

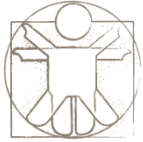
Sketchlet Tutorial

Scripting

sketchlet.sf.net

Željko Obrenović

obren.info/



Scripts

- In Sketchlet scripting languages can be used to quickly outline the behavior of sketches
- Scripts are proven, highly productive and simple to learn and use end-user development paradigm
- Define more complex interaction scenarios, without requiring intensive programming



Scripts

- We currently support several higher-level scripting languages including Javascript, Python, BeanShell, and Groovy (experimental support for Ruby, TCL, Sleep, Haskell, and Prolog)



Sketchlet Extends Scripting Languages

- Sketchlet Scripting Extensions
 - Working with Variables
 - Getting User Input
 - Working with Region and Sketch Properties
 - Pause and Wait
 - Graphics



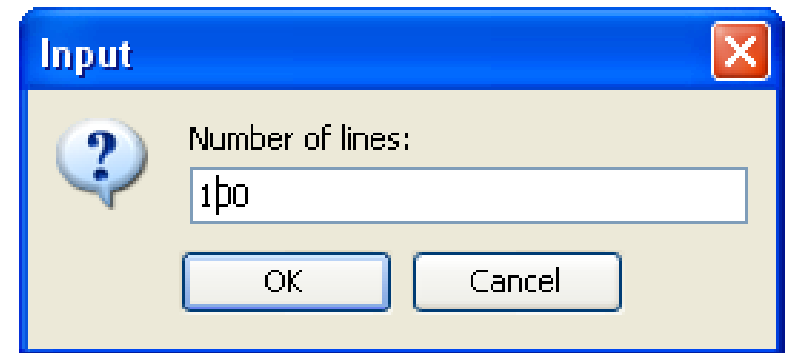
Extensions – Working with Variables

- **sketchlet.update**(String variable, String value)
- **sketchlet.update**(String variable, int value)
- **sketchlet.update**(String variable, double value)
- String **sketchlet.get**(String variable)
- String **sketchlet.getString**(String variable)
- int **sketchlet.getInteger**(String variable)
- double **sketchlet.getDouble**(String variable)
- int **sketchlet.getCount**(String variable)
- int **sketchlet.getTimestamp**(String variable)



Extensions – Showing Message and Getting User Input

- `sketchlet.showMessage(String message)`
- `String sketchlet.ask(String question)`
- `String sketchlet.askString(String question)`
- `int sketchlet.askInteger(String question)`
- `double sketchlet.askDouble(String question)`





Extensions – Working with Region Properties

- **setRegionProperty**(int number, String property, String value)
- **setRegionProperty**(int number, String property, int value)
- **setRegionProperty**(int number, String property, double value)
- String **getRegionProperty**(int number, String property)
- String **getRegionPropertyAsString**(int number, String property)
- int **getRegionPropertyAsInteger**(int number, String property)
- double **getRegionPropertyAsDouble**(int number, String property)



Position	
position x	horizontal position (left, 0 to 1000)
position y	vertical position (top, 0 to 1000)
relative x	relative horizontal position (0.0 to 1.0)
relative y	vertical position (0.0 to 1.0)
trajectory position	0.0 to 1.0
Size	
width	region width
height	region height
Orientation	
rotation	angle
Transparency	
transparency	0.0 to 1.0
Visible area	
visible area x	
visible area y	
visible area width	
visible area height	
Motion	
speed	pixels per second
direction	angle
Pen	
pen thickness	0, 1, 2...

Advanced / Coordinates	
x1	
y1	
x2	
y2	
Advanced / Sheer	
shear x	0.0 to 1.0
shear y	0.0 to 1.0
Advanced / 3D	
horizontal 3d rotation	0 to 360
vertical 3d rotation	0 to 360
Advanced / Perspective	
perspective x1	0 to 1, x top left corner
perspective y1	0 to 1, y top left corner
perspective x2	0 to 1, x top right corner
perspective y2	0 to 1, y top right corner
perspective x3	0 to 1, x bottom right corner
perspective y3	0 to 1, y bottom right corner
perspective x4	0 to 1, x bottom left corner
perspective y4	0 to 1, y bottom left corner
automatic perspective	left, right, top, bottom, parallel
perspective depth	relative perceptible depth 0.0 to 1.0



