Scripting Tutorial - Lesson 15: (Advanced) Using Mouse Controls with Multiple Classes

Download supporting files for this tutorial

Texas Instruments TI-Nspire Scripting Support Page

In lesson 14 we saw how to use Keyboard controls to select and work with multiple class objects. In this final lesson, we extend our mouse controls in the same way. Most of what we need has already been put in place using the **TrackedObject** variable. All that is really needed is to generalize the **mouseDown** function to work with out **Objects** table rather than the specific case of the Square previously defined.

We need to study the various parts of the **mouseDown** function closely in order to understand what is happening here.

The function begins just as **tabKey** did, with a loop through the objects in the **Objects** table, allocating a local variable **obj** to each in turn. As we might expect from previous mouse work, if the mouse down occurs within a particular object (using the **contains** function) then that becomes our selected object to be Tracked.

As before, first release it if the object was

```
function on.mouseDown(x,y)
    for i = #Objects, 1, -1 do
        local obj = Objects[i]
```

already selected so we know what state we are in. Then we define TrackedObject as obj, and indicate that it is **selected**.

The next two lines define two new global variables called **TrackOffsetx** and **TrackOffsety**. You may have noticed that if we click on an object somewhere other than the center, the object is inclined to jump to place the center under the mouse. These TrackedOffset values serve to avoid this little jump. They should initially be defined as zero in our **on.resize** function so that they exist when the page is created and changed.

Back to tracking multiple objects. The table.remove and table.insert commands serve to release any previous selected object and replace with the current one – essentially moving it to prime position in the table. This has the effect desired – the object being clicked on becomes the selected (Tracked)

```
            if obj:contains(x, y) then
                if TrackedObject ~= nil then
                    TrackedObject.selected
                    = false
                    end
                TrackedObject = obj
                obj.selected = true
                TrackOffsetx = TrackedObject.x
                - x
                TrackOffsety = TrackedObject.y -
                y
                table.remove(Objects, i)
                table.insert(Objects, obj)
                platform.window:invalidate()
                break
            end
        end
    end
    function on.mouseUp(x,y)
        if TrackedObject ~= nil then
            TrackedObject.selected = false
            end
        TrackedObject = nil
    end
    function on.mouseMove(x,y)
        if TrackedObject ~= nil then
            TrackedObject.x = x + TrackOffsetx
            TrackedObject.y = y + TrackOffsety
            platform.window:invalidate()
        end
    end
```

object and all is well.

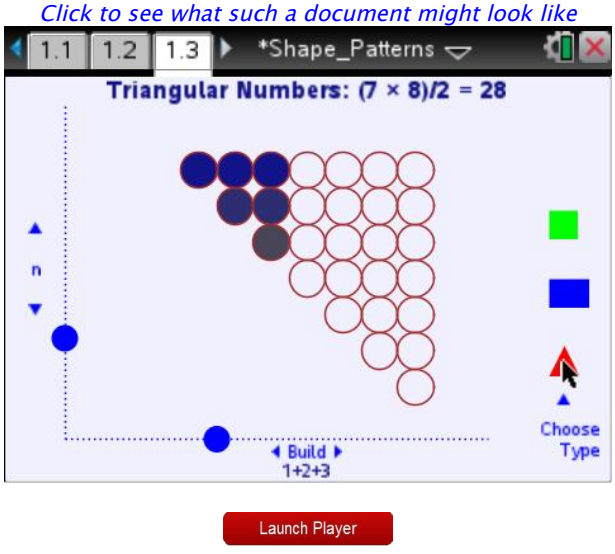
Only one other change is made to **mouseMove** and that is simply to include the TrackOffset values referred to previously.

Copy this code into script_tut14.lua and you now have completed your mission. Two objects may now be selected and controlled by either mouse or keyboard!

I would suggest that you now extend this script to include other objects. Simply define these and then insert their definition into the Objects table and you will be able to control them just as easily as you control the square and the circle. Multiple images will work just as well.

Time then, perhaps, to go back to the **Shape Patterns** document featured in lesson 11 and think about how you may now create your own rich and powerful documents which work equally well with both handheld and computer.

Click to see what such a document might look like



Congratulations on completing this series of lessons!

Back to Top
