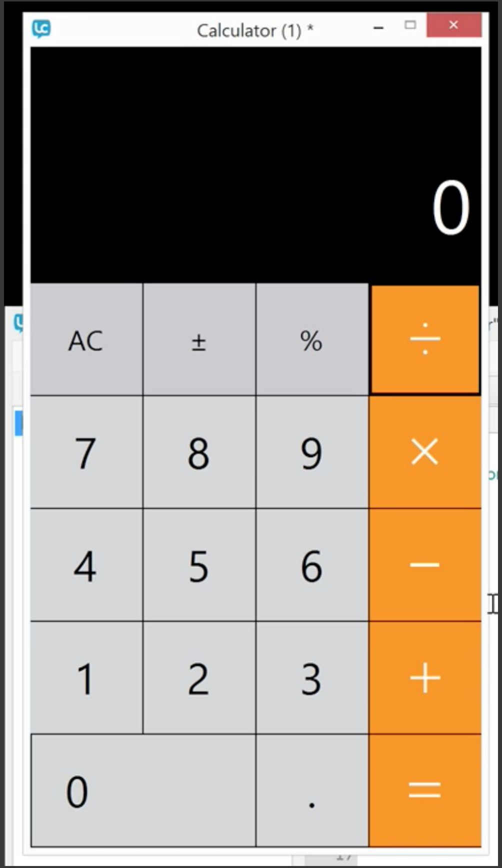
## User Interaction

In the original app a black border is shown when an operator is clicked, this remains as an indicator to show the user which operator is selected - giving a better user experience.

**Q**: How do we do that?
**A**: Code

We have already written some code in a **mouseUp** handler in the Simple Messages App. We are going to do the same here - when the user clicks on any one of the **divide**, **multiply**, **plus** or **minus** buttons, we want to add a black border to the button.



## Coding the Buttons

For each of the buttons **divide**, **multiply**, **plus** and **minus** do the following steps:

1. Open the Property Inspector for the button, in the Icons & Border pane set the **border width** property to 2

2. Open the Script Editor for the button and add the code on the right

A quick reminder on how to open the Script Editor for an object:

- In edit mode, select the object on the stack
- Open the Object menu
- Select “Object Script”

```
on mouseUp
  set the showBorder of me to true
end mouseUp
```

LIVECODE Calculator

## Coding the Buttons

So what does the code that we have added mean?

1. This line starts the **mouseUp** handler - when the **mouseUp** message is sent to the button, LiveCode looks for this line to see what code to execute

2. This line sets the property **showBorder** of the button to true - in other words it gives the button a border

3. This line ends the **mouseUp** handler - there is no more code for LiveCode to execute

```
1  on mouseUp
2     set the showBorder of me to true
3  end mouseUp
```

## LiveCode Key Concept

**me -** This keyword is used to reference the object that contains the currently running handler.

Use the **me** keyword within a handler to determine which object's script is executing.

For example, if the user clicks on the **divide** button the **mouseUp** message is sent to that button. There is a **mouseUp** handler in the script of button **divide**, so LiveCode executes the script of the button **divide**. The **me** keyword in this example refers to the button **divide.**

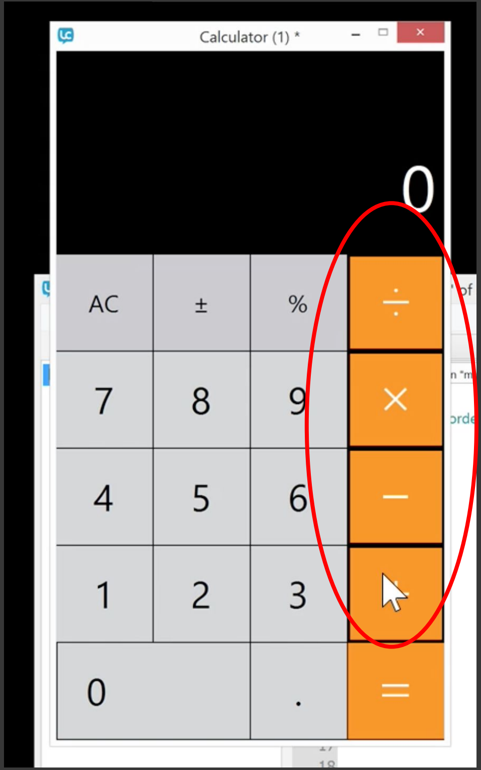
# Testing Your Code

Now switch to **Run** mode in the tools palette and test your code.

Press each button in turn and see what happens!

Do you notice a problem?

(Hint: look at the image on the right)



## Resetting the Border Color

Only the most recently selected operator should have a border, but the border never goes back to off when you click on another operator...

We need to use code to set the other buttons to no border. What we might do is open the Script Editor for each button and add a further 4 lines of code at the beginning of the **mouseUp** handler for each button:

```
on mouseUp
   set the showBorder of button "divide" to false
   set the showBorder of button "multiply" to false
   set the showBorder of button "minus" to false
   set the showBorder of button "plus" to false
   set the showBorder of me to true
end mouseUp
```

So that is an extra 16 lines of code, that all do the same thing… But why would you copy so much code to every button? Well, **you shouldn’t!**