Extensions – Working with Sketch Properties

- **setSketchProperty**(String property, String value)
- **setSketchProperty**(String property, int value)
- **setSketchProperty**(String property, double value)
- String **getSketchProperty**(String property)
- String **getSketchPropertyAsString**(String property)
- int **getSketchPropertyAsInteger**(String property)
- double **getSketchPropertyAsDouble**(String property)



Settings panel with tabs: **Set Properties** | Animate Properties | Map to Numeric Variables

Left sidebar icons: - -

Property	Value	Description
Color		
background color		red, blue, green, gray, yellow...
transparency		0.0 .. 1.0
Zoom		
		1.0 means 100%
zoom		1.0 means 100%
zoom center x		default is 0
zoom center y		default is 0
Offset		
background offset x		
background offset y		
regions offset x		
regions offset y		
Perspective		
perspective type		1 point or two point
perspective y		horizon
perspective x1		point 1 on the horizon
perspective x2		point 2 on the horizon

Right sidebar: [Explore](#) [Clear](#)



Extensions – Pause and Wait

- **sketchlet.pause**(double seconds)
- **sketchlet.waitForUpdate**(String variable)
- **sketchlet.waitUntil**(String expression)



Extensions – Graphics

- **graphics.clearCanvas()**
- **graphics.repaint()**
- **graphics.setColor(int r, int g, int b)**
- **graphics.setColor(int r, int g, int b, int transparency)**
- **graphics.setTransparency(float transparency)**
- **graphics.setLineWidth(double width)**
- **graphics.setFont(String name, String style, int size)**
- **graphics.translate(int x, int y)**
- **graphics.rotate(double angle, int x, int y)**
- **graphics.scale(double x, double y)**
- **graphics.shear(double x, double y)};**



Extensions – Graphics

- **sketchlet.drawText**(String text, int x, int y)
- **sketchlet.drawLine**(int x1, int y1, int x2, int y2)
- **sketchlet.drawRect**(int x, int y, int w, int h)
- **sketchlet.drawEllipse**(int x, int y, int w, int h)
- **sketchlet.drawCircle**(int center_x, int center_y, int r)
- **sketchlet.fillRect**(int x, int y, int w, int h)
- **sketchlet.fillEllipse**(int x, int y, int w, int h)
- **sketchlet.fillCircle**(int center_x, int center_y, int r)
- **sketchlet.drawImage**(String strPathOrURL, int x, int y)
- **sketchlet.drawImage**(String strPathOrURL, int x, int y, int w, int h)
- **sketchlet.getTextWidth**(String text)
- **sketchlet.getTextHeight**(String text)



Variable Declarations Inside Scripts

- When a script is called, Sketchlet variables will be redeclared within the script
 - Variables may be renamed to satisfy naming convention of scripting languages

Sketchlet Variable Name	Declaration in scripts
position x	position_x
motion-intensity	motion_intensity
a	a

- Read-only, use *sketchlet.update* to change the value of a Sketchlet variable



Script Editor

The screenshot displays the Sketch 4 application window. The main canvas shows a series of horizontal lines that become increasingly dense towards the right side. The interface includes a menu bar (Sketchlet, Edit, Variable, I/O Service, Script, External Tools, Settings, View, Blog), a toolbar with various icons, and a sidebar on the left with drawing tools. The bottom section contains a script editor with two tabs: 'clear.js' and 'script.js'. The 'script.js' tab is active, showing the following code:

```
1 amico.clearCanvas();
2 n = amico.askInteger("Number of lines:");
3 for (i = 0; i < n; i++) {
4   amico.drawLine(i * 10, 100, i * 20, 200);
5 }
6
```

To the right of the script editor are buttons for 'Run', 'Stop', 'Declarations', 'Extensions', and 'Reload'. On the far right, there is a 'Variables' panel with a table of system variables:

