TI-Nspire Script Samples

Contents

1 Copyright

2 TI-Nspire Scripting

Lua (pronounced LOO-ah) is widely used in all areas that can benefit from a simple, extensible, portable, and efficient scripting language, such as embedded systems, mobile devices, web servers, and games. It was designed from the outset to be integrated with software written in C and other conventional languages. Lua enjoys support today at its official web site, http://www.lua.org, and community web sites http://lua-users.org and http://luaforge.net.

Lua is free open-source software, distributed under a very liberal license (the well-known MIT license). It may be used for any purpose, including commercial purposes, at absolutely no cost, and under no obligation to give our software back to the open-source community.

Lua offers script developers a safe environment, automatic memory management, and good facilities for handling strings and other kinds of data with dynamic size.

TI-Nspire scripts run in a sandbox.

- They cannot directly communicate with other scripts or with other anchor applications (Graphs & Geometry, Lists and Spreadsheets, etc). The symbol table and clipboard are the only avenues of communication between scripts.
- Script documents cannot read or write to the file system, thereby preventing accidental or intentional wear on the flash storage of the TI-Nspire hand-held.
- Scripts are delivered in TI-Nspire TNS documents. They cannot interfere with Press-to-Test. Teachers can send a document containing a script as needed to the hand-held unit while in PTT mode.

The TI-Nspire implementation of Lua runs on all our platforms: Windows, Mac OS X, CX hand-held, and TI-Nspire with Touchpad.

3 Script Features

Most Nspire components are accessible to scripts though Lua extensions. This permits content developers to create scripts that can interact with the user through a compelling graphical interface, perform advanced mathematical calculations with the math engine, respond in real time to Vernier data collection probes, and share data with other script documents and the anchor applications (Graphs & Geometry, Lists and Spreadsheets, etc.) through the symbol table.

Each script resides in a page or in one pane of a page of a TNS document. Scripts may be mixed with anchor applications and other scripts in multiple pages to create lesson material with instructions, presentation, simulations, interactive exploration, and educational games.

4 Script Development Resources

Resources are presently available to aid developers in the creation of scripts. Documentation for the Lua language is available on the Internet and in books available from Amazon.com. Documentation for the Nspire component extensions to Lua is maintained in *Nspire Scripting Programming Interface.doc*.

Several script documents have been developed to serve as example code and inspiration. These highlight such concepts as reading and writing to the symbol table, scaling graphics to desktop or hand-held display, dealing with images, dragging with the mouse, animation, magnetic attractor transitions, saving and restoring state, displaying Unicode characters, UI controls, and reading sensor probes.

TI-Nspire Scripting Tools.exe is a utility that performs a couple of essential functions.

Primarily it packages scripts to send to the Nspire desktop software. The input to this function is a text file containing a Lua script. The script is sent to the Windows clipboard in a form acceptable to paste into a page of an Nspire document.

The second feature converts an image file into a Lua string. The output string is a binary pixel image suitable for display in the Nspire framework. To use this feature, select the menu command Tools > Image serialization to clipboard. From the File Open dialog box, open an image file. The image data is converted to a Lua string and placed on the Windows clipboard. The string can be pasted into a Lua script with your favorite text editor.

Several image file formats are supported: jpg, png, tiff, bmp, gif, pbm, pgm, and ppm.

5 Example Scripts

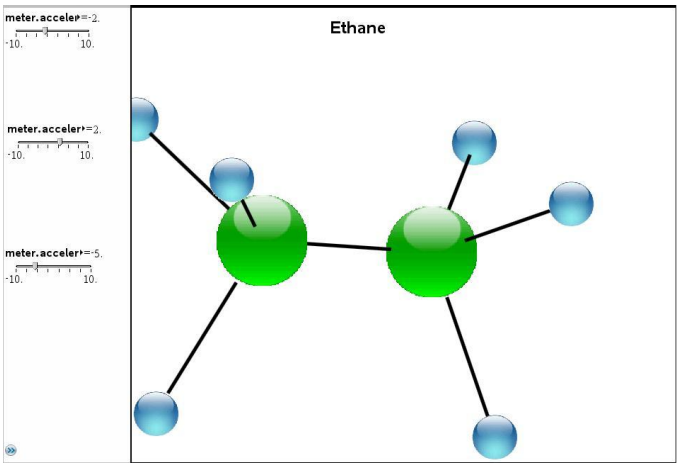
5.1 Ethane2

Description:

This script displays the ethane molecule in three dimensions. A student with access to the Vernier 3-axis accelerometer can rotate the molecule to view it from different perspectives. Students without access to the accelerometer can use sliders to tilt the model to different angles.

