. We partition visual information into two categories: visual objects and visual tools. *Visual objects*, in this paradigm, are syntactic, semantic, and pragmatic encodings of various visualization units defined at different levels of abstraction, ranging from the most basic visual variables (Bertin, 1983) to high-level complex visual forms, such as animated sequences of 3D pictures (Lohse et al., 1994). *Visual tools*, on the other hand, are visual techniques that are used to assemble or modify visual objects. These visual tools can be further distinguished by their functions: assembly tools that compose visual objects and transformational tools that modify existing visual objects. Visual tools are also defined at multiple levels of abstraction. Abstract visual tools deal with complex visual objects or serve higher level purposes (e.g., revealing an object's components), while more concrete visual tools act upon simpler visual objects for a more specific task (e.g., changing the color of an object).

Based on the characterization of domain information and categorization of visual information, we formulate a set of design rules that are used to guide the visual production process and ensure the coherence of visual discourse. To apply design rules for different visual objects at different stages, we define these rules in terms of their achievements at different levels of abstraction. At a high level, design rules are used to ensure global expressiveness or effectiveness (Mackinlay, 1986) of a visual discourse, while at a low level, they maintain semantic integrity or coordinate visual fitness within a single visual object synthesis.

In the rest of this article, we first discuss related research in the area of automated visual presentation design. Then we present our work, concentrating on the characterization and representation of domain and context information, the categorization and representation of visual information, and the formulation and usage of design rules. We also introduce a prototype system, IMPROVISE (Illustrative Metaphor PROduction in VISual Environments), which has been implemented to demonstrate the utility and coverage of our approaches. We also use concrete examples to illustrate how IMPROVISE generates different presentations. Finally, we present our conclusions and discuss future work.

2 RELATED WORK

Robertson's natural scene paradigm (NSP) (Robertson, 1991) is a graphics synthesis system that maps data variables to various features of the natural scene, such as surface condition, surface height, and density, based on the characteristics of the data. To direct the process, he analyzes the natural properties of data and interpretation goals, develops a catalogue that includes the capability of each visual variable that can be used in the natural scene, and establishes the mappings between the visual capabilities of the natural scene and the interpretation goals. However, NSP focuses only on addressing characteristics of quantitative data and their representation forms. In contrast, we attempt to find a more comprehensive and systematic treatment for representing heterogeneous data, including both quantitative and qualitative information.

Both APT (Mackinlay, 1986) and SAGE (Roth and Mattis, 1991) automatically generate 2D visual presentations of quantitative information. They both adopt a systematic data characterization method for quantitative data, and a visual language for describing various 2D graphs and charts. However, the complexity of data types handled by SAGE and the richness of its visual language are significantly advanced over APT. Unlike APT, which designs visual displays completely relying on the data characteristics, SAGE also incorporates information seeking goals into its design process (Roth and Mattis, 1990). However, both systems focus on generating individual displays for quantitative information. They do not address issues such as providing transitions between displays or integrating new information into existing displays.

IBIS (Seligmann and Feiner, 1991) automatically generates 3D realistic pictures individually or in series to depict physical objects and actions based on communicative intents. To map communicative intents to illustrations, IBIS stores the physical objects along with their features in the knowledge base, defines a set of illustration objects that actually appear in its pictures, and establishes a set of design rules. To make a series of coherent illustrations, IBIS addresses continuity and consistency issues by taking the previous picture into account while it designs the next one. For example, it maintains a consistent viewing specification whenever possible, and when designing composite illustrations that include nested insets, it allows child illustrations to inherit visual properties from their parent. However, IBIS focuses on generating a series of realistic pictures of 3D worlds: it does not consider design consistency issues at the symbol encoding level (e.g., mapping object attributes to visual variables). Neither does it explicitly address transition issues between pictures, or deal with integrating new information into existing presentations.

3 APPROACH

An effective visual presentation must reflect the properties of the information to be presented and satisfy the user or application's information seeking goals. In this section, we first present a data characterization taxonomy that comprehensively and systematically describes what information properties are related to visual presentations and how they are related. To capture the user or situation context in a visual production design, we briefly discuss our context information model. Following the analysis of domain and context information, we illustrate the organization and representation of various visual information, including both visual objects and visual tools. Once we have established the representation and characterization paradigm of domain and visual information, we describe our rule base, in which design rules are defined for design tasks at multiple levels of abstraction. We also describe the usage of the rules in the context of a visual production process.

3.1 Domain Information Characterization

As general human knowledge is quite complex in structure and abundant in quantity, we could not possibly develop a hierarchy that exhaustively represents every type of information and its properties. Instead, we have developed a taxonomy that adds meta-information on top of the domain data and can be easily extended to accommodate new additions. Our taxonomy describes data characteristics along six dimensions: data type, data domain, data attributes, data relations, data role, and data sense. *Data type* distinguishes a piece of information by whether or not it is divisible into smaller pieces. *Data domain* categorizes information in a semantic taxonomy; for example, whether or not it is a physical entity. *Data attributes* differentiate data based on properties such as an object's shape, and *data relations* specify connections among data. The last two dimensions characterize data pragmatically based on user information-seeking goals (*data role*) and user visual-interpretation preferences (*data sense*).

In our approach, each piece of domain information is considered as a domain object, which is represented in Figure 1a. Domain objects are organized by their domain types. There are four top-level domain types, to be described in detail later, which become the top-level object *classes*: *entity*, *concept*, *measurement*, and *event* (Figure 1b–c). Other data dimensions (e.g., type and attributes) become *slots* in a *class*. The slot components is used to record constituents of an object, while slot conveyElements indicates which aspect or components need to be conveyed in the current presentation. As a domain object may have different visual representations within a visual discourse, slot visual-rep records the *first* created visual representation and slot visual-extension records *multiple* visual representations that usually evolve from the original representation in the course of the visual discourse development.

To easily manipulate complex attributes or relations, we consider each type of *attribute* or *relation* as an object class itself (KRSL, 1993). Thus, we organize different types of attributes and relations (e.g., visual attributes vs. domain-specific attributes) into different PROPERTY or RELATION classes. For example, visual attributes such as *form*, and *material* become slots of VISUAL-PROPERTY class (Figure 1d).

(*****Insert Figure1 Near Here*****)

A specific domain object is therefore an *instance* of one of the classes or their subclasses. For example, the car Dante (Figure 1b) is an instance of class CAR, which in turn is a subclass of ENTITY; while the patient Jones's medical information (Figure 1c) is an instance of INFORMATION, which is derived from CONCEPT. As we examine each of the data characterization dimensions in detail, we also use these instances to illustrate how domain objects are represented in our approach.

Data Types

There are two basic data types: *atomic* and *composite*. An *atomic* object is the most basic, indivisible data unit. A *composite* object is a combination of atomic or composite objects. Knowledge of the data types can facilitate the construction of displays to represent them. Usually, designing or manipulating a visual presentation for a composite object is decomposed into tasks that design or manipulate visual presentations for its components. To reflect the data type of a domain object, the slot type is filled with either ATOMIC or COMPOSITE. If the value is COMPOSITE, the slot components is also likely to have multiple values corresponding to various parts of the current object. For example, both Dante and Jones-info are considered as composite objects with their own unique components(*********Insert footnote 1 here*************):

```
Dante of CAR
   (type COMPOSITE)
   (components [Dante-Engine] [Dante-Gas-Tank] . . .)

Dante-Engine (is-a CAR-ENGINE)
   (type ATOMIC)
   (components)

Jones-info of INFORMATION
   (type COMPOSITE)
   (components [Jones-demographics] [Jones-lab] . . .)

Jones-demographics of INFORMATION
   (type COMPOSITE)
   (components [Jones-identification] [Jones-medhistory]
                  [Jones-operation])
```

Data Domain

To construct an effective visual presentation, it is often important to take into account the data's domain, or the general ontology, which usually is used to categorize a broad range of objects and relations (KRSL, 1993; Russell and Norvig, 1995). Roth and Mattis (Roth and Mattis, 1990) characterize data with regard to quantitative domains such as time, space, temperature, and mass; for example, to allow temperature to be preferentially displayed along a vertical axis. We generalize this classification to accommodate a broader range of information. Each object belongs to one of the following top-level domains, which can themselves be further subdivided: *entity*, *concept*, *measurement* and *event.*

