Exercises for CS3650 Visual Languages and Programming

Exercise 1:

The Sentient Map is a novel visual user interface to be investigated and developed during this course. It combines the strong features of the map, the browser, and the index, based upon the concept of visual languages and active index. It will also serve as the focus point of term projects for this course.

The purpose of this exercise is to get a handle on the different type of maps, their personailties, their visual appearances and their teleactivities.

Each sentient map has a type, a personality, a visual appearance, and a set of teleactivities. The first part of this exercise is an interactive exercise. You can contribute different map types into a depository. For each map type, you will provide the following information:

(1) name of the map type,
(2) Creator's purpose for this map type,
(3) Viewer's purpose for this map type,
(4) A map example which is an URL linked to the map instance,
(5) Date of submission,
(6) Name of submitting person (i.e. you).

The second part of this exercise is to describe the external appearancees of teleaction objects of a map using visual grammars or multidimensional grammars. The teleaction objects are usually overlaid on some background. In fact, a sentient map can be defined recursively, as (a) a collection of teleaction objects including a background object, or (b) a composition of several sentient maps.

You can select a map type (such as the one introduced by you), and define those teleaction objects of interest (to whom?) using visual grammars.

Exercise 2:

(a) Analyze either the home page of S. K. Chang (http://www.cs.pitt.edu/~chang) or a web page selected by you (such as the sentient map) and express it as a generalized 2D string. The primitives of this home page consists of: "link", "text_block" and "image". The analysis can be done manually. (b) Design a visual grammar to generate acceptable html page layouts as visual sentences. (c) Design **on paper** a parser to parse a visual sentence to recognize it as an acceptable html page layout.

(Possible extension: You may want to refine the approach to develop techniques to design page layouts as well as to parse page layouts.)

Exercise 3:

(a) The formal icon system G for visual queries is:
r1 T ::= (Am, Ai, nil)
r2 T ::= ({VERm, T, A}, {VERi, T, A}, eval-and)
r3 E ::= (Tm, Ti, nil)
r4 E ::= ({HORm, E, T}, {HORi, E, T}, eval-or)
r5 Q ::= ({HORm, DB, E}, (HORi, DB, E}, eval-query}

The underlying context-free grammar is
r1 T ::= A
r2 T ::= T ^ A
r3 E ::= T
r4 E ::= E + T
r5 Q ::= DB + E

The elementary icons are the atoms (A) and the database icon (DB). Suppose the visual query is:

```
DB   A1    A2
           A3
```

where DB has attribute "DB_name=Employee", A1 has attribute "Salary=?", A2 has attribute "Sex= 'Female'", and A3 has attribute "Status='Single'".
The corresponding SQL query is:

```
SELECT Salary
FROM Employee
WHERE Sex = 'Female' and Status='Single'
```

Construct the parse tree and illustrate how the translation to the corresponding SQL query can be done, using the appropriate algorithms for eval-and, eval-or and eval-query. Do the same for the following query:

DB A1 A2 A3

(Hint: At each node, you can keep a string of the form C;T where C is the conditional predicate, and T is the target attributes set. Naturally, either C or T can be null. Suppose there are two nodes N1 with string C1;T1 and N2 with string C2;T2. If the parent node should do eval-or, then eval-or combines the strings as (C1 or C2); T1, T2. If the parent node should do eval-and, then eval-and combines the strings as (C1 and C2); T1, T2. Finally, the eval-query should rearrange the string into the SQL syntax.)

(b) Construct the conceptual graph for the parse tree, using conceptual-graph construction rules, similar to those introduced in the class.

(c) Suppose the input is the home page or a document similar to the home page, as in Exercise 2. With a suitable iconic system (and the corresponding grammar), by analyzing the home page we can obtain the parse tree. The objective is to perform parsing on home pages so that we can extract the spatial indexing information from the parse tree. An example of spatial indexing information is a news flash, which is a text box with certain special characteristics such as being adjacent to a blinking icon, or being beneath the banner (you define the spatial relationships). Another example is a list of links under a major heading, such as the courses recently taught by Professor Smith. Design a few (at least one) semantic operators such as VERm, HORm of the iconic operators VER = (VERm, VERi), HOR = (HORm, HORi), so that this kind of spatial indexing information can be extracted from the home page of interest. (Hint: Please reread the section on iconic operators in Chapter 1.)