It is table-driven so other molecules are easily modeled.

It works equally well on desktop as well as hand-held Nspire.



Programming Highlights:

Concepts: reading the symbol table; periodic repaint; scaling graphics; images

The Nspire symbol table associates variable names with values. Values in the symbol table may be numbers, strings, mathematical expressions, function definitions, or lists or matrices of numbers, strings, and mathematical expressions. Each *problem* in an Nspire document has a separate symbol table.

Nspire scriptscan access the contents of variables in the symbol table. Script Ethane2.lua demonstrates reading the values from the symbol table to control the angle of view of an ethane molecule. When a Vernier 3-axes accelerometer is connected via data collection cradle to the Nspire software or hand-held device, variables meter.acceleration, meter.acceleration2, and meter.acceleration3 are updated to indicate the orientation of the accelerometer with respect to earth's gravitational field.

The Ethane2 script app sets the timer to tick every ¼ seconds in its on.create() handler. Each tick of the timer calls the script's on.timer() handler. The timer handler invalidates the graphical contents of the output window, which in turn invokes the script's on.paint() handler. It is in the on.paint() that the real action takes place.

The on.paint() handler reads the amount of acceleration along the x-, y-, and z-axis with calls to var.recall().

```
local x = var.recall(AccelX) or 0
local y = var.recall(AccelY) or 0
local z = var.recall(AccelZ) or -9.8
```

AccelX was conveniently set to “meter.acceleration” earlier during the initial execution of the script. Note the use of boolean evaluation in the expressions above. If var.recall() cannot recall a variable from the symbol table, it returns nil. The value of the Lua “or” operator is the first expression that evaluates to a true or non-nil value. The above assignment sets x to a default value of 0 if symbol table variable meter.acceleration is undefined. This is often the case before a Vernier sensor has been plugged in.

In order for the ethane molecule to remain visible in the Nspire window whether displayed in the desktop software or the hand-held unit, scaling parameters are recalculated each time the script’s on.resize() handler is called. The on.resize() handler is called before the first paint event occurs and each time the user resizes the window.

This script maintains a 10 by 10 square coordinate system. In response to a resize event, routine Layout.resize() subdivides the shortest side by 10 to set the basic scaling factor of all further graphical scaling calculations. This assures that the ethane molecule will be visible in both the x and y dimensions.

Images of blue and green spheres are used to represent the hydrogen and carbon atoms of the ethane molecule. At the very bottom of the Ethane2.lua script are calls to image.new() to create these images from the binary string representation of the pixel data. These images must be resized whenever the coordinate system changes. This scaling is accomplished by calling the image:copy() method of the original image. See routine Sphere:resize() for an example of calling the copy() method to resize the sphere images.

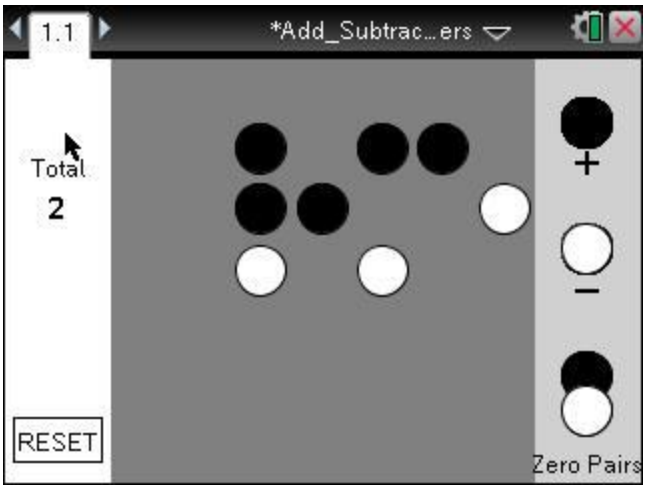
Note that the TI-Nspire Scripting Tools.exe has an option to create the binary string representation of images (bmp, gif, jpg, png, pbm, pgm, ppm, tiff) suitable for input to image.new().

5.2 Add_Subtract_Integers

Description:

Script Add_Subtract_Integers allows the student to manipulate positive and negative counters. Counters are represented by black (positive) and white (negative) circles. The student can click on a counter and drag it around. Counters dropped in the large central gray area are counted, i.e. they affect the total in the left panel of the display.

This is an early study in reproducing the action-consequence document Add_Subtract_Integers_ACnew as a script app. The user interface is poor and does not use color. It needs work to be presentable to users. This app works on the hand-held, but because it is heavily mouse-oriented, it works best on the desktop software.



