NEW PROGRAMS

A technique for identifying atoms from a screen image

D.A. Kuznetsov and R.A. Abagyan

European Molecular Biology Laboratory, Heidelberg, Germany

Improving the interfaces in molecular graphics applications, making them more natural and easy to use, is an important task, given the current complexity of the displayed objects and of modeling operations. Clicking near an atom center is the usual method of atom selection. However, this method has certain disadvantages when working with images composed of different atomic representations such as sticks, CPK, or dotted surfaces. We propose another technique allowing the user to obtain the correct answer when he or she clicks on any element of the atom image.

Keywords: molecular modeling, user interface

INTRODUCTION

With the current plethora of molecular graphics representations, it is now relatively straightforward to display the global properties of a molecule in many different ways. Given that the human brain is a powerful instrument of images recognition, the user can make an instant global assessment of the image of a molecule displayed on the screen. However, sometimes he or she needs to identify a certain part of the molecule (atom, residue, or domain) more exactly. The usual mechanism for doing this is *atom picking*. Generally it works in two steps: having selected with the cursor a point on the screen displaying the image, the user gives the instruction to the program to find the corresponding atom by a certain rule. The most simple rule is the best proximity of the *XY*-coordinate of the atom to the *XY*-coordinates (converted to real space) of the point selected on screen. This is the so-called coordinate-picking algorithm and most programs for molecular modeling[1-3] use this rule. (SYBYL has the possibility to define an atom covered by a CPK sphere by clicking to any pixel of the sphere. However, this algorithm is not applied to sticks and lines.) This method works reliably only in cases where the user can estimate the position of the center of the chosen atom. Such estimation is straightforward only if the molecule is represented by a calligraphic model or is covered by space-filling sticks. However, this method has two major disadvantages. First, it is difficult to use it when the molecule is represented by solid CPK spheres and, therefore, atom centers are hidden. The problem here is that in cases where the user clicks the atom center incorrectly, instead of no answer, the program sometimes reports the identifier of another atom, the center of which fell into the picking region instead of the intended one. The second drawback is that this method requires the user to click upon an atom center exactly, although a user usually understands the entity "atom" as the origin of covalent bonds together with adjacent halves of these bonds. These limitations prompted us to develop a method free of these disadvantages. The method was implemented with the usage of GL (the Graphic Library developed for the family of graphics workstation manufactured by Silicon Graphics) and was tested on Iris VGX and Indigo workstations.

Address reprint requests to Dr. Kuznetsov at European Molecular Biology Laboratory, Postfach 10.2209, Meyerhofstrasse 1, W-6900 Heidelberg, Germany.

Received 1 February 1993; accepted 26 April 1993

DESCRIPTION

The technique is built around calls to a GL procedure[4] which can identify parts of a displayed image according to preloaded identifiers (names). The procedure considers normally only a part of the image (more exactly, a part of the space). In fact, in GL there are two variants of this procedure, pick and gselect, which differ in how this part of the space is specified.

Pick uses the built-in specification of a picking region. The region includes the small vicinity of a clicked point in the *XY*-plane which is extended between clipping planes in the *Z*-direction. So, pick takes items of the image regardless of the *Z*-coordinate. Gselect gives one freedom to specify the picking region explicitly. According to the usage of either function, two variants of the technique were developed.

Variant I

As mentioned above, pick identifies parts of the image on the basis of numerical names assigned to these parts. Constructing an image of a molecule on a per-atom basis, and keeping primitives related to an atom in correspond-

ing objects, we have a natural way to use the atomic number as name. So, definition of an object containing a form of atomic representation (a CPK sphere, a stick, or a surface patch) would be the following:

⟨start of object⟩
⟨load name(number_of_atom)⟩
⟨color⟩
⟨primitives (lines, polygons...)⟩
⟨end of object⟩

Therefore, different types of atomic representations included in one or several objects can be associated with one atom if all the objects carry the number of the atom loaded as name. An image, composed of atomic representations with the structure described above, could then be a target for the pick. Having analyzed a clicked point of the image, pick returns in the provided buffer a list with the following structure:

$$\{1, N_1, 1, N_2, 1, N_3, 1, N_4, \ldots 1, N_n\} \quad (1)$$

where N_i are the numbers of the atoms possessing primitives, some parts of which have appeared in the vicinity of the clicked point. The atom from the list $\{N_1, N_2, N_3, N_4, \ldots N_n\}$ with the maximal Z-coordinate is the clicked atom.