An *entity* is an object that exists independently. It may correspond to a physical object in the real world (e.g., a car) or one that may be depicted as a physical object (e.g., a dragon). A *concept* is an abstract idea that either exists independently by itself or must be attached to other objects. Based on general AI ontologies (e.g., KRSL, 1993; Allen87), a concept could be further refined into general categories such as *attribute*, *information*, *spatial-extent*, and *temporal-extent*. A *measurement* is a quantitative or qualitative value, with or without its unit of measure, which is usually related to a concept, such as weight. An *event* is a composite object that describes something that happens at a given location or time with certain consequences. An event could be further distinguished by its cause: *natural event* and *human activity.* To visually represent every piece of domain information, we consider everything to be an object. For example, each patient has attributes such as age and gender. Instead of filling slots age and gender with simple strings or numerals for a specific patient, we create instances of class ATTRIBUTE:

```
(defclass PATIENT (is-a ENTITY)
    (slot age)
    (slot gender) . . .)

Jones of PATIENT
   (age [Jones-age])
   (gender [Jones-gender] . . .)

Jones-age of ATTRIBUTE
   (attributeName "age")
   (value [Jones-age-val])

Jones-age-val of MEASUREMENT
   (unit YEAR)
   (value 80)

Jones-gender of ATTRIBUTE
   (attributeName "gender")
   (value Female)
```

Data Attributes

Although, at first glance, information and its visual representations may appear to differ greatly from one application domain to another, there are basic connections between the nature of the information (its attributes) and the presentation style. Therefore, presentation-related properties can be abstracted for all types of information. Some properties are shared by different types of data, while others are unique to a particular data type. First, we describe the data attributes that are common to all data types: *form*, *material*, *transience*, and *importance.*

Form describes the natural shape properties of an object. Generally speaking, an object's form is either shaped, shapeless, or none. A *shaped* form refers to a category of objects that have solid, physical shapes in the real world, such as a person or a car. A *shapeless* object might have a shape under certain circumstances, but its shape changes too often or too abstract to be precisely depicted. An example of a shapeless object is water. All conceptual objects use *none* as their form description. While these general notions of shape property help determine appropriate visual forms at a high level (e.g., a concept usually is not represented figuratively), more detailed shape descriptions are needed to reflect the inherent visual forms of an object. For example, conveying a patient's chest x-ray is usually related to displaying the x-ray's *image*. To correctly present such inherent visual forms of different domain objects, we capture both general and detailed shape descriptions. A detailed description may indicate a particular type of visual form, such as a graphical model, or a specific image that an object *inherently* embodies. For example, the patient Jones has the following shape description:

```
(defclass SHAPE-PROPERTY
    (is-a VISUAL-PROPERTY)
    (slot description)
    (slot model)
    (slot image)
    (slot video)
    (slot textual-info))

Jones-attr
    of VISUAL-PROPERTY
    (form [Jones-form])
    (material [Jones-material])

Jones-material of COLOR
    (rgb 0.7 0.3 0.25)

Jones-form of SHAPE-PROPERTY
    (description SHAPED)
    (model [Jones-model])

Jones-model of MODEL
    (fileName "human.model")
```

Material refers to the possibly complex set of appearance variables that determine how an object interacts with light. Material models can range from simple RGB color through sophisticated physics-based illumination models. The above description fragment also indicates the patient's flesh color through its material description.

While the *form* and *material* describe a data object's natural visual appearance, other data attributes such as transience and importance are used to describe the intrinsic properties of the information mentioned in (Arens et al., 1993). These are especially useful for grouping or partitioning information when a large amount of information is to be conveyed. *Transience* describes whether an object is *dynamic* or *static.* Dynamic information, such as network traffic load, varies over time, while static information, such as a patient's gender, remains constant. *Importance* indicates that some information is more important than others (e.g., an abnormal patient condition is more important than a normal one), and this property allows us to select different graphical techniques to *visually* group or partition information (e.g., to highlight the important information).

We adopt the composite object attribute categorizations discussed in (Roth and Mattis, 1990) and (Arens et al., 1993), and extend them to address a wider range of information. The following attributes are tied directly to composite objects only: *ordering*, *scalability*, and *continuity*.

Within a composite object, components may be *ordered* in a certain way. To effectively map data onto visual objects and reflect the property of such a specific ordering, we need to understand the properties of various orderings and the mapping rules between different orderings and visual variables. For example, different *sizes* cannot normally be used to encode an *ordinal* ordering, as different *shapes* are ineffective in representing a *quantitative* ordering. How quantitative ordering affects graphic design decisions has been carefully examined in both (Mackinlay, 1986) and (Roth and Mattis, 1990). We extend the ordering attribute to qualitative data, and analyze how such qualitative ordering influences the spatial and temporal order of an information presentation. Within a composite object, its components may be *logically* ordered. For example, the components of patient Jones's demographics information are logically ordered such that identification information (e.g., age and gender) comes before medical history. Using this property, the display (with demographics shown at the top of Figure 2a) is arranged in a way that presents identification (MRN, age, gender) before medical history (History) based on a left-to-right reading order.

(*******Insert Figure2 Near Here*******)

Roth and Mattis (Roth and Mattis, 1990) further distinguish quantitative data sets using coordinates and amounts. *Coordinates* specify *points* in a domain (e.g., the purchase date of the car), and usually cannot be represented by scalable visual variables such as *size*. In contrast, *amounts*, which usually can be mapped to scalable visual variables, signify the quantitative *measurement* used to describe a particular attribute of an object (e.g., the price in dollars of a car).

Arens et al. (Arens et al., 1993) introduce the *continuity* property of quantitative data sets: data sets can be *discrete* or *continuous*. The continuity property can help the system determine the visual representation form. For example, the number of cars sold in each month within a year is a discrete set, which usually cannot be mapped to a smooth curve; in contrast the temperature in degrees over the course of a day is a continuous set, which could be well presented using a smooth curve, but not a discrete diagram such as a bar graph.

Data Relations

Characterization along the dimension of data relations captures various types of relationships among objects. We extend the scope of the data relations (e.g., functional dependency) mentioned in (Roth and Mattis, 1990) by including semantic relations among data such as the *constituent*, *attributive*, and *enumerative* relationships shown in Figure 3.

(*******Insert Figure3 Near Here*******)

The *constituency* relation corresponds to the *has-part* relation and can be further partitioned into physical constituency and conceptual constituency. *Physical constituency* either indicates the physical components of a physical object (e.g., a car has an engine and a gas-tank as its physical constituents) or refers to conceptual components that can be treated like physical ones (e.g., a computer network link's virtual links). *Conceptual constituency*, on the other hand, implies that a concept is conveyed by a set of sub-concepts; for example, a patient's identification information consists of patient's age, gender and medical record number. This distinction plays an important role in the process of generating a comprehensive visual display for the information. In an *attributive* relation, some objects describe certain aspects (attributes) of other objects. Furthermore, these relations can be effectively encoded by the corresponding image's visual attributes, such as its shape, size, or color. *Enumeration* includes both *isa* and *instance* relations (Rich and Knight, 1991). Thus, an enumerative relation shows class inclusion (e.g., pitbull *isa* dog) or class membership (e.g., Spot is an *instance* of dog). Figure 2a uses a label on a button to convey the concept of DRIP (i.e., intravenously administered drugs) received by the patient. In contrast, in Figure 2b, the drips button is associated with a panel that displays all drips items. There is an enumerative relationship between the drip concept and drip items: each item is an *instance* of DRIP. Normally, to convey the information comprehensively, we need to show the generic concept of the classification first (e.g., the button label), then *enumerate* the instances or specific subclasses of the concept in detail.

As described in the above example, the relations usually do not exist independently from each other. More often, one or more types of relations coexist in one piece of information. On the other hand, when the request is to display several relations together, it is necessary to take into account the interconnections between those relations in order to integrate all related information in one presentation.

Attributes of Data Relations. We use a straightforward combination of *dimensionality* (Arens et al., 1993), along with *coverage*, *cardinality*, and *uniqueness* (Roth and Mattis, 1990), to describe the attributes of a relation. In a dynamic application domain, such as network management, the properties of data relations can change over time. For example, normally there are three types of traffic in a particular kind of computer network link (its cardinality value is fixed-multivalued); however, at some times there might be only one or two types of traffic in certain of these links. Such observations lead to more design requirements: either we must be able to predict future changes in advance to make the design generic enough to cope with different expected situations, or we must redesign the visual presentation on-the-fly when it no longer effectively conveys the new information.

Although *data role* and *data sense* could be considered as part of *context information*, they are also directly tied to a specific domain object so we address them here.

