Used together, these features make a buffer's mode closely analogous to an object's class: the mode determines which commands are available and provides the variables upon which those commands' Lisp implementations depend.

For each buffer, the text-editing primitives maintain an "undo" log, which holds enough information to revert their effects. The undo log remembers the boundaries between user commands, so a user command that carries out many primitive operations can be undone as a unit.

When Emacs Lisp code operates on a buffer, it can use integers to specify character positions within a buffer's text. This is a simple approach, but insertions and deletions before a given piece of text cause its numeric position to change from one moment to the next. To track positions within a buffer as its contents are modified, Lisp code can create *marker* objects, which float with the text they're next to. Any primitive operation that accepts integers as buffer positions also accepts markers.

Lisp code can attach *text properties* to the characters in a buffer. A text property has a name and a value, and both can be arbitrary Lisp objects. Logically, each character's properties are independent, but Emacs's representation of text properties stores a run of consecutive characters with identical text properties efficiently, and Emacs Lisp includes primitives to quickly find positions where text properties change, so in effect, text properties mark ranges of text. Text properties can specify how Emacs should display the text, and also how Emacs should respond to mouse gestures applied to it. Text properties can even specify special keyboard commands available to the user only when the cursor is on that text. A buffer's undo log records changes to its text properties, in addition to the changes to the text itself. Emacs Lisp strings can have text properties, too: extracting text from a buffer as a string and inserting that string into a buffer elsewhere carries the text properties along.

An *overlay* represents a contiguous stretch of text in a buffer. The start and end of an overlay float with the text around them, as markers would. Like text properties, overlays can affect the display and mouse sensitivity of the text they cover, but unlike text properties, overlays are not considered part of the text: strings cannot have overlays, and the undo log doesn't record changes to overlay endpoints.

The View: Emacs's Redisplay Engine

As the user edits text and manipulates windows, Emacs's redisplay engine takes care of keeping the display up to date. Emacs redisplay has two important characteristics:

Emacs updates the display automatically
: Lisp code need not specify how to update the display. Rather, it is free to manipulate buffer text, properties, overlays, and window configurations as needed, without regard for which portions of the display will become out of date. When the time comes, Emacs's redisplay code looks over the accumulated changes and finds an efficient set of drawing operations that will bring the display into line with the new state of the Model. By relieving Lisp code