Pause Script Demo 0.3

This is a demonstration of how a script can be written that pauses for the user to do something that would otherwise be difficult or impossible to do from within the script. This might include for example, aesthetic cropping, image alignment, text attribute selection, etc.

One way to handle unscriptable user actions is to use multiple scripts. Run one script. Do whatever is not scriptable then run the second script. That is essentially what the Pause Script Demo does except it eliminates the need for multiple scripts. Basically what you are going to do is to record or write two (or more) scripts and then stitch them together so that all steps are in a single script. When you run the resultant script, it will do the first step and then exit. Running the script again will do the second step and so on until all the steps have been run.

Don't be scared away by all of the script stuff below. It's very easy to use. You just have to do a little bit of editing and almost all of it is cut and paste. The best way to explain it is look at an example and follow along with your own script. First we'll look at a very basic example of pausing and then we'll look at a more complicated example that shows some of the ways you can use the pause code in more detail.

Before we begin though, a few notes and caveats. The pause mechanism uses the currently open image to store the information needed to do the pause and to remember what's going on when the script is run subsequent times. Thus, pausable scripts must have an open image to run on. Also, the information is added to the image description so if you abandon a pausable script in the middle, you will leave some odd looking data there. To get rid of it, run the script again and when it asks you if you want to continue, say No and when it asks you if you want to stop all script steps, say Yes. Finally, if you are in the middle of a pausable script step and need to do something else, you can save the image in .pspimage format and come back to it later. The script information will be preserved and you can pick up where you left off.

Basic Pausing

Step 1: Start by recording two scripts that you want to use before and after the pause.

Step 2: Select the first script in the script list and press the Edit Selected Script button. Then press the Text Editor button.

The beginning of the script is pretty standard and won't change. It will look something like this:

```
from JascApp import *

def ScriptProperties():
    return {
        # Change this to be your own information
        'Author': u'Gary Barton',
```

```
        'Copyright': u'Copyright © 2003 Gary Barton',
        'Description': u'A demo of pausing a script',
        'Host': u'Paint Shop Pro',
        'Host Version': u'8.00'
        }
```

Step 3: Add the next two lines after those above. These lines tell the pauser what the name of the script is. The script name is used to distinguish between this script and another pausable script that might be run while this one is paused. My script is going to be called Demo.

```
# Put the name of your script here:
scriptName = "Demo"
```

For this demo, I'm just going to use some very simple effects, namely Glowing Edges in step one and Kaleidoscope in step two. Your script steps can have as many effects in them as you want.

Step 4: Change the Do routine name to DoStep1. Note, case is important so make sure you use a capital S.

```
def DoStep1(Environment):
    # Glowing Edges
    App.Do( Environment, 'GlowingEdges', {
            'Intensity': 1,
            'Sharpness': 20,
            'GeneralSettings': {
                'ExecutionMode': App.Constants.ExecutionMode.Default,
                'AutoActionMode': App.Constants.AutoActionMode.Match
                }
            })
```

Step 5: Now we are going to add a message to the end of the script so that the user knows what to do when this script step ends.

```
    App.Do(Environment, 'MsgBox', {
        'Buttons': App.Constants.MsgButtons.OK,
        'Icon': App.Constants.MsgIcons.Info,
        'Text': 'Crop the image as desired and then '
                'run the script again to continue.'
        })
```

Step 6: Select the second script and edit it. Copy the Do routine from the second script and paste it in after the routine above. Change its name from Do to DoStep2.

```
def DoStep2(Environment):
    # Kaleidoscope
    App.Do( Environment, 'Kaleidoscope', {
            'Angle': 162,
            'HorizontalOffset': 0,
            'Scale': -39,
            'VerticalOffset': 0,
            'NumOrbits': 0,
```

```
            'RadialSuction': 0,
            'NumPetals': 20,
            'EdgeMode': App.Constants.EdgeMode.Wrap,
            'FillColor': (4,223,51),
            'GeneralSettings': {
                'ExecutionMode': App.Constants.ExecutionMode.Default,
                'AutoActionMode': App.Constants.AutoActionMode.Match
                }
            })
```