Although the presented technique works well for a wire-frame model, a precalculated dotted surface with excluded hidden points and sticks composed of solid polygons, it does not work properly for CPK spheres. The reason is that pick analyzes the presence of drawings regardless of Z-coordinate. Therefore, it detects pixels of spheres which are obscured by other spheres, whereas the atom to which the element of the hidden surface belongs may well have a higher Z-coordinate. Such a case is shown in Figure 1. Correct selection of the clicked atom from the list $\{N_1, N_2, \ldots N_n\}$ formed in this case cannot be done solely on the basis of the Z-coordinate of atom N_1. In other words, the Z-order of surface elements does not necessarily correspond to the order of Z-coordinates of atomic centers, and therefore usage of the latter may lead to a situation where the wrong atom is reported instead of the clicked one. Thus, a correction of the algorithm is required, which is done in the variant described below.

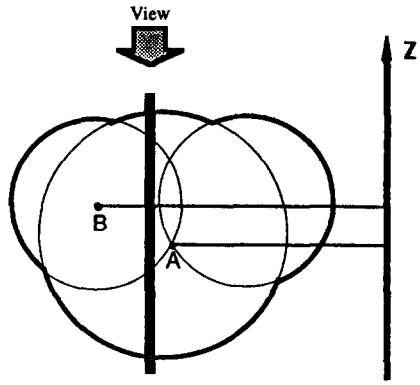

Figure 1. Misrecognition: instead of sphere A, *sphere* B *will be reported. Both spheres are picked, but the center of the sphere* B *has the higher* Z*-coordinate. The vertical bar intersecting both spheres represents the picking region.*

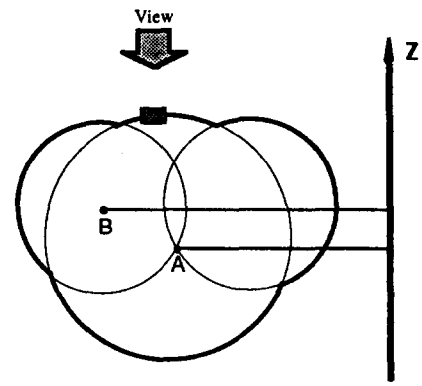

Figure 2. Selection of the area of space (approximated above by the filled rectangle) to be subjected to the test by gselect. *This area should be small enough to allow one to have within it only pixels of the clicked sphere. Because of such a specification of the space region, the sphere identifier can be unambiguously evaluated independently of the* Z*-coordinate of the center of the sphere.*

Variant II

This method uses the criterion of visibility of drawn elements which are close to the clicked point to select the atom to which these drawings belong, i.e., the atom which has been clicked. The method is built around the call to gselect. The gselect call requires the explicit specification of the part of space where it should detect the presence of drawings. This feature of gselect allows selection of a box in the space containing only the clicked pixel with a few pixels around it (Figure 2). Such selection of a box allows the consideration of only the pixels which belong to the atom chosen by the user, and therefore avoids the necessity of the final selection from the list of atoms. The calculation of the XY-coordinate of the element of the drawing which has been mapped to the clicked pixel is obvious and can be carried out by reversed projection, viewing and modeling transformations. The calculation of the Z-coordinate can be done on the basis of visibility of the pixel. Visibility of a pixel means that its Z-coordinate, mapped to the integer range of the Z-buffer, is kept in the Z-buffer. Because of the orthographic projection used,[4] the relation of the Z-coordinate with the corresponding stored Z-value[5] may be described by the following equation:

$$(Z_z - Z_{min})/(Z_{max} - Z_{min}) = (Z - Z_{near})/(Z_{far} - Z_{near}) \quad (2)$$

where Z_{min} and Z_{max} are the minimal and the maximal Z-values corresponding to the Z_{near} and Z_{far} clipping planes, and Z_z and Z are the Z-value and the required Z-coordinate of the selected pixel.

