OpenEXR Documentation

Release 1.0

James Bowman

May 25, 2009

CONTENTS

CHAPTER

ONE

INTRODUCTION

OpenEXR is an image format developed by ILM. Its main innovation is support for high dynamic range; it supports floating point pixels.

This module provides Python bindings for the OpenEXR C++ libraries. They allow you to read and write OpenEXR files from Python.

Note that this module only loads and stores images. It does not do any image manipulation operations. For that you might want to use one of:

- Python's standard array module. You can access the raw data of FLOAT and UINT images.
- The Python Imaging Library. This library supports a single FLOAT channel image format.
- Numeric or NumPy. It's just math, so you will have to write your own imaging operations. Supports UINT and FLOAT formats.
- Module vop. NumPy subset, but faster. Supports FLOAT and HALF.
- OpenCV. Supports multi channel UINT and FLOAT formats.

CHAPTER

TWO

OPENEXR — READ AND WRITE EXR FORMAT IMAGES

2.1 Available Types

class InputFile()

A InputFile object is used to read an EXR file.

```
>>> import OpenEXR
>>> print len(OpenEXR.InputFile("GoldenGate.exr").channel('R')), "bytes of red data"
2170640 bytes of red data
```

The following data items and methods are supported:

header()

Return the header of the open file. The header is a dictionary as described below.

channel(*cname, [pixel_type, [scanLine1, [scanLine2]]]*)

Read a channel from the OpenEXR image. *cname* is the name of the channel in the image, for example "R". This method reads the channel data in the format specified by *pixel_type* (see Imath.PixelType), or the format of the data specified in the image itself by default. If *scanLine1* and *scanLine2* are not supplied, then the method reads the entire image. Note that this method returns the channel as a Python string: the caller must then convert it to the appropriate format as necessary.

close()

Close the open file. Calling this method is mandatory, otherwise the file will be incomplete. However, as a convenience, the object's destructor calls this method, so any open files are automatically closed at program exit.

class OutputFile(*filename, header*)

Creates the EXR file *filename*, with given *header*. *header* contains the image's properties and is a dict created by Header().

```
>>> import OpenEXR, array
>>> data = array.array('f', [ 1.0 ] * (640 * 480)).tostring()
>>> exr = OpenEXR.OutputFile("out.exr", OpenEXR.Header(640,480))
>>> exr.writePixels({'R': data, 'G': data, 'B': data})
```

The following data items and methods are supported:

writePixels(*dict, [scanlines]*)

Write the specified channels to the OpenEXR image. *dict* specifies multiple channels. If *scanlines* is not specified, then the entire image is assumed. dict specifies each channel's data as channel:data, where channel and data are both strings. This method uses the file's header to determine the format of the data

(FLOAT, HALF or UINT) for each channel. If the string data is not of the appropriate size, this method raises an exception.

currentScanLine()
Return the current scan line being written.

close()
Close the open file. This method may be called multiple times. As a convenience, the object's destructor calls this method.

2.2 Available Functions

isOpenExrFile(*filename*)
Returns True if the *filename* exists, is readable, and contains a valid EXR image.

```
>>> import OpenEXR
>>> print OpenEXR.isOpenExrFile("no-such-file")
False
>>> print OpenEXR.isOpenExrFile("lena.jpg")
False
>>> print OpenEXR.isOpenExrFile("GoldenGate.exr")
True
```

Header(*width, height*)
Convenience function that creates the EXR header for an image of size *width* x *height* with EXR mandatory entries set to appropriate defaults. An EXR header is a dictionary - see *EXR header values* for details of legal header contents.

```
>>> import OpenEXR
>>> print OpenEXR.Header(640,480)
{'compression': ZIP_COMPRESSION,
 'pixelAspectRatio': 1.0,
 'displayWindow': (0, 0) - (639, 479),
 'channels': {'R': FLOAT (1, 1), 'B': FLOAT (1, 1), 'G': FLOAT (1, 1)},
 'dataWindow': (0, 0) - (639, 479),
 'screenWindowCenter': (0.0, 0.0),
 'screenWindowWidth': 1.0,
 'lineOrder': INCREASING_Y}
```

2.3 EXR header values

This module represents EXR headers as regular Python dictionaries. In this dictionary the keys are strings, and the values are such that OpenEXR can determine their type. The module `Imath` provides many of the classes for attribute types.

```
>>> import OpenEXR
>>> print OpenEXR.InputFile("GoldenGate.exr").header()
{'tiles': None,
 'capDate': '2004:01:04 18:10:00',
 'compression': PIZ_COMPRESSION,
 'latitude': 37.827701568603516,
 'pixelAspectRatio': 1.0,
 'altitude': 274.5,
```

```
'displayWindow': (0, 0) - (1261, 859),
'focus': inf,
'comments': 'View from Hawk Hill towards San Francisco',
'screenWindowWidth': 1.1499999761581421,
'channels': {'R': HALF (1, 1), 'B': HALF (1, 1), 'G': HALF (1, 1)},
'isoSpeed': 50.0,
'utcOffset': 28800.0,
'longitude': -122.49960327148438,
'dataWindow': (0, 0) - (1261, 859),
'screenWindowCenter': (0.0, 0.0),
'aperture': 2.7999999523162842,
'preview': <Imath.PreviewImage instance 100x68>,
'owner': 'Copyright 2004 Industrial Light & Magic',
'expTime': 8.0,
'lineOrder': INCREASING_Y}
```