Data Role

To generate effective presentations, not only do we need to know the data's *intrinsic properties* (e.g., data type or data attributes), but we also need to identify the information-presentation tasks. Casner (Casner, 1991) discusses how user tasks can affect the presentation design, while Roth and Mattis (Roth and Mattis, 1990) also analyze data properties based on the user's or application's information-seeking goals. We use the *data role* dimension to characterize the functional role each piece of information plays in a visual presentation context. In other words, data role specifies visual tasks that need to achieved. It could be given by a user during interaction, or could be expressed by domain experts during knowledge acquisition.

Based on previous task taxonomies (e.g., Roth and Mattis, 1990; Wehrend and Lewis, 1990), we develop a visual task taxonomy to describe various visual tasks more comprehensively (Zhou and Feiner, 1998). A visual task description actually implies a desired visual relation among different objects. For example, visual task (*******Insert Footnote 2 here*******) Identify<identifier, object> signifies that there should be a visual identification relation between two objects: the identifier and the object to be identified. Each visual task may also be further distinguished by different subtypes of the task, depending on the properties of the domain objects involved. For example, visual task Identify<[Jones-demographics] [Jones-info]> uses Jones's demographics (e.g., mrn and age) to identify other medical information. Visual tasks could help the system to make certain design choices, for example, the system can map the identifier to a prominent feature of the display (e.g., the demographics is the display's heading in Figure 2). Depending on the nature of the task, we store them in the *data role* slot of related objects. For example, an IDENTIFY-TASK is associated with the identifier, and a LOCATE-TASK is stored with its locator:

Task1 of IDENTIFY-TASK	Jones-demographics	**Task2** of LOCATE-TASK	Jones of PATIENT
(identifier [Jones-demographics])	of INFORMATION	(locator [Jones])	(role [**Task2**])
(object [Jones-info])	(role [**Task1**])	(object [Jones-infusion-line])	

Data Sense

To construct an effective visual presentation, the user or application domain visual preference should also be taken into account. For example, one way to distinguish the elements of a small set of objects is to use a different color for each. However, such a presentation would not be effective for a color-blind user. Presentation preferences might also be a function of a specific application domain, perhaps because users in that domain have been trained to use a particular presentation style (e.g., caregivers prefer viewing drip items in a table chart vs. in a bar graph). This notion of visual preference leads to our last characterization dimension, *data sense*, which signifies a preferred way to encode the data visually.

Like data role, data sense can be specified by an end user to indicate his /her individual preference, or captured in the process of knowledge acquisition. Alternatively, data sense might also be inferred, based on other data characteristics. Thus far, we have limited data sense only to the set of five high-level presentation style preferences: *label*, *list*, *plot*, *symbol*, and *portrait*, although it could be extended to include a wide variety of low-level visual preferences (e.g., color or shape). These five senses roughly define five different categories of visual representation forms (Lohse et al., 1994). One way to display information is through a textual *label* (e.g., the patient's name displayed at the top of Figure 2). A label can be enriched by adding graphics, such as underscoring or a button shape, to convey additional information.

In contrast to the label sense, the *list* sense states that a composite object should be displayed in a tabular form in which all its components are listed textually as table entries. As in the *label* sense, both physical and conceptual objects can be represented in the *list* sense. Furthermore, in the *list* sense, all components of a composite object are either listed as an attribute-value pair or as a textual string (e.g., an entry in an indentation chart).

Both *plot* sense and *list* sense indicate that there is a composite object and that all its components need to be represented. Generally, visual displays in plot sense are schematic depictions, which refer to the typical genres of conventional diagrams or graphs. Therefore, in our applications, plot is further distinguished using *structure* (organizing information by their physical structure), *map* (displaying information by a map), and *trace* (organizing information by their temporal order in a time chart). Quantitative information can usually be effectively expressed in *plot* sense. The research of (Mackinlay, 1986) and (Roth and Mattis, 1990) is mostly concerned with how to generate appropriate visual presentations automatically in *plot* sense.

Representing a piece of information in *symbol* sense involves the use of a visual variable or an icon to symbolize the information. For example, a thermometer icon with its scale can be used to symbolize the temperature concept. Both physical and conceptual objects can be represented in symbol sense. We may use a much simplified (abstracted) version of a real image to symbolize a physical object (e.g., using a straight line to symbolize a medical device in Figure 2) or to represent a concept by implying the special connection between the real object and the concept (e.g., the thermometer and the temperature).

Compared to symbol sense, visual representations require much more detail and precision when the information needs to be displayed as a realistic *portrait*. Such precise and detailed visual representation is necessary for people to carry out certain tasks, such as the design of a product's physical appearance.

(*******Insert Figure4 Near Here*******)

We have summarized the presentation-related data taxonomy in Figure 4. To tailor visual presentations to a specific audience or situation, a production system also needs to be equipped with knowledge about the audience (e.g., audience identity or task), the occasion (e.g., the general purpose of the presentation or the time constraint), and the system environment (e.g., the available media and display device). Next, we briefly enumerate this related information and its usage.

3.2 Context Information Modeling

We refer to information about the user, occasion and environment as *context information*. The effectiveness of visual presentations is a function of context information. The more comprehensive the context information is, the better the visual presentations will suit the user or situation it represents. Other researchers have modeled and utilized context information, such as Friedell's situation space (Friedell, 1983) and Mackinlay's display categories (Mackinlay, 1986), to index the knowledge base and facilitate the selection of synthesis operations. However, the modeled context information is usually very limited. As modeling the user or the environment is a sizable task itself, we decided to analyze and model several types of key information. A collection of such information not only directly affects the presentation design, but also models the presentation situation more systematically and comprehensively than any other model does.

Although context information comes from a wide variety of sources, we partition them into three categories: *audience*, *occasion,* and *environment.* These roughly correspond to the information categories that a speaker is expected to know before s/he gives a verbal presentation (Ross, 1997). To make an effective presentation, whether verbal or visual, context information plays an important role in design, as well as delivery.

Audience

The audience is the ultimate judge of a presentation's effectiveness. Knowing the audience better helps to customize the presentation to the audience's interest. Currently, we focus on two types of audience information: *audience identity* (e.g., a nurse or a physician in a medical domain) and *audience preference*. *Audience preference* is further characterized by analyzing audience information-seeking goals, described as *data roles*, and audience visual preference, summarized as *data sense* in Section 3.1.

Occasion

Occasion information specifies the particular circumstances under which the presentation is being given. Currently, we consider *presentation type*, *mood, location*, and *medium*.

Presentation type indicates a specific way of delivering a visual presentation. For example, a presentation may be presented *on-line*, *printed* as a document, or *recorded* on a videotape. Different situations will favor different design decisions. For example, producing a high-quality NTSC video tape requires a different picture aspect ratio, selection of colors, and level of details than those used in making an effective on-line presentation.

Mood describes the nature or the atmosphere of the visual presentation (e.g., *formal* vs. *informal*). Mood information could help the system to choose an appropriate style of illustration. For example, a presentation set for a *formal* mood may have a different set of requirements (e.g., using the Times font vs. the Tekton font) than it does in an *informal* mode. As most of our presentations are generated for formal occasions, we do little in modeling mood information. Nonetheless, we have provided an avenue along which other researchers could go further to extend the situation information model.

Location information indicates where the visual presentation takes place. Like *mood* information, mode information also affects the design in many ways since a different location might put different requirements on the presentation. For example, in a healthcare application domain, a presentation made for doctors at their offices might differ greatly from one made for them in the operating room.

Medium indicates what media are available. For example, if speech needs to be incorporated in the presentation, the visual design process must deal with certain media coordination issues (Dalal et al., 1996b).

Environment

Since the visual presentations always need to be designed on certain computers or presented on certain displays, there is another type of information that specifies the existing physical system environment under which the presentation is being planned and realized. This type of information is referred as *environmental information*. Currently, we have considered two types of environmental information: one is the *platform* on which the whole system runs, the other is the *display* on which the visual presentation appears.

Platform information is tightly coupled with the efficiency of the system. The more powerful the platform is, the more efficiently the system can plan and realize the visual presentation. *Platform information* also further constrains the visual and interactive capabilities of the system. For example, a high-end computer with a fast graphics card capable of hardware texture mapping enables the user to interact with a high-quality presentation, and it allows the system to use texture as a visual cue with little or no sacrifice of interactivity. Platform information naturally could be characterized through its main features: *cpu*, *graphics board*, and *accessories* (e.g., texture mapping board).