Programming Highlights:

Concepts: dragging with the mouse; animation to a magnetic attractor

Counter objects have a contains() method which returns true if the mouse pointer is within the bounds of the circle representing the counter.

Counter method pickup() is called when the user clicks on a counter. This routine allows the counter object to extricate itself of its current location and prepare to be dragged around.

Counter method setxy() is called by the dragging routine to update the counter's current location as it is being dragged around.

Counter method drop() is called when the user releases the mouse button. The code for this method decides where the counter was dropped (inside or outside the gray counting area?) and then either nudges itself over to a magnetic grid point or zips back to the counter well.

Event handler on.mouseDown() determines which counter it clicked down on and then sets up the on.mouseMove() handler to track the counter object.

Each time the mouse pointer moves, the on.mouseMove() event handler is called which in turn calls the counter's setxy() method to set its location to match the mouse pointer's movements.

When the user releases the mouse button, the on.mouseUp() event handler calls the counter's drop() method to set the counter free.

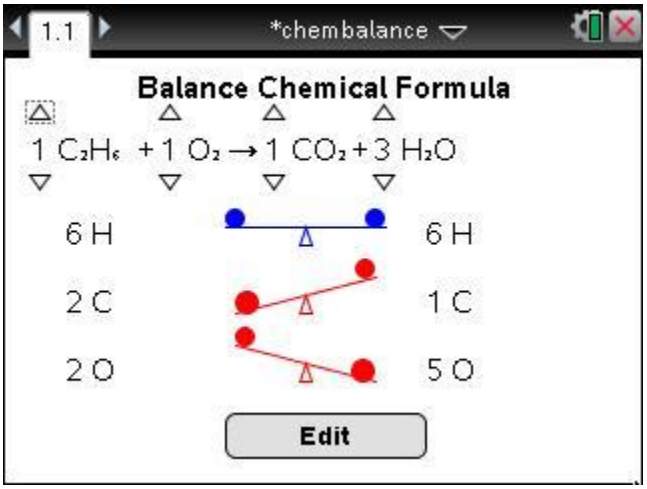
Animation is used in this script to ease counters to magnetic grid points or to zoom back to the counter well. When a counter is dropped, it calls animation.zipto() to add itself to the list of animated objects. Each tick of the timer calls animation.step() to update the location of all the objects in the animation list. As an object nears its final resting place, it takes smaller steps to signal to the user that the object is approaching its destination. When an object arrives at its destination, it is removed from the animation list.

5.3 Chembalance

Description:

This allows the student to enter a chemical reaction then balance the reactants and resultants with a colorful user interface.

This script works well on hand-held and desktop Nspire. Keypad navigation (tab, backtab, enter) caters to hand-held users. Additionally all buttons and text fields respond to mouse clicks for ease of use in desktop Nspire.



Programming Highlights:

Concepts: focus management; saving & restoring state; Unicode characters

The mouse pointer is difficult to use on the hand-held device, so keypad navigation is all the more important for hand-held ease of use. The chembalance.lua script maintains a list of objects (buttons and text fields) that can receive input from the keyboard—the focus list. The student can press tab or shift + tab to navigate forward and backward through the focus list. The user can press enter or space to click a button.

The mouse pointer is much easier to use on the desktop version of Nspire. All buttons and text fields respond to mouse clicks in addition to keyboard navigation aids.

This script implements the on.save() and on.restore() event handlers. The implementation of on.save() saves the chemical production entered by the user. The implementation of on.restore() restores the chemical formula to be displayed and explored by the user.

Standard chemical notation employs subscripts to denote the number of elements in a molecule. The Unicode character numbers of the subscript digits are encoded by calls to string.uchar(…). For example the Unicode encoding for subscript “0” is string.uchar(0x2080). The output of this function is a Lua string containing the UTF-8 encoding of Unicode character U+2080. See routine MoleculeText:format() in the

chembalance.lua script for an example of how digits are converted to subscripts. It uses table Subscript to look up substitutions for digit-to-subscript conversions.

5.4 Countdowntimer

Description:

This is a simple script that allows the user to specify a number of minutes and seconds to count down. The interface shows a digital clock that decrements until it gets to 0:00.

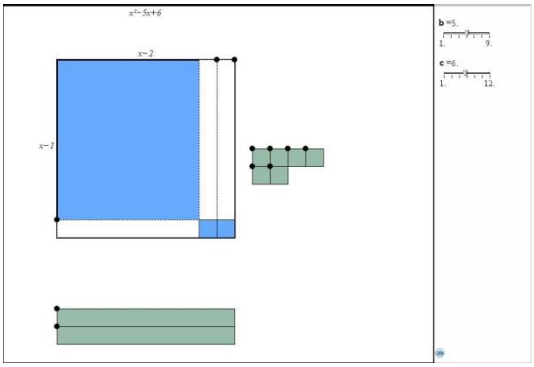
It works well on desktop and hand-held software.