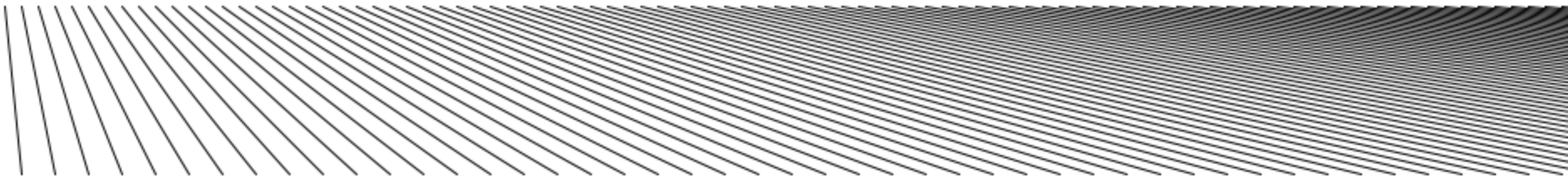
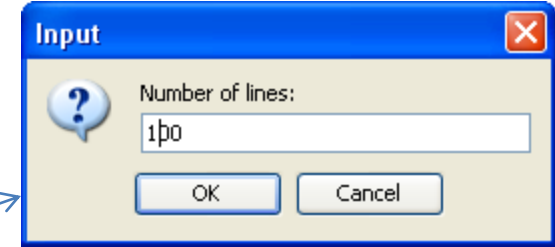
Variable Name	Value	Descr
time_hour	10	
time_minute	15	
time_second	34	
mobile-image-path	C:\DOCUME~1\ZO...	
mobile-image-base64		
sms-send-number	number	
sms-send-message	message	
sms-received-from	number	
sms-received-message	message	
mobile-text-alert	message	
mobile-vibration	1000	
mobile-screen-width	240	
mobile-screen-height	235	
mobile-key-pressed	?	
mobile-key-code	50	

At the bottom left, the text 'Sketching mode' is visible.



Example

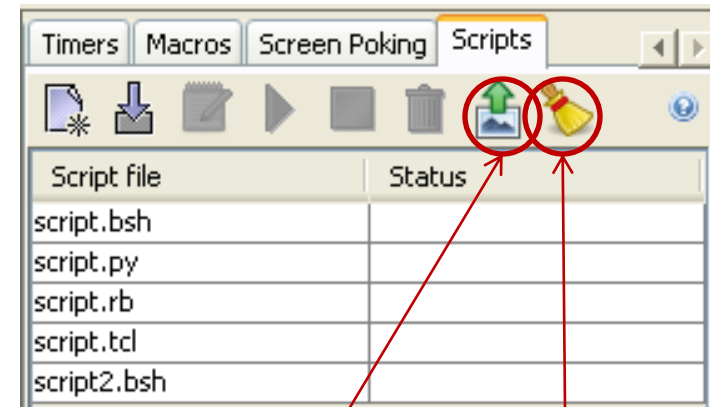
```
graphics.clearCanvas();  
n = sketchlet.askInteger("Number of lines:");  
for (i = 0; i < n; i++) {  
    graphics.drawLine(i * 10, 100, i * 20, 200);  
}
```





Merging Image Generated by Scripts and Background Sketch Image

- Scripts draw in a separate layer on top of the sketch
- The image from this layer can be merged with the background sketch image (i.e. it becomes a part of that image)



