A script to export a directory of Jpeg files to Xcf format.

Motivation

While it is easy to do, opening a Jpeg file and saving it in the native gimp *.xcf format is tedious and time consuming. You may also want to perform the same operation, like setting the grid spacing as an example, on all of the images that you import as part of the process. Automating the parts of the editing process that are the same for each image saves time, produces consistent results, and allows you to focus your attention on the part of the image editing process that really needs your attention.

This tutorial is a scripting example is intended to illustrate automating part of the editing process. The script opens all of the Jpeg images in the source directory, and saves them as xcf files in the target directory. It is not a basic scripting tutorial, a basic understanding of writing gimp scripts is assumed.

Requirements that are different than most scripts.

1. The script needs to be available and run when there is no image open.
2. The script has to be able to provide a GUI to identify the Source and Target directories, identify all of jpg files in the source directory and convert them.
3. The script needs to be aware of the type of operating system of the host machine, or more specifically whether the filename separator is a “\” or a “/”.
4. The script has to be smart enough to not overwrite work in progress, (Xcf files with the same name in the target directory).

Let's look at the first requirement, to be able to run the script without an open image. We are working on a group of several images not just one, and since we want to start our script from a know point, we don't want to have an image open for edit, just the gimp framework or application.

The availability of a script from within the Gimp application is established and controlled within the script-fu-register block. After the function name are an ordered set of strings of required information followed by an optional number of GUI widget declarations that may feed selected information to the script. The required strings are:

1. **Function Name -** Matches the name of the function in the opening 'define' statement.
2. **Menu Label -** The label identifying the function in the menus and the way to invoke the script.
3. **Description** – The description that will be displayed about the script in the Procedure Browser.
4. **Author** – Name of the Author
5. **Copyright -** Copyright holder and or Date.
6. **Creation Date** – Date you wrote the script.
7. **Valid Image Type -** The type of image the script is permitted to work upon.

The content of the “Valid Image Type” is the way that we specify whether we can run the script without an image open. You might be tempted to use an image type of “*” to specify running all of the time,

but that is not quite what it does. The "Valid Image Type" is a way to restrict the script from running in an unintended context, "*" is the way to specify "any open image". What we want instead is an **empty string - ""** - to specify running without an image open.

Our second requirement, to be able to use the GUI to browse to our Jpeg source file directory and our Xcf target file directory is handled first in the registration block by using the "**SF-DIRNAME**" widgets to label the GUI selection fields and pass the collected results as inputs to the script. Once we have the path names to our source and target directories we will use the "**file-glob**" (os.listdir in python-fu) function within the main program to create a list of all of the image files in those directories.

You should be able to see the Registration Block items that we have discussed so far in the attached example script.

We address the third requirement of determining the operating system type within the function itself. We want the function to be portable, so it can run on either the Linux desktop unit you have in your home, or the Windows laptop you carry out on the road. We need to know the OS type in order to determine the pathname separator, either a "\" or "/" and deal with it in the script.

Fortunately once we have the pathname to the source directory it is easy to determine which operating system generated it by using the script-fu "strbreakup" function. We can compare the number of pieces by breaking on the "/" and "\" character and use the result with the greater number of pieces to determine the platform's operating system. If we are using the python-fu scripting language we can check the value of os.name directly.

Our fourth requirement to not overwrite existing *.xcf files which may have ongoing work in progress is solved by building a list of files in the target directory and checking that list for existence before writing each new file.

Conclusion

The example script is pretty straightforward, building a list of file names, and running through that list in a loop, opening each source file and exporting it in turn. An equivalent python-fu version is included as well. You should be able to use this example as a model to build similar functions, examples might be to convert a directory of xcf files to jpg files, to scale original jpg files to a smaller size, etc.

The code discussed in this tutorial is highlighted in the full script examples.

When using the gimp widgets to Browse through the file system, you will probably need to select "Other" to get to navigate where you really want. When using an automated script it is ALWAYS a good idea to run on a COPY of your original images.