5.5 Factoring-trinomials

Description:

This is another study of rendering an action-consequence document as a script. It is modeled after Factoring_Trinomials_Part_2_Overlapping_Areas.

This scriptworks on the hand-held but works best on desktop Nspire. it is not easy to discover how to use it.



Programming Highlights:

Concepts: reading the symbol table; animation

This script uses animation to highlight the relationship between the coefficients of a trinomial expression and the geometric representation of the factors of a trinomial. As the user changes the coefficients of the trinomial expression, rectangles and squares fly in or out of the display to illustrate the effect of each change. The symbol table is used to communicate the values of the coefficients as set by the user with sliders.

5.6 Hotmolecules

Description:

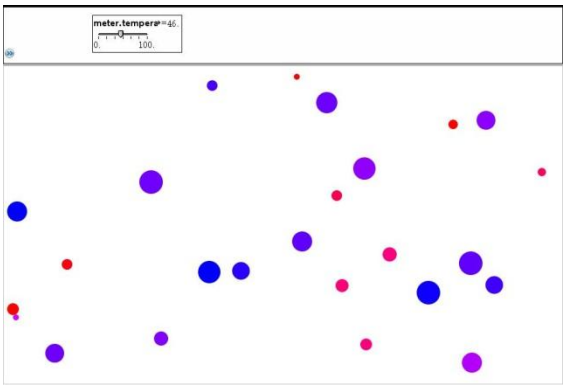
This script bounces many circles against each other and the walls of the display area. The speed and color of the circles are influenced by the temperature reading of an attached Vernier temperature probe.

Programming Highlights:

Concepts: animating many objects; reading sensor probes

The temperature probe communicates the current temperature value through symbol table variable meter.temperature. The script reads the value of meter.temperature to compute the energy that flows into the circles.

This app works equally well on desktop and hand-held Nspire.

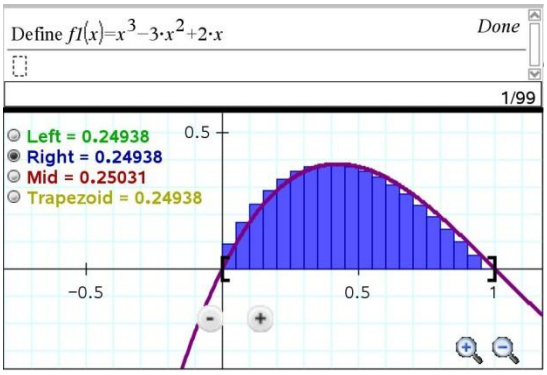


5.7 Riemann

Description:

This script is a demonstration of Riemann integration—a method of integrating the area under the curve of a function by summing the area of successively thinner rectangles.

It works on the hand-held, but because of the dependence on a mouse interface it works best on desktop Nspire.



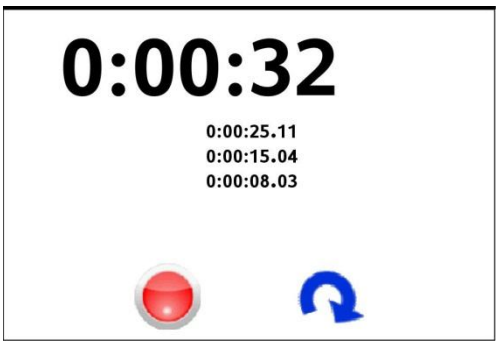
5.8 Stopwatch

Description:

This scripts provides a basic stopwatch with lap counter. The time of each lap is exported to a list in the symbol table which can be used by Data & Statistics or Lists & Spreadsheets for further analysis.

It works well on hand-held and desktop Nspire.

, the script should be updated to allow the user to calibrate the stopwatch against a known good time base.



5.9 Unitconversion

Description:

This provides easy access to many unit conversions, even on non-CAS software.

It works well on hand-held and desktop Nspire.

Unit Conversion	
172 cm_ =	
1.72e+015 fermi	0.342003 rd
1.72e+010 Å	0.00855007 furl
1.72e+006 µ	0.00172 km
67716.5 mil	0.00106876 mi
1720 mm	0.000928726 Nmi
172 cm	1.14975e−011 au
67.7165 in	1.81808e−016 ltyr
5.64304 ft	5.57414e−017 parsec
1.88101 yd	
1.72 m	
0.940507 fath	