Merge the image generated by scripts with the background sketch image

Clear the image generated by scripts



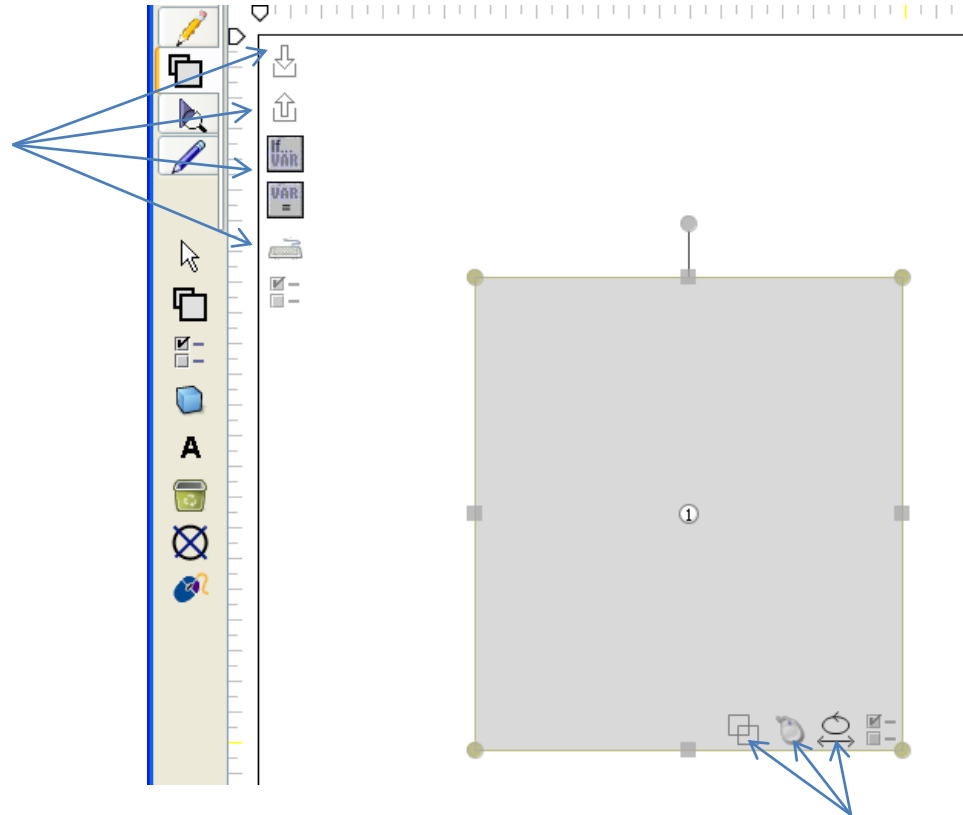
Calling Scripts

- Scripts can be called from several places
 - On active region mouse events
 - On sketch events (entry or exit)
 - On variable updates ("On Variable Update" actions)
 - On keyboard events
 - From other macros, as one of the commands
- Drag-and-Drop on any sketch or region event
- Directly specify in settings

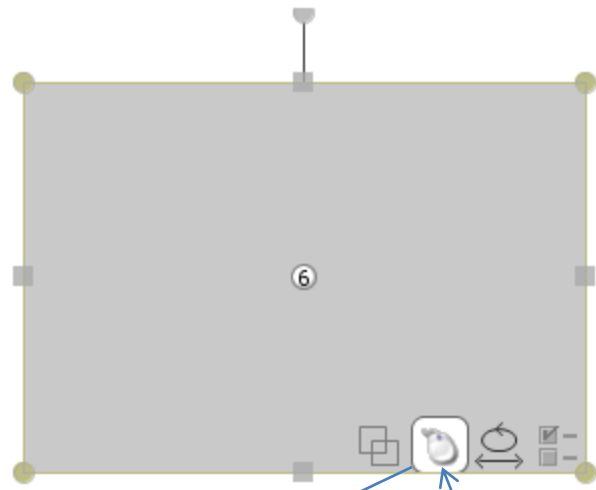


Drop Event Anchors

Anchors for connecting sketch events (on entry, on exit, on variable update, on keyboard event) by drag-and-drop of variables, timers and macros. You can also double-click on these icons to open current settings for these events and properties.



Anchors for connecting region events (region overlap, discrete mouse events, continues mouse events) by drag-and-drop of variables, timers and macros. You can also double-click on these icons to open current settings for these events and properties.



Drag-and-drop of the script on the mouse event icon of the active region.

Mouse Event

Mouse Event: Left Button Press

Action: Start macro

Param1: Script:script.js

Param2:

OK Cancel

Timers Macros Screen Poking Scripts

Script file	Status
script.js	



To Learn More About Scripting Languages

- JavaScript
 - <http://www.w3schools.com/js>
 - <https://developer.mozilla.org/en/JavaScript>
 - [https://developer.mozilla.org/en/A re-introduction to JavaScript](https://developer.mozilla.org/en/A_re-introduction_to_JavaScript)
- BeanShell
 - <http://www.beanshell.org/>
- Groovy
 - <http://groovy.codehaus.org/>
- Python
 - <http://www.python.org/>
- ...