The Scripts

The following scripts are the script-fu and python-fu versions of the jpg to xcf function discussed above

script-fu-example-jpg-to-xcf.scm

```
;=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;;;  Function - script-fu-example-jpg-to-xcf
;;;
;;;  Converts all jpeg images in selected directory to gimp xcf
;;;  format.
```

```
;;;
;;;  Filename Case insensitive. (converts xyz.jpg or XYZ.JPG)
;;;
;;;  Interactive program to be run WITHOUT AN IMAGE LOADED.
;;;
;;;  Program prompts for a source (jpgs) and target (xcfs)
;;;  directories.
;;;
;;;  Program runs on either Linux or Windows Host O/S, using the
;;;  appropriate path - filename separator ( "/" or "\").
;;;
;=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
( define ( script-fu-example-jpg-to-xcf
   sourceDirectory targetDirectory )
   ( let*
      (
         ; Declare and Init local variables
         ( returnVal #f )
         ; Guess host OS based on directory path separator
         ( isLinux (  >
            ( length ( strbreakup sourceDirectory "/" ) )
            ( length ( strbreakup sourceDirectory "\\" ) ) ) )
         ; Form path/file patternSource based on OS
         ( patternSource ( if isLinux
            ( string-append sourceDirectory "/*.[jJ][pP][gG]" )
            ( string-append sourceDirectory "\\*.[jJ][pP][gG]" ) ) )
         ( patternTarget ( if isLinux
            ( string-append targetDirectory "/*.[xX][cC][fF]" )
            ( string-append targetDirectory "\\*.[xX][cC][fF]" ) ) )
         ; List of files to be converted formatted for current Host
         ; O/S
         ( filelistSource ( cadr ( file-glob patternSource 1 ) ) )
         ( filelistExists ( cadr ( file-glob patternTarget 1 ) ) )
         ( checkFileExists filelistExists )
         ; Place holders for image variables - updated per image
         ( theImage 0 )
         ( theDrawable 0 )
         ( currentFile "" )
         ( baseName "" )
         ( outFilename "" )
         ; Constants used to assign values to parasites on new image
         ( doIt #t )
         ( checkFile "" )
      ) ; End declaration of Local Variables
      ;
      ; Run if images closed, message if not.
      ( if ( < 0 ( car ( gimp-image-list ) ) )
         ( gimp-message "Close open Images & Rerun" )
         ( begin
            ;
            ; Run within scope of let* and local variables
            ; 'baseName' is filename without .jpg extension
            ; 'outFilename' is filename with .xcf extension
            ; Step through each file in list with while loop.
            ( while ( not ( null? filelistSource ) )
               ( set! doIt #t )
               ( set! checkFileExists filelistExists )
               ( set! currentFile ( car filelistSource ) )
```

```
                ; Get open and get Image ID of current file
                ( set! theImage
                   ( car ( gimp-file-load RUN-NONINTERACTIVE
                      currentFile currentFile ) ) )
                ( if isLinux
                   ; Target path-filename if Host OS is Linux
                   ( begin
                      ( set! baseName ( car ( reverse
                         ( strbreakup currentFile "/" ) ) ) )
                      ( set! baseName ( car ( strbreakup baseName "." ) ) )
                      ( set! outFilename ( string-append
                           targetDirectory "/" baseName ".xcf" ) )
                   ) ; End begin - Host OS is Linux
                   ; Target path-filename if Host OS is Windows
                   ( begin
                      ( set! baseName ( car ( reverse
                         ( strbreakup currentFile "\\" ) ) ) )
                      ( set! baseName ( car ( strbreakup baseName "." ) ) )
                      ( set! outFilename ( string-append
                           targetDirectory "\\" baseName ".xcf" ) )
                   ) ; End begin - if Host OS is Windows
                ) ; End if isLinux
                ; Check to see if outFilename exists so we don't overwrite
                ( while ( not ( null? checkFileExists ) )
                   ( set! checkFile ( car checkFileExists ) )
                   ( if ( string=? outFilename checkFile )
                      ( set! doIt #f ) )
                   (set! checkFileExists ( cdr checkFileExists ) )
                ) ; End while checkFileExists
                ( if doIt
                   ( begin
                      ; Get / set Drawable ID, need it for file save.
                      ( set! theDrawable ( car
                         ( gimp-image-merge-visible-layers theImage 0 ) ) )
                      ; Save file - gimp xcf format
                      ( gimp-xcf-save
                         RUN-NONINTERACTIVE theImage theDrawable
                         outFilename outFilename )
                   ) ; End begin
                ) ; End if doIt
                ( gimp-image-delete theImage )
                ; Update while loop iteration parameter
                (set! filelistSource ( cdr filelistSource ) )
             ) ; End while
          ); End outer begin
       ) ; End outer if
       ( set! returnVal #t )
    ) ; End let*
) ; End define
;
( script-fu-register "script-fu-example-jpg-to-xcf" ; Function Name
    "1 ) Import JPG to XCF (Directory)"    ; Menu Label
    "This script is an interactive script to convert all of the jpegs
    in a source directory into Gimp xcf format files in a target
    directory.  The script is designed to be run WITHOUT ANY IMAGE
    LOADED. Runs from Gimp shell in Linux and Windows."
    "Stephen Kiel"       ; Author
    "2013, Stephen Kiel" ; Copyright
```

```
   "June 2013"          ; Creation Date
   ""                   ; Valid Image Type - No Image required
    ; We actually don't want any images open when we run this
    ;   script, so it must be available from the menu when an
    ;   image is not loaded.  This script will determine the IDs
    ;   of the Image and Drawable itself rather than having them
    ;   passed as parameters.
   ; Interactive widgets
   SF-DIRNAME "JPG Originals (source) Directory" ""
   SF-DIRNAME "XCF Working   (target) Directory" ""
) ; End script-fu-register
( script-fu-menu-register
   "script-fu-example-jpg-to-xcf" "<Image>/Example-Scm")
```