While *platform information* affects the efficiency of a generation system, *display information* determines whether the on-screen visual displays are suitable for a particular viewing medium. For example, visual highlighting achieved by placing red text on a yellow background might be fine for a color display, but may be illegible on a black-and-white display. Thus we could use features such as *color*, *resolution*, and *size* to characterize the display.

3.3 Visual Information Characterization

To design appropriate visual presentations for heterogeneous information, the system must be equipped with certain knowledge about the graphic design itself, such as the characteristics of different visual forms and visual techniques. Visual objects are defined based on the syntactic, semantic and pragmatic features of various visual forms, while visual tools are encoded visual techniques, which can assemble and transform visual objects.

Visual Objects

Although numerous visual forms have been carefully analyzed or studied by various researchers in different contexts, most studies (e.g., Winn and Holliday, 1982; Tufte, 1983; Kosslyn, 1989; Tufte, 1990; Keller and Keller, 1993; Tufte, 1997) are limited to qualitative analysis of visual forms based on their communicative capabilities. Qualitative, functional analysis of visual forms is inadequate for automating the visual production process. Visual communication takes place at three levels: syntax, semantics and pragmatics (Goldsmith, 1984), as does visual object synthesis. In other words, we not only need to understand the functional characteristics (*semantics*) of visual objects, we also need to learn their structural features (*syntax*), as well as their communicative capabilities in fulfilling a particular perceptual task for a particular user or application (*pragmatics*). We divide visual objects into five types based on their syntactic, semantic and pragmatic features: *visual primitives*, *visual unities*, *visual structures*, *visual frames*, and *visual discourse*. Visual objects thus form a hierarchy. Visual primitives, at the bottom, are the simplest visual form. Visual discourse, at the top, is the most sophisticated visual form, which comprises a rich semantics and is capable of rendering a comprehensive visual explanation.

(*******Insert Figure5 Near Here*******)

Visual Primitives. *Visual primitives* (Figure 5b) basically are visual variables. There are six types of visual variables: shape, material, texture, size, position, and orientation (Bertin, 1983). These variables are the most basic building blocks that participate in visual object synthesis. Without fixed meaning apart from a visual context, individual visual primitives can hardly communicate any information or be used to fulfill a particular information-seeking goal.

Visual Unities. *Visual unities* are actually the smallest *meaningful* visual units, and are constructed from visual primitives and other visual objects (Figure 5c). For example, a shiny red sphere is an instance of VISUAL-UNITY:

```
ShinyRedSphere of VISUAL-UNITY
    (geometry (Sphere (radius 1.5)))
    (appearance (Material (diffuseColor 1.0 0.0 0.0)
                          (shininess 0.5)))
    (geoProperty (GeoProperty (position 1.0 1.0 1.0)))
```

```
LabeledSphere of VISUAL-UNITY
   (geometry
      (VisualUnity (geoProperty 0.0 0.0 1.1)
                   (Text (string "Sphere")))
      (Sphere (radius 1.0)))
```

Here, geoProperty records the geometric specifications of this visual unity, such as position, rotation, or scale, and appearance stores its visual attributes, such as material and drawing style (e.g., solid vs. wire-frame). All shape information is stored in slot geometry. Note that a visual unity may consist of one or more shape objects, including other visual unities; hence the geometry slot is designed to hold multiple values. For example, the LabeledSphere instance represented above contains two objects in its geometry slot: a textual label that is a visual unity itself at position <0.0 0.0 1.1>, and a sphere at its default position <0.0 0.0 0.0>.

A visual unity must have a *shape*, either geometric shape or text, in order to be recognized as an *object* (e.g., a sphere). Visual unities could be used to convey certain information even though the conveyed information might be ambiguous or incomplete. Nonetheless, the main purpose of a visual unity is to be embedded into a higher level visual object to convey information more comprehensively. For design efficiency, we do not always construct every visual unity from scratch. Instead, we store visual unities representing atomic domain objects in the knowledge base, and use them as building blocks for composing more complex visual displays. We refer to a collection of such pre-stored visual unities as a *visual lexicon* (Zhou and Feiner, 1997a).

To deal with different visual media, visual unities are not just limited to graphical objects. They also include textual documents, static images, and video clips. These premade visual unities can also participate in visual object synthesis. For example, an organization chart can be composed from the images of a company's employees.

Visual Structures. *Visual structures (*Figure 5d) spatially organize various visual unities and other visual structures in a specific way to convey abstract concepts or explain how processes work (Winn, 1987). For example, a 2D line graph that plots height against age can be composed of line segments and two orthogonal axes with textual labels. This plot indicates how height changes with the age. Conventional visual structural classification used in cognitive psychology (Winn and Holliday, 1982) identifies three basic structures that fall between text and realistic images: charts, graphs, and diagrams. Based on the visual representation classification proposed by Lohse (Lohse et al., 1994), we have extended our visual structure classification. For example, graphs are further partitioned into scatter plots, bar graphs, etc. A visual structure could also be a cartogram (i.e., a spatial map overlaid with visual objects, such as dot maps that use symbols to show the location of individual objects), which does not quite fit into any of the three basic categories. To suit our applications' needs, we have studied three visual structures extensively: table charts, structure diagrams, and cartograms (Figure 6); their definitions and usage will be further discussed in Section 4. A visual structure definition not only captures its structural constituents (e.g., cells in a TABLE-CHART, or core in a STRUCTURE-DIAGRAM), but also specifies certain visual parameters such as headingPlacement or verticalAlignment.

(*****Insert Figure6 Near Here*****)

Visual Frames. *Visual frames* contain single or multiple visual structures or visual unities as their frameElements (Figure 5e). For example, a visual frame representing a computer network's topological structure and its traffic statistics over a time interval could include a cartogram (network nodes and links overlaid on a geographical map) and a textual table chart. The cartogram represents the topological structure, and the table chart conveys the traffic information.

If a visual structure is equivalent to a sentence, then a visual frame is the visual counterpart of a paragraph. In addition to orchestrating various visual structures in a single display window, visual frames also position cameras and arrange multiple viewports or windows on a screen. In both VISUAL-UNITY and VISUAL-STRUCTURE representations, the slot holderId indicates which window to use for the current visual object. A visual frame partitions the screen into different windows based on the holder descriptions (e.g., the holder's size and position).

Unlike a textual paragraph, a visual frame can be either static or dynamic. A *static visual frame* remains on the screen from its startTime to its endTime without any change. A *dynamic frame*, on the other hand, is associated with temporal visual transformations. In essence, static visual frames are just collections of static visual objects, and dynamic visual frames include a series of animated key frames, which must be specified through a visual transformation (see below) and a set of visual objects involved in the transformation.

Visual Discourse. *Visual discourse* is the most advanced visual form, which consists of a series of visual frames (Figure 5f). Visual discourse organizes visual frames along a time line: it specifies how long a frame should last and what the transition should be between the frames. Here is an instance of VISUAL-DISCOURSE that consists of two static visual frames:

```
Discourse1                          frame1 of VISUAL-FRAME              frame2 of VISUAL-FRAME
   of VISUAL-DISCOURSE                 (frameElements [element1])          (frameElements [element2])
   (transition CROSS-FADE)             (startTime 0.0)                     (startTime 5.0)
   (frames [frame1] [frame2])          (endTime 5.0)                       (endTime 11.0)
                                    element1 of VISUAL-UNITY            element2 of VISUAL-UNITY
                                       (geometry [Cube])                   (geometry [Sphere])
```

The first visual frame, frame1, containing a cube, will appear on the screen for 5 seconds. Next, frame2, containing a sphere, will replace frame1 and stay on the screen for 6 seconds. The transition from the first frame to the second uses a *cross-fade* technique (Zettl, 1990).

Even though our object-oriented representation of a visual object looks like a template definition, there are substantial differences between them. A template is a *completely* specified parametrized description, of which all parameters are *fully* constrained in terms of their values. Moreover, the semantics of internal relationships among various parameters are usually not explicitly specified or made known to the user. In addition, using a template is quite straightforward: it is instantiated by supplying all its parameters. For example, suppose that we have a bar graph template, in which parameters such as the location of the bar (x), and the height of the bar (y) are specified in terms of their value types and ranges. Using a template is similar to making a function call to retrieve the result by supplying the values of each parameter.

