Python - import all image sequences under folder

```
"""Finds and creates nodes for file sequences in an entire directory tree"""
# This has the user select a folder, then creates Read nodes
# for all files and sequences under that folder
# lewis@lewissaunders.com 03/2011

import os, nuke, re, time

from curses.ascii import isprint

# Returns a printable version of the input
def printable(input):
    return ''.join(char for char in input if isprint(char))

# Returns a list of all folder paths beneath the given head folder
def folders(head):
    head = head.rstrip('/')

    ret = []
    ret.append(head)
    for (root, dirs, files) in os.walk(head, followlinks=True):
        for d in dirs:
            ret.append(root + '/' + d)
    return ret

# Returns a Nuke-style list of image sequences under the given folder
def seqs(path):

    prog = nuke.ProgressTask("Looking...")
    prog.setProgress(0)

    all_folders = folders(path)
```

```
    chunk_list = []

    i = 0
    for f in all_folders:
        prog.setMessage("Looking under " + os.path.basename(f))
        prog.setProgress((100 * i)/len(all_folders))

        time.sleep(0.01)

        raw = nuke.tcl('filename_list -compress "' + f + '"')

        if raw == None:
            continue

        # Little state machine which munches through the string,
        # working out how to separate it
        inside = False        # True when current char is inside {..} pair
        sep = False        # True when the current char is { or } or space
        prev_inside = False
        prev_sep = True
        chunk = f + '/'

        for c in raw:
            if c == '{':
                sep = True
                inside = True
            elif c == '}':
                sep = True
                inside = False
            elif c == ' ' and not inside:
                sep = True
            elif c == ' ' and inside:
                sep = False
            else:
                sep = False

            if sep:
                if not prev_sep:
                    # End of word
                    chunk_list.append(chunk)
                    chunk = f + '/'
```

```
            else:
                # Start or middle of chunk
                chunk = chunk + c

            prev_sep = sep
            prev_inside = inside

        # Get final chunk
        chunk_list.append(chunk)
        i = i + 1

    # Lose folders and .ifds, keep files
    chunk_list_no_folders = []
    for f in chunk_list:
        if os.path.isdir(f):
            continue
        if re.search('\.ifd ', f) != None:
            continue
        if re.search('\.ifd.old', f) != None:
            continue
        if printable(f) != f:
            # Non-ASCII characters - skip it
            continue
        chunk_list_no_folders.append(f)
    chunk_list_no_folders.sort()

    del prog

    return chunk_list_no_folders

# Prompt for a folder then create a pile of Read nodes
def doit():
    # Get the selected node, and the path on it's 'file' knob if it
    # has one, or failing that, it's 'proxy' knob, if it has that.
    sel_node = None
    default_dir = None
    try:
        sel_node = nuke.selectedNode()
    except:
        pass
    if ( sel_node is not None ) and ( sel_node != '' ):
```

```
        if 'file' in sel_node.knobs():
            default_dir = sel_node['file'].value()
        if (default_dir == None or default_dir == '') and 'proxy' in sel_node.knobs():
            default_dir = sel_node['proxy'].value()

    # Revert default_dir to None if it's empty so that the file browser
    # will do it's default unset behaviour rather than open on an empty path.
    if default_dir == '':
        default_dir = None



    #
    # (Above lines robbed from nukescripts.create)
    #



    p = nuke.getClipname("Read folder recursively...", default=default_dir, multiple=False)

    if p == None:
        return

    sequences = seqs(p)

    # Pops up a panel with an editable textbox with all the paths
    lines = ''
    for s in sequences:
        lines = lines + s + '\n'
    p = nuke.Panel('Found these files...')
    p.addMultilineTextInput('Found:', lines)
    p.setWidth(1720)
    if not p.show():
        return

    for f in p.value('Found:').splitlines():
        if f != '':
            nuke.createNode('Read', 'file {%s}' % f)

doit()
```

Revision #1
Created 2022-11-21 10:03:31 UTC by Anthony
Updated 2022-11-21 10:03:54 UTC by Anthony