Exercise 4:

(a) Suppose the input is the home page or a document similar to the home page, as in Exercise 2. By analyzing the home page we can obtain the generalized 2D string. The objective is to perform spatial reasoning on home pages, so that we can answer queries such as: (Q1) find home pages containing a logo image of some company (any company) on the left-hand or right-hand side; (Q2) find home pages containing information about location of hotels in Pittsburgh area, either directly (such information already in home page) or indirectly (such information linked to home page); (Q3) find home pages containing information about social events between September 1, 1999 and October 31, 1999. Using generalized 2D string as the knowledge structure, demonstrate how you would process the documents to answer the above queries. In addition to, or in lieu of, the generalized 2D string, what knowledge structures you would maintain to do spatial reasoning?

(b) The purpose of this exercise is to experiment with the visualization of the sentient map. A sentient map for multimedia information retrieval has three modules: 1) The visual query processor allows the user to formulate a visual query. It parses the visual query and translates it into the underlying query language such as the SigmaQL. 2) The visualizer provides the visualization of the underlying multimedia database and displays visual feedback to the user. 3) The search engine searches the underlying multimedia database and sends results back to the visualizer.

In this exercise you will deal with the visualization module of the sentient map. In the next exercise you will add the actions to the sentient map. Professor Tsuhan Chen of CMU implemented the NetIce system for teleconferencing on the Internet. A 3D room is displayed, with avatars representing teleconferencing participants. This system will be even more useful, if the avatars can point at sentient maps displayed on the wall to perform spatial reasoning. The Netice program you can download is called MyViewH.cpp, available from this web site. For this exercise, all you have to do is to replace a window or a door by your map. Then you can manually display another map. You should have several maps ready for visualization, so that later you can use them in the term project (if you choose Project I). The maps you will display must be thematically related. You can use my web site as an example.

Exercise 5:

The purpose of this exercise is to add the actions to a sentient map. The tool we will use is the Active Index Cell Manager or IC Manager. Using this tool, a click on the sentient map becomes a message sent to the IC Manager, which then performs the appropriate action or actions. In this exercise you can assume the user's click is already captured and processed, and a message RETRIEVE is sent to the IC Manager. IC Manager then invokes a program called SearchEngine(), which is a C function that displays a message "Search Engine is invoked". In this exercise, the SearchEngine does nothing. For Project I, the RETRIEVE message will have parameters associated with it, for example, one important parameter being the SigmaQL query string. The SearchEngine searches the multimedia database and returns the retrieved objects to the visualizer.

Project Suggestions:

The project can be selected by the student, in consultation with the instructor. It is reasonable to expect the student will spend around forty (40) hours to do the project.

The project should be centered on the concept of the **Growing Book**, which will use **sentient map** as the visual interface, and how to design visual information systems with the help of visual languages and multidimensional languages such as the SigmaQL language.

Common Part of all Projects: Any one selecting the Growing Book project will be assigned a chapter to annotate by adding the < ts > tags. The more detailed the < ts > tags, the better. Any one selecting the SigmaQL project should implement a SigmaQL query translator to translate a SigmaQL query into retrieval requests.

Project I: Design the visual interface for the Growing Book, which is a multimedia database. The visual interface consists of the visual query processor and the visualizer. If you are a two-person team, you will design both the visual interface and the search engine. If you are working alone, you will only design the visual interface. (I say "design", because you don't have time to implement an entire system. Therefore, your design is a mockup, consisting of both real codes and demo examples.)

Project II: Investigate the transformation algorithm to transform a Sigma query with respect to a data model (which could be a set of

multimedia specification schemas MSSs) that describes the Growing Book. There is an online presentation on the Sigma query you can study. The related paper is: S. K. Chang, G. Costagliola, E. Jungert, "Querying Multimedia Data Sources and Databases", 1999. (The Sigma query is also included in the presentation on information fusion.)

Project III: Design and implement (a part of the) gesture interface for the Growing Book.