In contrast, a visual object specification in our approach is more like a *partial plan,* whose parameters and the parameter relationships are only *partially* specified. The definition of a visual object is also fully recursive. In other words, the representation merely indicates the structural components of a visual object, but nothing is said about the specific features of its components. In fact, these components may be created from scratch and they could recursively contain very complicated visual objects themselves. For example, a cell within a TABLE-CHART could be a table chart itself, which then could contain an array of complex visual objects. Because a visual object is recursive and may be partially specified, synthesizing a visual object is a planning process rather than just a function call. For example, the task of designing a table chart includes the subtasks of planning the designs of its cells.

To determine unspecified components or attributes of the visual object, various semantic relationships must be explicitly specified for components at different levels. For example, to refine a *complex* table chart, we not only need to know the relationships between its cells, but also need to learn the relationships within each cell. Based on the semantic relationships (e.g., proximity and similarity) among the objects, the system then may determine the cells' alignment strategy and spacing values (e.g., smaller spacing values for objects that have closer relationships). Thus, our system builds visual objects by recursively refining the descriptions of their structural components (Zhou and Feiner, 1997). In other words, a *completed* visual object has specific values for all its key slots (see *optional* slot below), while an unfinished object has at least one unspecified key parameter. Therefore, we use the slot isComplete to indicate whether the current visual object construction is finished (Figure 5a).

Visual Tools

To synthesize and manipulate visual objects, we need a set of tools. Visual tools are encoded visual techniques that can be categorized based on their functions: assembly tools, transformation tools, and camera tools.

Assembly Tools. *Assembly tools* design visual objects from scratch by mapping the properties of information to the visual constituents or attributes of selected visual objects. These tools specify the mappings between a domain object and a visual object at multiple levels of abstraction. For example, *abstract* assembly tools, such as DesignTable, specify abstract structural ingredients of a particular visual object. By "abstract", we mean that a structural mapping only indicates that certain types of visual objects can participate in the construction, but does not specify the details of the expected visual ingredients. For example, DesignTable may indicate that the heading must be an instance of VISUAL-UNITY, but not specify any details of this instance. Here is part of the representation for assembly tool DesignTable:

```
(defclass DesignTable (is-a ASSEMBLY-TOOL)
   (operands (?domain-obj (type DOMAIN-OBJECT)) (?table-chart (type TABLE-CHART)))
   (heading (?heading (type VISUAL-UNITY) (importance OPTIONAL)))
   (decompositions
      (decomposition1
         ;; Retrieve the total number of parts and use assembly tool DesignVisRep to design visRep for each part
         (subactions (:loop (?i 1 (num-of-parts ?operand1))
                           (DesignVisRep <(get-part ?operand1 ?i) ?visualRep[?i]>))))
   ) . . .
```

In this representation, DesignTable encodes ?domain-obj as ?table-chart. To assemble ?table-chart, this tool may divide ?domain-obj into several subparts through one of its methods (decomposition1) and call other assembly tools (DesignVisRep) to synthesize the visual representations for these subparts. The decision to choose a specific decomposition method is based on the characteristics of the input (?domain-obj). If the current ?domain-obj is a composite data object, decomposition1 is usually selected (Zhou and Feiner, 1997b). In this case, the visual representations for the subparts will become the structural constituents (heading or cells) of ?table-chart.

To ensure the legitimacy of a synthesized visual object, a set of constraints must be properly asserted and maintained through the entire design process. Since assembly tools are responsible for instantiating visual objects, certain syntactic constraints can be conveniently expressed along with these tools and asserted with the creation of a new visual object. Within assembly tools, these constraints are usually specified through a set of constrained variables. For example, a syntactic constraint for TABLE-CHART requires that its heading be a VISUAL-UNITY and states that it is an *optional* component. This constraint is specified through a variable ?heading in DesignTable, along with its required *type* and *importance* value (see the representation for DesignTable above).

Transformational Tools. Unlike assembly tools, which create visual objects from scratch, *transformational tools* modify the existing visual objects so that they can be used in new visual presentations. Transformational tools modify visual objects in two ways: *appearance transformations* (e.g., Transfigure changes an object's material) and *geometry transformations* (e.g., Scale modifies an object's size).

Like assembly tools, transformational tools may also be defined at multiple levels of abstraction. The abstractness of transformational tools embodies two forms: Either the visual objects involved are too complex to be modified directly, or the transformational techniques are too abstract to be carried out directly. In either case, the system cannot directly apply the transformational tools to the visual objects before the abstract objects or tools are replaced by more concrete ones. To deal with abstract visual objects, the system needs to divide them into more concrete pieces so that the transformational tools can be applied directly. For example, if we want to Emphasize a complex table chart, we may need to divide the table chart into smaller units so that emphasis can be applied with finer granularity. In contrast, an abstract transformational tool may be replaced by a set of lower-level transformational tools that can be applied easily. For example, the tool Emphasize still cannot be utilized directly even with concrete visual objects. It must invoke more concrete transformational tools such as Highlight (to highligh the object to be emphasized) or Focus (to focus on the object to be emphasized).

Camera Tools. While the first two types of tools deal with domain and visual objects, *camera tools* involve setting or changing a camera (e.g., camera Zoom). Camera tools could also be classified as primitive or abstract. Usually, an abstract camera tool consists of a set of primitive camera tools. For example, Zoom and CameraMove together could form an abstract CameraSearch tool.

Visual Design Rules

Visual design rules guide the automated production process, playing different roles at different levels of design. Generally, visual design rules are used to assure the global *expressiveness* and *effectiveness* of a visual discourse at the highest level of design. Nevertheless, they may also be used to maintain syntactic, semantic, or pragmatic integrity during the synthesis of an individual visual object at a lower level. Under the categorization of expressiveness and effectiveness, we first discuss visual design rules in general. Then we present rule representation in conjunction with the properties of domain and visual information.

Rule Type. Rules for visual design can be divided into two categories: expressiveness rules and effectiveness rules (Mackinlay, 1986). Under this broad classification, rules can be defined more concretely to ensure a particular aspect of an expressiveness or effectiveness criterion. For example, to satisfy a specific expressiveness or effectiveness criterion, the rule could even be further refined at the object level. Such a rule might deal with the layout issues of a particular visual object (e.g., separating the heading from the cells in a table chart design) or handle certain features of the object (e.g., setting the font size of a table heading). Here, we briefly enumerate a few important expressiveness and effectiveness criteria. We hope that our analysis extends previous efforts (e.g., Mackinlay, 1986; Senay and Ignatius, 1994) enough that more accurate and comprehensive visual design guidelines can be formulated and utilized.

Expressiveness rules are concerned with both content and integrity of a visual discourse. First, expressing the proper *content* means that the visual presentation should be capable of conveying *everything* that needs to be conveyed. However, there is more than just expressing whatever the system is told to convey. Expressiveness rules are also concerned with the ability to express the desired information comprehensively, distinctively, generally, and discretely. In regard to the *comprehensiveness* and *distinctiveness* of visual presentations, we refer to the ability to present *all* related information, as well as to emphasize the *unique* attributes of the information. For example, if we are to produce a patient record summary for caregivers, we need to include all related information in the summary, as well as to stress the patient's abnormal conditions. While *generality* regards the ability to express high-level groupings or abstractions of information (Mullet and Sano, 1995), *discreteness* describes the ability to "break" the information into parts so that the user can access the information selectively. *Generality* and *discreteness* together ensure that the information is expressed in a way that allows the user to selectively attend to certain pieces of information, as well as to perceive all the information as a coherent whole. The second important aspect of the expressiveness criteria is the integrity of a presentation. A presentation's *integrity* refers to how well it reflects the true meaning of the information. For example, quantitative information can be expressed by size but *not* by *shape* or *orientation*. Hence, integrity rules ensure that appropriate visual encodings are chosen for different information.

While expressiveness criteria ensure that the desired information is presented correctly, *effectiveness* criteria concern these aspects of a presentation: accuracy, clarity, appropriateness, immediacy, consistency, continuity, and unity. *Accuracy* and *clarity* determine whether the visual displays accurately and clearly convey the information as intended. For example, accurate scales must be used for representing quantitative information and proper contrast must be employed to separate visual objects from each other. Effective visual presentations must not only convey information accurately and clearly, but appropriateness assures that the presentation must be well suited to the task, and the targeted user. For example, displaying textual information to an illiterate person would be inappropriate. Moreover, *immediacy* allows the information to be perceived effortlessly. For example, immediacy criterion could be concerned with establishing the necessary visual groupings and introducing preattentive visual features (Treisman, 1982) in a display. As our goal is to generate coherent visual discourse, the consistency, continuity and unity criteria do their parts to maintain global and local coherence. *Consistency* ensures that visual encodings are consistent within or across displays, *continuity* provides smooth transitions between displays, and *unity* deals with effectively integrating new information into existing displays.

