Different methods for interpreting mouse gestures
Kyle Erickson
8/23/2004

Bounding box

The first method I will discuss is the bounding box method. This involves drawing a box around the gesture and dividing it up into a grid. The gesture is then defined by the areas that it passes through. The grid would be set up as follows.

The gesture on the right would then be parsed as 74563, if you had drawn it from left to right.

There are many advantages to this method. The first is that it’s very easy to code. The design is simple and creating and defining gestures is very easy. It would also be pretty accurate when actually interpreting the gestures. It would deal with erratic mouse behavior well because human errors are of no consequence as long as it stays in the area. For this same reason, it handles curved gestures as well as straight ones.

Some downsides to this method are the lack of diagonals. This is because it would be very hard to figure out which boxes it actually crosses through. For example, you couldn’t go directly from box 7 to box 5, so it would be hard to decide which ones it actually crosses through. Another disadvantage of this method is the relative few gestures that can be defined. The gestures have to be fairly simple and relatively large. Small detailed gesture movements are ignored by this method.

Direction

This method involves determining the direction that the mouse was moved between two points. The gesture is defined by the combinations of directions that the mouse is moved. This is the method I spend the most time on. The nice thing about this method is that it is
pretty easy to code. It is also easy to define gestures. Another nice thing about this method is that it allows for diagonals. This method also allows for small as well as large movements.

The biggest downside of this method is that it is not terribly accurate. Any inconsistencies are considered changes of direction. There are methods to even out the gesture, which I experimented with, but all of these result in the loss of resolution of the gesture. Some of the methods I tried were just taking a sampling of points, or averaging the points with the ones around it. The other thing about this method is that the length of each direction doesn’t matter. This means that the gesture could be out of proportion with itself.

**Corner detection**

Another method is corner detection. At first this may seem similar to the change in direction method, but is actually quite different. This involves figuring out which points are the corners, and then looking at the relationship between those corners. The advantage of this is that it would be very accurate, provided the algorithm detects the corners properly. This method also takes into account the proportions of each part of the gesture. I think the way to approach this method would be to have the gesture so that each point is evenly spaced. When java, and I believe C++ grab the mouse events, they do it at a certain time interval. This means that the faster you move the mouse, the farther the points are apart. If the points of the gesture are spaced evenly, then they can be used for corner detection. Here is an example:

In the picture on the left, the distance between A and C is significantly greater than the distance between A and C in the picture on the right. So in a series of three points, the distance between the first and last points is a direct measure of the angle. This method could be implemented for corner detection.

One major disadvantage of this method is that it can be difficult to implement. The hard part I think would be to get all the points evenly spaced. Another downside is that it doesn’t allow for curves.
Radius of Curvature

This is a method that I believe shows a lot of promise. It has to do with figuring out the radius of curvature between consecutive points. The smaller the radius, the sharper the curve. At first it may seem that there is a curve up, down, left right, and even possibly a curve in the diagonal directions. Upon further examination, this is not true, there are only three directions, curve to the left, curve to the right or straight. There is also sharpness. For these reason, it would be important to keep track of the beginning and end points so as to get the general direction of the gesture. It is also important to note that the sharpness of the curves is proportional to the size of the gesture. The biggest advantage of this method is that it is designed for curves. Providing a good algorithm I think that this can be quite accurate.

Downsides include that it is difficult to implement. Another consideration is inconsistencies on the part of the user. I would suggest to get around this that the algorithm be designed to compare consecutive radii to check for inconsistencies.

I like the possibilities for defining gestures with this one. There are a lot of different types of gestures that can be defined. This method also has really good resolution. It has the abilities to distinguish between circles and spirals. It also seems that this is a fairly natural way of gesturing because people generally don’t move the mouse in strict right angles and straight lines.