In the developed technique, the clicked pixel is taken with a one-pixel border around it to make the algorithm less sensitive to operator error. Using lrectread, with previous settings by readsource to read values from the Z-buffer, one can obtain the Z-values of all 9 pixels. The pixel closest to the viewer is the one with the lowest Z-value (in GL the Z-buffer increases in depth), thus it should be selected from the set of all 9 pixels as the pertinent one. The conversion of the selected Z-value to the Z-coordinate on the basis of Equation (2) is straightforward.

A pretreatment of Z-values is required if one intends to use the technique on machines with differing precision of the Z-buffer, i.e, with either 24-

bit hardware Z-buffers or 32-bit software Z-buffers (e.g., on Iris VGX or Iris Indigo workstations, respectively). The problem is that negative Z-values obtained from a 24-bit Z-buffer become misinterpreted as soon as they fall within the data flow. The reason is that bit patterns of negative values in the 24-bit arithmetic represent large positive values in the 32-bit arithmetic. Such Z-values should be used only after preliminary treatment to convert them to the conventional size data representations. This can be done by bitwise addition of the value 0xFF000000 (called an extender) to the negative Z-values obtained from a 24-bit Z-buffer. The algorithm which implements this idea is described below.

The extender can be constructed from the bit pattern representing the value -1, which can be composed from the value Z_{max} provided by GL. Taken as a complement to a 32-bit unsigned variable, the bit pattern of the value -1 in an unknown arithmetic (either 24- or 32-bit) becomes either the extender for negative values, in the 24-bit arithmetic, or 0, in 32-bit arithmetic. The obtained extender should be added bitwise to the negative Z-values. Filtering out all positive values, one can carry out this process by masking through the value which is complementary to Z_{max}. The following fragments of C source text illustrate this algorithm:

```
unsigned long z_max, extendr, mask, z_array[9];
int i;
...
extendr = 2*z_max + 1; /* It is -1, 0xFFFFFFFF for 32-bit
                                and    0xFFFFFF for 24-bit */
extendr = !extendr;            /* It is          0 for 32-bit
                                and  0xFF000000 for 24-bit */
mask = !z_max;
...
if (z_array[i] & mask)
   z_array[i] |= extendr;
```

After completion of these procedures, one obtains the *XYZ*-coordinate of the selected pixel. As mentioned above, gselect requires explicit specification of the area where it detects presence of drawings. Having specified by ortho the box vicinity in the object space around the found point as the area to control one can invoke gselect. With such a small area, gselect instead of returning a long list with Structure (1) normally returns only the first two elements of the above list, namely $\{1, N_1\}$. The second element is the required atom number.

PROBLEMS

Neither variant of the technique is perfect. The disadvantage of the first variant has been described. The second variant also has a problem, albeit a minor one. It comes up when the user tries to click a primitive which is occluded by drawings without preloaded names such as characters represented by raster fonts. The characters may belong to atomic labels or to any other text information added to the image. This problem occurs because in GL a font glyph itself is invisible for either pick or gselect and thus can not carry any information about an atom to which it belongs (if it is part of an atomic label) but it does make a trace in the Z-buffer. However, as the second variant is designed mostly to work with CPK spheres, atom (residue) labels are usually buried under spheres and therefore in the Z-buffer; traces of labels in such cases are overwritten by traces of spheres. Because of the reason mentioned above, the presence of labels does not cause a malfunction of the algorithm. In rare cases of trials to click an atom through characters which do not belong to it (through image banners, etc.) the algorithm reports no answer instead of a "false positive" answer.

REFERENCES

1 Insight II, version 2.1.0 (March 1992), Biosym Technologies, 9685 Scranton Road, San Diego, CA 92121-2777

2 Quanta, version 3.3, Molecular Simulations Inc., 200 Fifth Avenue, Waltham, MA 02154

3 SYBYL, version 6.0 (November 1992), TRIPOS Associations, Inc., 1699 S. Hanley Road, Suite 303, St. Louis, MO 63144-2913

4 Graphics Library Programming Guide (007-1210-040), Silicon Graphics, Inc., 2011 N. Shoreline Blvd., Mountain View, CA 94039-7311

5 The Z-value of a pixel is a numerical value from the Z-buffer corresponding to the pixel. The Z-coordinate of a pixel is the Z-coordinate in real space of the element of the drawing which was mapped into the pixel.