COMBINING GESTURES AND DIRECT MANIPULATION

Dean Rubine

Information Technology Center
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA
412-268-3650
Rubine@andrew.cmu.edu

INTRODUCTION

A gesture, as the term is used here, is a handmade mark used to give a command to a computer. The attributes of the gesture (its location, size, extent, orientation, and dynamic properties) can be mapped to parameters of the command. An operation, operands, and parameters can all be communicated simultaneously with a single, intuitive, easily drawn gesture. This makes gesturing an attractive interaction technique.

Typically, a gestural interaction is completed (e.g. the stylus is lifted) before the gesture is classified, its attributes computed, and the intended command performed. There is no opportunity for the interactive manipulation of parameters in the presence of application feedback that is typical of drag operations in direct manipulation interfaces. This lack of continuous feedback during the interaction makes the use of gestures awkward for tasks that require such feedback.

The video presents a two-phase interaction technique that combines gesture and direct manipulation. A two-phase interaction begins with a gesture, which is recognized during the interaction (e.g. while the stylus is still touching the writing surface). After recognition, the application is informed and the interaction continues, allowing the user to manipulate parameters interactively. The result is a powerful interaction which combines the advantages of gesturing and direct manipulation.

EXAMPLE 1: GDP

GDP is a mouse-based drawing program that utilizes the two-phase interaction to create, copy, move, rotate, scale, delete, and group lines, rectangles, ellipses, and text. For example, the create-line gesture consists of positioning the mouse, pressing the button and making the gesture (a straight segment). The user stops moving the mouse (while continuing to press the mouse button) and the gesture is recognized. A line is created, one endpoint of which is placed at the start of the gesture, the other endpoint at the current mouse location. The user drags around the latter endpoint (rubberbanding) until the mouse button is released. The other creation gestures work similarly, utilizing the starting point of the gesture as one parameter and interactive dragging to determine another.

The gestures that operate on existing objects (move, copy, rotate-scale, group, and delete) work similarly. The copy gesture begins on an object. After it is recognized (and the mouse button is still being pressed) a copy of the object appears and may be dragged. The start of the rotate-scale gesture determines the object to be rotated as well as the center of rotation; after it is recognized the user drags a point on the object interactively rotating and scaling the object. The group gesture collects together all objects encircled by the gesture into a single composite object; after recognition, additional objects may be added to the group by touching them with the mouse cursor until the mouse button is released.

All the gestures in GDP are single strokes. While this restricts the possible gestures, it has the important property that the user holds the mouse button down during the entire two-phase interaction, releasing the button only at the end. Thus the physical tension and relaxation of the interaction correlates nicely with the mental tension and relaxation involved in performing a primitive task in the application.

Though not shown in the video, it is possible to tie other gestural attributes to application parameters in GDP. For example, one version uses the length of the create-line gesture to determine the thickness of the resulting line and the orientation of the create-rectangle gesture to determine the initial orientation of the rectangle.

EXAMPLE 2: EAGER RECOGNITION

Example 1 illustrated the two-phase interaction in which the user makes the gesture, stops moving the mouse while holding the mouse button, the gesture is recognized, and direct manipulation begins. The mouse must remain still for an interval of time (one-quarter second by default) before recognition occurs. In some contexts, forcing the user to stop is awkward. For example, using a gesture to turn a
knob would involve making the turn-knob gesture (which would presumably be the beginning of a natural turn motion of the knob), stopping, and then resuming the knob turning, this time with application feedback.

It seems desirable to have the gesture recognized as soon as enough of it has been seen to do so unambiguously. This is the goal of *eager recognition*. Thus the user would begin the turn the knob, the knob-turning gesture would be recognized, and then the knob would start to turn. No stopping or other explicit indication of the end of the gesture is necessary. What begins as a gesture smoothly becomes a direct-manipulation interaction.

The second example shows GDP with eager recognition enabled. Since the create-rectangle gesture, an ‘L’, is the only expected gesture that begins with a downward stroke, it is recognized almost immediately, a rectangle created, and direct manipulation of its corner begins. The copy gesture, a ‘C’, works similarly: as soon as the curvature of the ‘C’ is apparent, the gesture is recognized, the object is copied, and the copy is dragged. The line gesture, a straight segment, is not eagerly recognized. This is because the system recognizes that the gesture in progress may also be a move gesture, an arrow drawn with a single stroke. Thus the line gesture is only recognized when the mouse motion stops, as in the non-eager GDP. The rotate-scale and delete gestures are eagerly recognized.

EXAMPLE 3: GSCORE

GSCORE is an editor for musical scores which uses gestures and the two-phase interaction. For example, after the time-signature gesture is recognized, the z and y coordinates of the mouse interactively control the numerator and denominator of the time signature. GSCORE has separate gestures for whole notes, half notes, quarter notes, eighth notes and sixteenth notes (actually, there are two gestures for each except whole notes, one for upward stems, and one for downward stems). While it would be possible to have a single note gesture and then interactively control the duration and stem direction of the note, having separate gestures appears to result in faster interactions. It is possible to edit the set of gestures and their meanings (at runtime) to try out various interfaces, so the two approaches may be compared.

The video shows how a new gesture is added to GSCORE. Gestures are associated with classes of views on the screen (e.g. a note responds to a different set of gestures than a staff), so the user first clicks to examine the set of gestures associated with a particular view. Adding a new gesture involves pressing the “new class” button and then entering fifteen examples of the new gesture. There is a click-and-drag interface to an Objective-C interpreter through which the semantics of the gesture are specified. The new gesture may be tried immediately. Although not shown, it is possible to evaluate the new classifier by testing all the training examples. This evaluation indicates when a new gesture is mistaken for an existing gesture (or vice versa), indicating that the new gesture needs to be redesigned to look different.

EXAMPLE 4: MULTIPLE FINGER GESTURES

The two-phase interaction may be used with multi-finger input. The video shows MDP, a version of the drawing program that uses a Sensor Frame as a multi-finger input device. The mouse gestures of GDP are mapped to single-finger gestures in MDP. After the gesture is recognized, additional fingers may be brought into the sensing plane of the Sensor Frame to control additional parameters. For example, after recognizing create-line, the first finger rubberbands one endpoint of the new line (as in GDP), and additional fingers control the line’s color and thickness.

Multiple finger gestures are also recognized. The training of the undo gesture is shown, and later the use of undo is demonstrated. The two-phase interaction allows the amount of “undoing” to be determined interactively after the gesture is recognized. Also interesting is the two finger rotate-scale-translate gesture. After recognition, each of the two fingers attaches to a point on the object. By spreading the fingers apart or moving the fingers closer together the object can be scaled, by changing the orientation of the fingers the object can be rotated, and by moving the two fingers in parallel the object can be translated. It is seen that multiple finger interaction allows for the intuitive manipulation of many parameters simultaneously.

CONCLUSION

The purpose of the video is to demonstrate the two-phase interaction technique, in which gesture and direct-manipulation are combined. While previously described in print [1, 2, 3], the video shows the dynamic aspects of the interaction in ways that print cannot. In particular, it shows how the two-phase interaction is applicable in two applications, a drawing program and a score editor, and for both single-pointer and multiple-pointer input devices. Eager recognition, a technique for smoothing the transition between the gesture and the direct manipulation phases of the interaction, is also shown. While the video demonstrates the potential of combining gesture and direct manipulation, user testing is needed to determine if this potential can be realized.

REFERENCES