Step 7: Finally, we tack on the pause script code which makes the pausing possible. Download this code from the link below. Edit the Pause Script Demo script and copy the pause code from the start to the end markers and place it at the end of your script. The pause code will look like the following:

```
# ----- Pause support start -----
# Author: Gary Barton
# Revision: 0.3
# For updates to this script, visit pixelnook on the web at
#           http://pixelnook.home.comcast.net
#
# Place this code at the end of your script. Rename your original Do
# routine to DoStep1. You may include as many additional steps (DoStep2,
# DoStep3, etc.) as you want. Each step name must begin with 'DoStep'. The
# steps will be done in ALPHABETICAL order. If you really think you are
# going to have more than 9 steps, use DoStep01, DoStep02, ... instead to
# avoid problems with DoStep10 coming right after DoStep1.

try:
    from rexec import r_eval
    REval = r_eval
except:
    REval = eval

def Do(Environment):

... Code continues ...

# ----- Pause support end -----
```

Save the script and run it. It will do the glowing edges and display the message. Click OK and then crop the image if you want to. Run the script again and it will do the kaleidoscope effect.

For reference, here is what the complete script should look like:

```
from JascApp import *

def ScriptProperties():
    return {
        # Change this to be your own information
        'Author': u'Gary Barton',
        'Copyright': u'Copyright © 2003 Gary Barton',
```

```
        'Description': u'A demo of pausing a script',
        'Host': u'Paint Shop Pro',
        'Host Version': u'8.00'
        }

# Put the name of your script here:
scriptName = "Demo"

def DoStep1(Environment):
    # Glowing Edges
    App.Do( Environment, 'GlowingEdges', {
            'Intensity': 1,
            'Sharpness': 20,
            'GeneralSettings': {
                'ExecutionMode': App.Constants.ExecutionMode.Default,
                'AutoActionMode': App.Constants.AutoActionMode.Match
                }
            })

    App.Do(Environment, 'MsgBox', {
        'Buttons': App.Constants.MsgButtons.OK,
        'Icon': App.Constants.MsgIcons.Info,
        'Text': 'Crop the image as desired and then '
                'run the script again to continue.'
        })

def DoStep2(Environment):
    # Kaleidoscope
    App.Do( Environment, 'Kaleidoscope', {
            'Angle': 162,
            'HorizontalOffset': 0,
            'Scale': -39,
            'VerticalOffset': 0,
            'NumOrbits': 0,
            'RadialSuction': 0,
            'NumPetals': 20,
            'EdgeMode': App.Constants.EdgeMode.Wrap,
            'FillColor': (4,223,51),
            'GeneralSettings': {
                'ExecutionMode': App.Constants.ExecutionMode.Default,
                'AutoActionMode': App.Constants.AutoActionMode.Match
                }
            })

# ----- Pause support start -----

... Pause support code ...

# ----- Pause support end -----
```

A more complicated example

In the next section, we will look at the Pause Script Demo script itself. This script shows how to save data in one script step and use it in another. The script asks for some text in step one and uses it in step two.

Stay tuned for part two of this description...

Options to explore:

- Use it for cropping, text attribute selection, text positioning
- Display help or usage reminder messages
- Don't like the result of a step? Undo it and run it again.

Installation instructions:

Download the script. Unzip it and place it in your Scripts-Restricted folder.

To Run:

Open or create a new image. Select the Pause Script Demo script in the scripts list and press play. Enter some text and press return. Now switch to the Text Tool and select the font, size, etc. that you want. Press play to run the script again and it will continue with step two.

Change history:

Changes in 0.1 (10/22/03):

- Original version.

Changes in 0.2 (11/18/03):

- Fixed minor issue when run trusted.

Changes in 0.3 (02/09/04):

- Wasn't cleaning up image description properly
- If you have a different image active at the end of a step, the image active at the beginning of the step is cleaned up.
- No longer requires an open image at the beginning of the first step. You must have an open image by the end of the first step though, or the script will fail.