

# COMS W4735x: Visual Interfaces to Computers

Fall, 1999

## Assignment 2: Due November 18, 1999

### Course Homework Structure

This assignment is worth 20% of your course grade. It is due on 11/18. The final paper of 15 pages, or the final project with at least a 5 page write-up, is worth 50% of your course grade and is still due 12/14. There still are no exams.

### Description of Spatial Relations: Design Specifications

The goal of this assignment is to write a program that describes the location of a "visitor" to the "Columbia campus". More accurately, you are to modify the standard image-viewing program "xv" so that it displays a binary image of the main campus as seen from above, and which responds to any mouse click within it with an English description of where the visitor would be. For example, it could print out "Inside Low" or "Between Schapiro and Uris" or "South of Dodge and North of Journalism".

Much of the code will be found on the web page and/or in the class account, as are the required images, and other aids to the construction of the program. Your main job is to encode the buildings' shapes, to determine their spatial relationships with the (x,y) coordinate that the mouse click will give you, and to filter out any relationships that are unnecessary because they can be easily inferred. For example, if the visitor is "South of Uris", it is not necessary to also say "South of Schapiro", as this can be inferred from the relation that "Uris is South of Schapiro".

To do this assignment, you will probably need an account in the CLIC lab, and you will probably need to do your programming in C. However, the data for the assignment are portable (they are ".pgm" files), and it is not necessary to use "xv" as the base for the point-and-click interface, if you can create your own point-and-click interface or modify any existing one. If you do not use the code and other materials provided, it is your obligation to make your approach understandable to the grading staff.

To help structure the assignment, it is broken down into four steps with equal credit, with the full assignment worth the 20 points toward your final grade.

#### 1 (For 5 points): Basic infrastructure.

Sitting in the class account is a compressed software package consisting of a modified version of the code for "xv", a file that you will be modifying, a Makefile to generate your own custom version of "xv", some utility files, and three data files. This package is "~cs4735/xv-zips/xv-3.10a-cs4735.tar.gz". Unzip it, untar it, and type "make"; you should get a version of "xv" that can be executed using the command "xv-cs4735 campus.pgm labeled.pgm". The title bar of this version should read "xv 3.10a (Columbia University, CS4735) <unregistered>". Left clicking on the campus map displayed by this program will print to the standard output (e.g. to the window that you typed the command) the x and y coordinates of the mouse location in the image. All this has been tested on the CLIC machines, but it should also run on any other Sun Unix environment supporting C and X-windows.

In more detail, the following program files were modified for the assignment: "Makefile", "xv.h", "xv.c", "xvevent.c", where the last is the event handler which detects the mouse click and then calls the function "do\_stuff" to process whatever the mouse click's x and y coordinates mean. None of these files need to

be changed, but if you want to get tricky, you should look carefully at the way in which "do\_stuff" is called and used.

The following program files were added for the assignment: "pgm\_helper.h", "pgm\_helper.c", "hw2.h", "hw2.c", where the last is the only code that needs to be modified, namely, the code for "do\_stuff". Probably you will want to construct here whatever data structures are to be built from the imagery, as well as the code to get building shapes, to calculate and store building relationships, and to do the spatial filtering.

The following data files were created for the assignment: "campus.pgm", "labeled.pgm", "table.txt". The first file is a binary image of the main campus as seen from above, where ones represent the buildings and zeros represent the space between them. The second file is an integer-valued image based on the first, in which each building is given a code integer, and all the pixels belonging to the same building are encoded with the same integer; zero still means empty space. The third file is a text file which translates the code integer into a string, so that the answer can come out in English.

Now, having compiled and verified that the modified "xv" works, change the code so that it prints out, for each non-zero code integer that is clicked on, the following building shape features: the (x,y) of the center of mass, the area, and the upper left and lower right coordinates of its minimum bounding rectangle (MBR), and the English string of its name. That is, the output to the standard output should be in the form: x click, y click, code integer, x center, y center, area, x upper left, y upper left, x lower right, y lower right, name. The file "labeled.pgm" has been created explicitly to make these tasks easy; to get the shape features for building N, one only has to scan the image for the occurrences of the code integer N. Finding the center of mass and the area for building N is identical to what was required in the first assignment. It is not too hard to find the minimum and maximum x and y values occupied by building N, either; this gives the MBR.

Test the program by clicking on buildings, and by clicking on empty space. Turn in your code and output, unless you proceed to the next step.

## **2 (For 5 points): Create simple spatial relations.**

Now, design and code the boolean-valued functions for In(x,y,N), North(x,y,N), South(x,y,N), East(x,y,N), and West(x,y,N). Each function takes the coordinates of the mouse click (x,y), and the building code integer N, and returns true if the "visitor" is in the given relationship to the "building". Note that "In" is particularly simple, and that the other four are symmetric with respect to each other. But you have to decide how best to define those four without reference to any specific building, just to their generic shape features. It is probably more humanly accurate to have North be more complicated than simply " $y > \text{miny}[N]$ ". Now, for every mouse click, print out all spatial relationships that apply to (x,y), using the name of the relationship and English string of the building. This should generate on the order of about as many descriptions as there are buildings, depending on your definitions. Note that the order in which you produce the descriptions should not be arbitrary, but should be sorted in some intelligent way.

Test the program, and make sure you have documented your design decisions for the definitions of the relationships and the way you have decided to order them. Turn in some examples of the use of step, unless you proceed to the following one.

**3 (For 5 points): Filter spatial relationships.**

Now, represent all the possible spatial relationships of the mouse click to the buildings and the buildings to each other, and filter the relationships using their transitivity. For example, if the visitor is North of Low, and it is known from the map itself that Low is North of Butler, then it is not necessary to say that the visitor is North of Butler; this can be inferred. This filtering can be done by representing all the inter-building relationships as a graph or graphs even before the first mouse click, by then adding in the visitor relationships to these graphs once they become known, and then by detecting by the appropriate search algorithms those relationships that cannot be inferred and, therefore, must be output in English. You must handle the relationship In, as well as the four compass directions, and whatever is left must come out in some reasonable order (as in the prior step).

Make sure your graph representations and algorithms are clearly explained. Turn in multiple examples of this (note that some visitor positions will legitimately be East or West of several buildings even after filtering), unless you proceed to the next step. See if you can find the most narrowly specified description (i.e., the place with the smallest total area that is uniquely described), and the most widely described one. You must turn in something for this step, even if you proceed.

**4 (For 5 points): Creative step.**

To get credit for this step, you must add at least one more filterable trinary spatial relationship to the mix, and filter it. For example, you can do Between (which is static, involving one visitor position and two buildings). Or do MovingToward (which is dynamic, involving the visitor's present location, the visitor's prior location, and one building), or MovingAlongSideOF (which is similar). Make sure you document your reasoning, particularly if you choose something other than the two suggested above.

Turn in several examples of these, including some in which it more than one trinary relationship survives the filtering.

And, as a general rule, whatever you do in code or in writeup, style counts. It is your obligation to write it all up so that the instructor and/or the TAs can understand it on the very first try.