Values in the dictionary may be: string

```
header['owner'] = 'Copyright 2007 James Bowman'
```

float

```
header['isoSpeed'] = 50.0
```

int

```
header['version'] = 1001
```

dict

A dict represents the image's channels. In the dict, the keys are the channel names, and the values are of class `Imath.Channel`:

```
header['channels'] = { 'L' : Imath.Channel(PixelType(OpenEXR.HALF)),
                       'Z' : Imath.Channel(PixelType(OpenEXR.FLOAT))}
```

`Imath.Box2i`

```
header['dataWindow'] = Imath.Box2i(Imath.point(0,0), Imath.point(640,480))
```

`Imath.Box2f`

```
header['regionOfInterest'] = Imath.Box2f(Imath.point(75.0,75.0),
                                         Imath.point(100.0,100.0))
```

`Imath.V2f`

```
header['originMarker'] = Imath.point(0.378, 0.878)
```

`Imath.LineOrder`

```
header['lineOrder'] = Imath.LineOrder(Imath.LineOrder.INCREASING_Y)
```

`Imath.PreviewImage`

A preview image, specified by height, width, and a string of length 4*width*height. The pixels are in RGBA order.:

```
header['preview'] = Imath.PreviewImage(320,200,pixels)
```

or to use a PIL image as an EXR preview:

```
header['preview'] = Imath.PreviewImage(im.size[0], im.size[1], im.convert("RGBA").tostring()
```

`Imath.Compression`

```
header['Compression'] = Imath.Compression(Imath.Compression.PIZ_COMPRESSION)
```

CHAPTER

THREE

IMATH — SUPPORT TYPES FOR OPENEXR LIBRARY

class Box (*min=None, max=None*)
Box is a 2D box, specified by its two corners *min* and *max*.

class Box2f (*min=None, max=None*)
Box2f is a 2D box, specified by its two corners *min* and *max*.

class Box2i (*min=None, max=None*)
Box2i is a 2D box, specified by its two corners *min* and *max*.

class Channel (*type=HALF, xSampling=1, ySampling=1*)
Channel defines the type and spatial layout of a channel. *type* is a `PixelType`. *xSampling* is the number of X-axis pixels between samples. *ySampling* is the number of Y-axis pixels between samples.

```
>>> import Imath
>>> print Imath.Channel(Imath.PixelType(Imath.PixelType.FLOAT), 4, 4)
FLOAT (4, 4)
```

class Compression (*v*)
Compression can have possible values: `NO_COMPRESSION`, `RLE_COMPRESSION`, `ZIPS_COMPRESSION`, `ZIP_COMPRESSION`, `PIZ_COMPRESSION`, `PXR24_COMPRESSION`.

```
>>> import Imath
>>> print Imath.Compression(Imath.Compression.RLE_COMPRESSION)
RLE_COMPRESSION
```

class LineOrder (*v*)
LineOrder can have three possible values: `INCREASING_Y`, `DECREASING_Y`, `RANDOM_Y`.

```
>>> import Imath
>>> print Imath.LineOrder(Imath.LineOrder.DECREASING_Y)
DECREASING_Y
```

class PixelType (*v*)
PixelType can have possible values `UINT`, `HALF`, `FLOAT`.

```
>>> import Imath
>>> print Imath.PixelType(Imath.PixelType.HALF)
HALF
```

class `PreviewImage` (*width, height, pixels*)

PreviewImage is a small preview image, intended as a thumbnail version of the full image. The image has size (*width*, *height*) and 8-bit pixel values are given by string *pixels* in RGBA order from top-left to bottom-right.

For example, to create a preview image from a JPEG file using the popular Python Imaging Library:

```
>>> import Image
>>> import Imath
>>> im = Image.open("lena.jpg").resize((100, 100)).convert("RGBA")
>>> print Imath.PreviewImage(im.size[0], im.size[1], im.tostring())
<Imath.PreviewImage instance 100x100>
```

class `V2f` (*x, y*)

V2f is a 2D point, with members *x* and *y*.

class `V2i` (*x, y*)

V2i is a 2D point, with members *x* and *y*.

class `point` (*x, y*)

Point is a 2D point, with members *x* and *y*.

CHAPTER

FOUR

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*

MODULE INDEX

I

O

INDEX

T

U

V

W

Z