We have attempted to establish a comprehensive categorization of various design rules in an orthogonal manner, but understanding human perceptual capabilities is not an easy task. Therefore, the categories introduced here do not necessarily cover an exhaustive set of rules, nor are they perfectly orthogonal to each other. For example, *clarity* and *immediacy* might both refer to the same rule that uses good contrast to express information grouping clearly and reinforce a perceptual immediacy. Nevertheless, this does not appear to be a problem. Intuitively, it is common that one design rule might be able to achieve more than one expressiveness or effectiveness criterion, just as one visual expression could be used for a number of different tasks.

A general rule for ensuring a specific criterion could be easily formulated (e.g., a rule that specifies using good contrast to separate the foreground from the background to satisfy the immediacy criterion). However, it is also important to establish rules that are specific enough to deal with unique features of various visual objects.

Rule Representation. To efficiently utilize design rules, we must be concerned with both generality and specificity in rule representation. *Generality* refers to the ability to abstract the commonalities for different objects or situations, while *specificity* indicates that distinctive rules may be needed for handling unique situations and giving specific instructions. A general rule representation not only facilitates rule reuse for different applications, but also keeps the rule base from becoming unmanageably large. In contrast, a specific rule representation guarantees that proper actions will be taken for specific situations.

Visual design rules may include domain-specific information at different levels of detail. For example, we may mention the information *domain* (e.g., DRUG or LAB-REPORT) in an expressiveness rule to satisfy the comprehensiveness criterion for different objects; or we may query specific attributes of the domain object in a rule to ensure accuracy. To be able to reuse these rules for different objects in the same application or different applications, we abstract the rules' features by integrating procedural representations. For example, instead of writing different rules for different objects to ensure the comprehensiveness criterion (rule 1 and rule 2), we write one rule but define different *virtual functions* for different domain objects (rule 3):

```
Rule1
   IF: (?obj (is-a DRUG)) & Present(?obj)
   THEN: Present(?obj:name ?obj:dosage ?obj:time)
Rule2
   IF: (?obj (is-a LAB-REPORT)) & Present(?obj)
   THEN: Present(?obj:name ?obj:status)
```

```
Rule3
   IF: (?obj (is-a DOMAIN-OBJECT)) & Present(?obj)
   THEN: Present(components (?obj))

Function components (?obj DRUG) (. . .)
Function components (?obj LAB-REPORT) (. . .)
```

However, to capture distinct features of certain objects and express specific design strategies under different situations, we must formulate different rules. For example, to effectively establish the perceptual hierarchical groupings by proximity to ensure immediacy, the grouping strategy could be very different for a TABLE-CHART (e.g., by constraining spacing values in rule 4) than for a CARTOGRAM (e.g., by creating a cluster in rule 5):

```
Rule4
   IF: (?obj (is-a TABLE-CHART) (spacing ?spacing)) &
       (?cell (is-a TABLE-CHART) (spacing ?subspacing)) &
       Cell(?cell ?obj)
   THEN: ?subspacing < ?spacing
```

```
Rule5
   IF: (?obj (is-a CARTOGRAM)) &
       NodeOf(?obj ?node1) & NodeOf(?obj ?node2) &
       Distance(?node1 ?node2) < threshold
   THEN: CreateCluster(?cluster ?node1 ?node2)
```

Moreover, rules themselves could be classified as abstract or concrete. Abstract rules are defined to cover general situations at a high level, while concrete rules may be tailored to deal with a specific setting (e.g., a special audience or domain task) at a lower level. For example, we have defined two rules to describe how to map a data set onto a visual form:

```
Rule6
   IF: (?obj (is-a DOMAIN-OBJECT)(type COMPOSITE)) &
       Quantitative(components(?obj))
   THEN: Plot(?obj)

Rule7
   IF: (?obj (is-a LAB-REPORT)(type COMPOSITE)) &
       Quantitative(components(?obj))
   THEN: Tabulate(?obj)
```

In this example, rule 6 describes a general principle to present a quantitative data set, while rule 7 specifies a mapping for a particular quantitative set (lab reports) in a medical domain. There are two distinct advantages to establishing rules at different levels of abstraction. First, the general rules are usually domain-independent so that they could be reused for different applications, while the more specific rules could be used to satisfy special requirements. Second, we could resolve potential conflicts using the generality/specificity features of the rules (see below).

If multiple rules are activated simultaneously, then conflicting design decisions may need to be resolved. In the example above, a lab report object actually activates rule 6 and rule 7 at the same time, but the two rules suggest different actions. To resolve conflicts, we order rules by their specificity. In particular, the more specific the rule is, the higher the priority it gets. In our case, rule 7, which deals with only LAB-REPORT, is more specific than rule 6, which may match any instance of DOMAIN-OBJECT. Therefore, rule 7 will override rule 6 to *tabulate* the lab report items instead of *plotting* them. Furthermore, it may be very difficult to tell whether one rule is more specific than another. For example, rule 8 may be modeled for a certain presentation environment, and rule 9 may be targeted for a particular audience, and both may together suggest conflicting actions (e.g., wireframe vs. shaded rendering). To resolve this type of conflict, we need to establish additional rules to order the rules' conditions themselves; for example, complying with the presentation situation may be more important than satisfying the audience's preference (rule 10):

```
Rule8
   IF: (?obj (is-a VISUAL-OBJECT)) &
       Complex(?obj) &
       (?machine (graphics NONE))
   THEN: Wireframe(?obj)

Rule9
   IF: (?obj (is-a VISUAL-OBJECT)) &
       Complex(?obj) &
       (?audience (type NURSE))
   THEN: Solid(?obj)

Rule10
   IF: CapabilityFactor(?machine) &
       PreferenceFactor(?audience)
   THEN: Constraint(?machine) >
         Constraint(?audience)
```

In summary, we establish rules at multiple levels of abstraction based on the specificity of their conditions. Furthermore, conditions themselves could be recursively ordered to determine rule priority. We are able to control the rule base's growth by reusing the general rules, and to resolve conflicts using the specificity of the rule conditions and their recursive ordering.

So far, we have described our approaches in three areas of research: domain information characterization, visual information analysis and classification, and rule formulation and application. Our results in these three areas must work together to produce an effective visual design.

4 IMPROVISE

We are developing IMPROVISE as a proof-of-concept system to demonstrate the generality and comprehensiveness of our research approach. IMPROVISE automatically designs coherent visual discourse for heterogeneous information in interactive environments. To represent heterogeneous information, IMPROVISE supports a wide range of visual techniques and visual media, ranging from individual 2D static images to dynamic 3D animation sequences.

4.1 Overview

Like any other visual production system, IMPROVISE consists of a *knowledge base*, an *inference engine*, a *visual realizer*, and an *interaction handler*.

IMPROVISE's knowledge base contains domain, context, and visual information. Using our data characterization taxonomy, domain information is characterized and stored as *domain objects*. In addition, the knowledge base contains a set of context information about the audience and the environment. Like domain information, visual objects and visual tools are represented in an object-oriented paradigm. In contrast, visual design rules are represented in *production rule* format using CLIPS (JSC-25012, 1993). To efficiently represent and manipulate certain information, we utilize *procedures* to encode knowledge that describes *how* instead of *what*. For example, attribute queries (e.g., how close two objects are) and the details of visual tools (e.g., how to enlarge an object) are captured in procedural formalism.

We have implemented a hierarchical-decomposition partial-order planner (Zhou and Feiner, 1997a) as IMPROVISE's inference engine. Using this planner, IMPROVISE achieves computational efficiency by reducing the amount of search needed compared to systems that use simple search-based algorithms. In addition, a top-down hierarchical decomposition strategy enables IMPROVISE to facilitate the interleaving of planning and execution and to ensure both local and global coherence within a visual discourse.

To realize the visual design produced by IMPROVISE, we have developed an integrated visual language that accommodates both high-level design descriptions and low-level graphics specifications. Because the language is independent of any underlying graphics rendering language and includes common graphics specifications (e.g., geometric transformations), it can be easily interfaced with common graphics rendering packages. For example, our visual language currently interfaces with Open Inventor (Wernecke, 1994).

We have also integrated various interaction metaphors and styles into design rules to produce visual affordances (Gibson, 1977). Thus, IMPROVISE's interaction handler may receive user interactions in the course of a presentation. To provide an interruptible and responsive visual discourse, we are currently working on a reactive planning algorithm to deal with replanning the visual discourse design.