example-jpeg-to-xcf.py

```
#! /usr/bin/env python
from gimpfu import *
import os
import re

########################################################################

def exampleJpgToXcf(srcPath, tgtPath):
    """Registered function exampleJpgToXcf, Converts all of the
    jpegs in the source directory into xcf files in a target
    directory.  Requires two arguments, the paths to the source and
    target directories.  DOES NOT require an image to be open.
    """
    ###
    open_images, image_ids = pdb.gimp_image_list()
    if open_images > 0:
        pdb.gimp_message ("Close open Images & Rerun")
    else:
        allFileList = os.listdir(srcPath)
        existingList = os.listdir(tgtPath)
        srcFileList = []
        tgtFileList = []
        xform = re.compile('\.jpg', re.IGNORECASE)
        # Find all of the jpeg files in the list & make xcf file names
        for fname in allFileList:
            fnameLow = fname.lower()
            if fnameLow.count('.jpg') > 0:
                srcFileList.append(fname)
                tgtFileList.append(xform.sub('.xcf',fname))

        # Dictionary - source & target file names
        tgtFileDict = dict(zip(srcFileList, tgtFileList))
        # use os.name to yield a platform based path name w/ separator
        # e.g. '/' for Linux, '\' for Windows.
        if os.name == 'posix':
            srcPath = srcPath + '/'
            tgtPath = tgtPath + '/'
        elif os.name == 'nt':
            srcPath = srcPath + '\\'
            tgtPath = tgtPath + '\\'
        # Loop on jpegs, open each & save as xcf
```

```
        for srcFile in srcFileList:
            # Don't overwrite existing, might be work in Progress
            if tgtFileDict[srcFile] not in existingList:
                tgtFile = tgtPath + tgtFileDict[srcFile]
                srcFile = srcPath + srcFile
                theImage = pdb.file_jpeg_load(srcFile, srcFile)
                theDrawable = theImage.active_drawable
                pdb.gimp_xcf_save(0, theImage, theDrawable, tgtFile, tgtFile)
                pdb.gimp_image_delete(theImage)

########################################################################

register (
    "exampleJpgToXcf",         # Name registered in Procedure Browser
    "Convert jpg files to xcf", # Widget title
    "Convert jpg files to xcf", #
    "Stephen Kiel",          # Author
    "Stephen Kiel",          # Copyright Holder
    "June 2013",             # Date
    "1) Import JPG to XCF (Directory)", # Menu Entry
    "",     # Image Type
    [
    ( PF_DIRNAME, "srcPath", "JPG Originals (source) Directory:", "" ),
    ( PF_DIRNAME, "tgtPath", "XCF Working (target) Directory:", "" ),
    ],
    [],
    exampleJpgToXcf,   # Matches to name of function being defined
    menu = "<Image>/Example-Py"  # Menu Location
    )   # End register

main()
```