

Make those text boxes the right size!

A surprisingly common cause of user irritation is trivial to avoid.

Suppose we're designing a GUI form that needs a user-entered text field with a maximum length. Consider a **city name**, defined in our database as a 15-position text item. Suppose a user wants to enter "Arlington Heights", which contains 17 characters.

We obviously have a mismatch between what our program will accept and what the user wants to enter. Let's look at some unacceptable ways that many programs handle that situation.

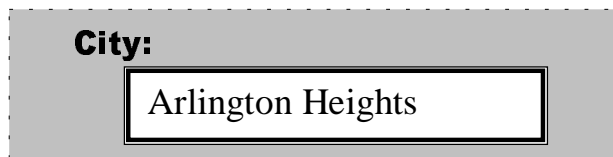
Bad idea #1: a box too big

Suppose we define a text box like this:

A screenshot of a GUI form. It has a label "City:" in bold. Below the label is a text box with a double border. The text box contains the text "Arlington Heigh". The text box is significantly larger than the text it contains, leaving a large amount of empty space.

The program simply stops accepting characters after the limit is reached. It may or may not *beep* to alert the user to the field overflow.

A worse strategy is to let the user type the whole name:

A screenshot of a GUI form. It has a label "City:" in bold. Below the label is a text box with a double border. The text box contains the text "Arlington Heights". The text box is larger than the text it contains, but the text is cut off at the end.

and then to chop off the excess characters when we store the data.

In both cases the box *looks* big enough to the user, even though the program can't handle the input.

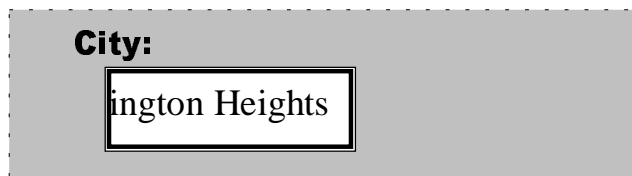
Bad idea #2: a box too small

Realizing that we've been too stingy with our maximum length, suppose we redefine the database field to accept up to 18 characters.

But now another common mistake is to make the box *smaller* than the longest possible name, and then either let the user keep typing with no feedback:

A screenshot of a GUI form. It has a label "City:" in bold. Below the label is a text box with a double border. The text box contains the text "Arlington Heig". The text box is smaller than the text it contains, so the text is cut off.

or scroll the field, so the user can't see it all at once:

A screenshot of a GUI form. It has a label "City:" in bold. Below the label is a text box with a double border. The text box contains the text "ington Heights". The text box is smaller than the text it contains, so the beginning of the text is cut off.

In both of those cases the data probably got entered and stored correctly, but the user who doesn't get to see the whole field is left to wonder.

Every one of the four above techniques is inexcusably user-unfriendly. Yet we keep seeing them in mainstream software products from major software vendors, including vendors most known for pioneering and promoting graphical interfaces.

An obvious solution: a box just right

The user-friendly strategy, of course, is to size the box so that the longest acceptable value fits exactly. To make that work we have to do one more thing:

Use **Courier** or a similar monospace font inside the text box.

That should be a standard rule for almost *all* text boxes on almost all forms. It's surprising (and irritating) that it isn't the *default* choice in so-called "visual" programming tools.

Now the user knows exactly what's permitted and can see the entered data item on the form:

A screenshot of a form field. The label "City:" is in a bold, sans-serif font. Below it is a text box with a double-line border. Inside the text box, the text "Arlington Heights" is displayed in a monospace font. The entire form field is set against a light gray background with a dashed border.

If you like prettier fonts, use them in your captions, menus, buttons, etc., but never in a text box or other field to be entered by the user.