We have used IMPROVISE in two very different application domains. One is a computer network management application that visualizes a network's structure and behavior (Crutcher et al., 1995). The other is a healthcare application that provides a caregiver with a multimedia summary of a patient's medical record (Dalal et al., 1996a).

In the computer network management domain, we use IMPROVISE to help network researchers or network managers understand aspects of network activities (Crutcher et al., 1995). A presentation tool helps these users to establish network service hierarchies, and develop or verify their network management algorithms. In the medical domain, our research focuses on effectively organizing patient information and presenting it coherently as a multimedia summary (Dalal et al., 1996a). Since the multimedia summary contains coordinated text, speech, and visual presentations, constructing a working system is a collaborative effort that involves researchers from several areas including knowledge representation, natural language generation, and graphics generation. Of the various components in the MAGIC multmedia summarization system (Dalal et al., 1996b), IMPROVISE serves as the graphics generator. MAGIC addresses two types of caregivers: nurses and doctors. Based on their needs, the type of information and the style of the summary may vary.

4.2 Example

The example illustrated here comes from MAGIC's medical application domain and involves designing a complex patient record presentation from scratch. The task is to present a patient's information to nurses after the patient's coronary artery bypass graft (CABG) operation. For the sake of clarity, we explain the example in two steps: we first demonstrate how IMPROVISE designs the patient information overview for the nurse (Figure 2a), then we show how IMPROVISE extends the overview to integrate more details about the drips (Figure 2b). Although the finished presentation contains coordinated speech, text, and graphics, we focus here on how IMPROVISE acts as a graphics generator in a collaborative multimedia system.

According to our informal design studies (Dalal et al., 1996a), nurses in our domain prefer that all relevant information be placed near a representation of the patient's physical body and displayed at once (rather than successively revealed). Based on the current communicative intent (Summarize<?patient-info>) and the audience identity (nurse), IMPROVISE formulates a visual goal to Structure all patient information together based on these rules:

```
IF: Summarize<?patient-info> & (?audience (type Nurse))
THEN: IsPhysical(?structure) & Arrange<?info, ?struct>

IF: IsPhysical(?struct) & Arrange<?info, ?struct>
THEN: Structure<?info>
```

To satisfy the goal Structure, IMPROVISE employs an assembly tool DesignStructureDiagram to make a structure diagram, since a structure diagram is capable of conveying a physical structure (the patient's body) and expressing the spatial coordinates of related objects (e.g., medical devices) (Lohse et al., 1994). However, DesignStructureDiagram only produces an empty STRUCTURE-DIAGRAM, and IMPROVISE needs to fill in all the missing parts. According to the definition of a structure diagram (Figure 6b), it has three structural components: a *heading* that is used to identify the rest of the information, a *core* that serves as the spatial locator for other objects, and one or more *elements* that will be placed around the *core*. IMPROVISE needs to transform different parts of the patient information into appropriate visual constituents of the diagram. Based on domain-specific characteristics (e.g., the content of demographics information), IMPROVISE partitions information into different semantic groups and establishes the visual relationships among them. In particular, demographics is used to *identify* other information, and the patient's body is used to *locate* all medical devices. Consequently, IMPROVISE determines the structural ingredients of the diagram based on a set of semantic constraints (Figure 7). The representation of the demographics becomes the *heading* of the diagram to identify it, and the representation of the patient body becomes the *core* of the diagram; the representations for the rest (e.g., infusion lines) are the diagram's *elements*, which will be arranged around the *core* by their spatial coordinates.

(*****Insert Figure7 Here*****)

Furthermore, IMPROVISE encodes demographics into a table chart. This decision is based on the fact that demographics information is a *composite* instance of INFORMATION, which has no specific physical form and by default is better represented in LIST sense. As it does when constructing the structure diagram, IMPROVISE employs the assembly tool DesignTable to first instantiate an abstract table chart, in which each table cell is undefined. Based on the syntactic specifications of TABLE-CHART (Figure 6a), the system refines the abstract table chart by determining its *heading* and populating its *cells*. Based on the grouping information provided by MAGIC's high-level general content planner (Dalal et al., 1996b), IMPROVISE makes three cells at this level: *identification* including *age*, *gender*, and *mrn* in one cell, *medical history* by itself in one, and *surgeon* and *operation* in the third cell to describe the operation.

Each cell to be defined will visualize a composite data object (e.g., cell1 represents three data items). Much as a table chart is selected to represent demographics information, table charts again are used to represent each of the three groups. In other words, each table cell itself in the abstract table chart will be depicted as a table chart. Continuing with the same top-down refinement, each constituent is eventually refined to the point where IMPROVISE needs to determine the representation of each *atomic* objects (e.g., *age*) in the chart.

To avoid constructing every visual presentation from scratch, we store the possible visual representations for *atomic* objects in a visual lexicon (Zhou and Feiner, 1997b). Instantiations of visual lexical items are directly used as building blocks in a design. At this point in our example, each item (e.g., *age* or *gender*) is already atomic, so IMPROVISE searches through the visual lexicon to find appropriate representations for them. In this case, most items are instances of ATTRIBUTE with single or double values (e.g., medhistory has two values), hence IMPROVISE chooses a straightforward representation, which directly lists an object in an attribute-value pair format, with both its attribute name and value represented as simple text strings. Moreover, IMPROVISE decides to use the patient's *name* as the *heading* of the table chart to label the demographics information.

For every medical instrument (e.g., SG with cordis), IMPROVISE needs to express its physical location, its name, and its contents (e.g., DRIPS button) or parameters. Each device is a physical ENTITY and needs to be spatially located; thus it must be represented *symbolically*. Moreover, as the representations of devices are the elements of the diagram, they are constrained to be instances of VISUAL-UNITY by definition. To assemble the VISUAL-UNITY for each medical instrument, IMPROVISE creates an empty VISUAL-UNITY using the assembly tool DesignUnity. Each VISUAL-UNITY is then refined to contain these parts: a line representation, a textual name, and a visual representation for its contents or parameters if they exist. The line representation comes from the stored knowledge that certain medical instruments could be *symbolically* represented using lines. The textual representation of the names is determined by the fact that names are instances of NAME-ATTRIBUTE, which normally appears in LABEL sense. Moreover, a conventional user interface metaphor (a labeled button) is used to illustrate associated contents or parameters of devices. This decision is made based on the information characteristics and the current presentation intent: The contents/parameters are *composite* instances of INFORMATION, and the current goal is only to depict them *qualitatively* without enumerating their details. As an interaction metaphor, the button signifies that it could be pressed later to reveal more details about the contents or parameters at the user or application's request (see below). Figure 8 illustrates IMPROVISE's visual design tree. We have also annotated the automatically generated displays in Figure 9 and Figure 10 to show the relationships among their components.

(******Insert Figure 8 Near Here*****)

Continuing with our example, suppose IMPROVISE needs to convey the details of the drip content, which is currently hidden under the drips button. In our case, the drip content contains a set of drip items, each of which is an instance of DRIP and has attributes such as *name* and *dosage*. Moreover, revealing the details of the drip content means specifying the name and the dosage of each drip item. Considering that both *name* and *dosage* are instances of ATTRIBUTE and need to be listed, IMPROVISE decides to use a table chart to represent all drip items. This decision is similar to that made when IMPROVISE chooses a table chart to encode demographics information, as described above. However, in this case IMPROVISE is more concerned with integrating the new representation into the existing display effectively. Recall that the drip content of Swan-Ganz is currently represented as a button. To establish the perceptual connection between the content summary and its details, IMPROVISE must integrate the newly created table chart as a spatial *extension* of the button. In this case, the table chart is illustrated to resemble a pull-down menu, which is usually considered an extension of a pressed button (Figure 2b).

(*****Insert Figure9, Figure10 Near Here******)

So far, we have described how IMPROVISE constructs a diagram piece-by-piece and fits the pieces together to produce a *conceptually coherent* display. However, an effective presentation must also be *spatially coherent*; for example, ensuring coordinated scales of components and maintaining proper spacings between components. Here, we need to specify the font sizes of the text strings, the spacing between various components, and the scales of geometric shapes (e.g., the sizes of the buttons, and the lengths of the lines in Figure 11). These parameters are determined based on various syntactic, semantic and pragmatic constraints. For example, one constraint states that the spacing between two table cells should be larger than the spacing in the same dimension between the sub-table cells (e.g., spacing *spacing1* should smaller than spacing *spacing2* in Figure 11). Although this constraint appears to be one of the syntactic constraints used in a table chart's construction, it is not hard to justify the semantic rationale behind this constraint: domain objects having closer relationships should be placed closer to each other to satisfy *gestalt* grouping theory (Mullet and Sano, 1995). In the demographics table chart, the object *age* is represented by its attribute name Age and its value 80. The relationship between the attribute name and its own value is considered an *intrarelationship* within a concept, and is closer than any *interrelationship* existing outside of the concept (e.g., between the concepts *age* and *medical history*).

(******Insert Figure11 Near Here*****)

To produce consistent presentations for similar tasks (e.g., presenting a patient's medical overview to a nurse), some constraints, such as the length of the infusion lines and the spacing between various lines, are based on statistical results, which we have obtained through analyzing both domain-specific (e.g., for different patients) and domain-independent graphical representations (e.g., different hand-made structure diagrams). For example, the length of the infusion line is usually at least 1/3 of the diagonal length of the body's bounding box to best illustrate the spatial ratio between the body and the line. There are also other syntactic constraints specified for different visual objects to ensure the clarity and uniformity of the illustration. For example, a syntactic constraint for structure diagrams requires that all elements on the same side of the *core* component (i.e., the body in our case) be aligned with each other (e.g., left alignment in Figure 11) and not overlap with each other (e.g., must be separated by a minimum spacing). These spatial constraints are formulated as numerical constraints in IMPROVISE and are eventually solved by the numerical constraint solver STM (Gleicher, 1994).

5 CONCLUSIONS AND FUTURE WORK

Automated visual production systems should use appropriate visual formalisms to illustrate information. Our goal is to develop systems that can automatically design coherent visual discourse for presenting heterogeneous information in interactive environments. By providing a data characterization taxonomy that builds on previous work, we aim to establish a better understanding of the connections between data properties and visual designs. To better harness various visual forms and techniques, we have classified visual information into visual objects and visual tools based on its role in a visual production process. In addition, we have established a set of visual design principles at different levels of abstraction to guide the visual production process.

To demonstrate the utility and coverage of our approaches, we have introduced IMPROVISE, a prototype system that is being developed to automatically generate visual discourse in two domains. We have used concrete examples that are generated by IMPROVISE to explain our visual production approach.

As we continue enriching our visual object catalog by adding new visual formalisms, we are also refining the syntactic, semantic and pragmatic constraints of our existing visual objects and visual tools. Furthermore, we are integrating a reactive planning algorithm into our current system to produce interruptible and responsive visual discourse, which could be interrupted during the ongoing presentation and respond to user interaction.

ACKNOWLEDGMENTS

This research is 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. We wish to thank Rahmad Dawood for his assistance in realizing the visual language parser, Justin Vallon for implementing part of the planner, and Liqing Xu for developing some of the visual techniques.

REFERENCES

Arens, Y., Hovy, E., and Vossers, M. (1993). The knowledge underlying multimedia presentations. In Maybury, M., editor, *Intelligent Multimedia Interfaces*, chapter 12, pages 280–306. AAAI Press/The MIT Press, Menlo Park, CA.

Bertin, J. (1983). *Semiology of Graphics*. Univ. of Wisconsin Press, Madison, WI. (trans. by W.J. Berg).

Casner, S. (1991). A task-analytic approach to the automated design of graphic presentations. *ACM Trans. on Graphics*, 10(2):111–151.

Crutcher, L., Lazar, A., Feiner, S., and Zhou, M. (1995). Managing networks through a virtual world. *IEEE Parallel and Distributed Technology*, 3(2):4–13.

Dalal, M., Feiner, S., McKeown, K., Jordan, D., Allen, B., and alSafadi, Y. (1996a). MAGIC: An experimental system for generating multimedia briefings about post-bypass patient status. In *Proc. 1996 AMIA Annual Fall Symp*, pages 684–688, Washington, DC.

Dalal, M., Feiner, S., McKeown, K., Pan, S., Zhou, M., Hoellerer, T., Shaw, J., Feng, Y., and Fromer, J. (1996b). Negotiation for automated generation of temporal multimedia presentations. In *Proc. ACM Multimedia '96*, pages 55–64, Boston, MA.

Feiner, S. (1985). APEX: An experiment in the automated creation of pictorial explanations. *IEEE Computer Graphics and Applications*, 5(11):29–37.

Friedell, M. (1983). *Automatic Graphics Environment Synthesis*. PhD thesis, Case Western Reserve University.

Gibson, J. (1977). Notes on direct perception and indirect apprehension. In Reed, E. and Jones, R., editors, *Reasons for Realism: Selected Essays of James J. Gibson*, pages 289–293. Lawrence Erlbaum Associates, Hillsdale, NJ.

Gleicher, M. (1994). *A Differential Approach to Graphical Interaction*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891.

Goldsmith, E. (1984). *Research into Illustration: An Approach and a Review*. Cambridge University Press, Cambridge.

JSC-25012 (1993). *CLIPS Reference Manual*. Software Technology Branch, Lyndon B. Johnson Space Center. CLIPS Version 6.0, JSC-25012.

Keller, P. R. and Keller, M. M. (1993). *Visual Cues: Practical Data Visualization*. IEEE Computer Society Press and IEEE Press.

Kosslyn, S. (1989). Understanding charts and graphs. *Applied Cognitive Psychology*, 3:185–226.

KRSL (1993). *Knowledge Representation Specification Language Reference Manual*. DARPA/Rome Laboratory Planning and Scheduling Initiative Knowledge Representation and Architecture Issue Working Group. Version 2.0.2.

Lohse, G., Biolsi, K., and Rueter, H. (1994). A classification of visual representations. *Communications of the ACM*, 37(12):36–49.

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

Mullet, K. and Sano, D. (1995). *Designing Visual Interfaces*. SunSoft Press, Mountain View, CA.

Rich, E. and Knight, K. (1991). *Artificial Intelligence*. McGraw-Hill, Inc., second edition.

Robertson, P. (1991). A methodology for choosing data representations. *IEEE Computer Graphics and Applications*, 11(3):56–67.

Ross, R. (1997). *The speechmaking process*. Allyn and Bacon, Boston, MA, 11th edition.

Roth, S. F. and Mattis, J. (1990). Data characterization for intelligent graphics presentation. In *Proc. CHI '90*, pages 193–200, New Orleans, LA. ACM/SIGCHI.

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

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

Seligmann, D. (1993). *A Visual Language for Automated 3D Graphics Generation*. PhD thesis, Department of Computer Science, Columbia University, New York, NY.

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

Senay, H. and Ignatius, E. (1994). A knowledge-based system for visualization design. *IEEE Computer Graphics and Applications*, 14(6):36–47.

Treisman, A. (1982). Perceptual grouping and attention in visual search for features and for objects. *J. of Experimental Psychology: Human Perception and Performance*, 8(2):194–214.

Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT.

Tufte, E. R. (1990). *Envisioning Information*. Graphics Press, Cheshire, CT.

Tufte, E. R. (1997). *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, Cheshire, CT.

Wehrend, R. and Lewis, C. (1990). A problem-oriented classification of visualization techniques. In *Proc. Visualization '90*, pages 139–143. IEEE, Los Alamitos, CA.

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

Winn, W. (1987). Charts, graphs, and diagrams in educational materials. In Willows, D. and Houghton, H., editors, *The Psychology of Illustration: Basic Research*, volume 1, chapter 5, pages 152–198. Springer-Verlag, New York.

Winn, W. and Holliday, W. (1982). Design principles for diagrams and charts. In Jonassen, D., editor, *The Technology of Text*, volume 1, pages 277–299. Educational Technology Publications, Englewood Cliffs, NJ.

Zettl, H. (1990). *Sight Sound Motion: Applied Media Aesthetics*. Wadsworth Publishing Company, Belmont, CA, second edition.

Zhou, M. and Feiner, S. (1997a). The representation and use of a visual lexicon for automated graphics generation. In *Proc. IJCAI '97*, pages 1056–1062, Nagoya, Japan.

Zhou, M. and Feiner, S. (1997b). Top-down hierarchical planning of coherent visual discourse. In *Proc. IUI '97*, pages 129–136, Orlando, FL.

Zhou, M. and Feiner, S. (1998). Visual task characterization for automated visual discourse synthesis. In *Proc. ACM CHI '98*, Los Angeles, CA.