avifilelib Documentation

Release 0.9.1

Michael Uhl

Apr 01, 2018

Contents

avifilelib provides a mechanism to load frames from avifiles.

The primary user class in this library is `avifilelib.AviFile`.

The following is a simple example showing how to iterate over the frames in an AVI file:

```
>>> import matplotlib
>>> matplotlib.use('TKAGG')
>>> import matplotlib.pyplot as plt
>>> import avifilelib
>>> a = avifilelib.AviFile('sample.avi')
>>> for ct, frame in enumerate(a.iter_frames(stream_id=0)):
        _ = plt.imshow(frame, origin='lower')
        plt.gcf().savefig('frame_{:02d}.png'.format(ct))
>>> a.close()
```

Contents:

CHAPTER 1

avifilelib package

1.1 Submodules

1.1.1 avifilelib.avi module

AVI File object

This module contains the AviFile class.

class avifilelib.avi.**AviFile**(*file_or_filename*)

Bases: object

An AVI file

Parameters **file_or_filename** – An open file (or file-like object) or a string containing a file name

avih

Convenience alias for self.avi_header

close()

Close the underlying file.

idx1

Convenience alias for self.index_v1

iter_frames(*stream_id*)

Iterate over the frames in stream *stream_id*

Parameters **stream_id** (*int*) – Id of the stream to iterate over.

movi

Convenience alias for self.stream_content

strl

Convenience alias for self.stream_definitions

class avifilelib.avi.**AviFileHeader**(*micro_sec_per_frame*, *max_bytes_per_sec*, *padding_granularity*, *flags*, *total_frames*, *initial_frames*, *streams*, *suggested_buffer_size*, *width*, *height*, *reserved*)

Bases: object

AVIMAINHEADER structure

Parameters

- **micro_sec_per_frame** (*int*) – Frame timing
- **max_bytes_per_sec** (*int*) – Data rate
- **padding_granularity** (*int*) – Pad data to multiples of this value
- **flags** (*avifilelib.enums.AVIF*) – Flags
- **total_frames** (*int*) – Total number of frames
- **initial_frames** (*int*) – See AVIMAINHEADER
- **streams** (*int*) – Number of streams
- **suggested_buffer_size** (*int*) – Suggested buffer size in bytes
- **width** (*int*) – Frame width in pixels
- **height** (*int*) – Frame height in pixels
- **reserved** (*list*) – A list of four integers

classmethod load(*file_like*)

1.1.2 avifilelib.bmp module

Decoders for Microsoft BMP formats.

This package provides decoders capable of decoding frames encoded using the Microsoft Bitmap formats.

class avifilelib.bmp.**BMP16Decoder**(*width*, *height*, *colors=None*)

Bases: *avifilelib.bmp.BMPDecoderBase*

Decoder for 16-bit bitmaps.

This class implements a decoder for the 16-bit bitmaps.

BIT_COUNT = 16

BLUE_MASK = 31

BLUE_SHIFT = 0

COMPRESSION = 0

GREEN_MASK = 992

GREEN_SHIFT = 5

RED_MASK = 31744

RED_SHIFT = 10

decode_frame_buffer(*buffer*, *size*, *keyframe=True*)

Decode a frame from a bytes object.

Decodes a single frame from the data contained in *buffer*.

Parameters

- **buffer** (*bytes*) – A bytes object containing the frame data.
- **size** (*int*) – Size of the data in the buffer.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

class avifilelib.bmp.**BMP24Decoder**(*width*, *height*, *colors=None*)
Bases: *avifilelib.bmp.BMPDecoderBase*

Decoder for 24-bit bitmaps.

This class implements a decoder for the 24-bit bitmaps.

BIT_COUNT = 24

COMPRESSION = 0

decode_frame_buffer(*buffer*, *size*, *keyframe=True*)
Decode a frame from a bytes object.

Decodes a single frame from the data contained in *buffer*.

Parameters

- **buffer** (*bytes*) – A bytes object containing the frame data.
- **size** (*int*) – Size of the data in the buffer.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

class avifilelib.bmp.**BMP32Decoder**(*width*, *height*, *colors=None*)
Bases: *avifilelib.bmp.BMPDecoderBase*

Decoder for 32-bit bitmaps.

This class implements a decoder for the 32-bit bitmaps.

BIT_COUNT = 32

COMPRESSION = 0

decode_frame_buffer(*buffer*, *size*, *keyframe=True*)
Decode a frame from a bytes object.

Decodes a single frame from the data contained in *buffer*.

Parameters

- **buffer** (*bytes*) – A bytes object containing the frame data.
- **size** (*int*) – Size of the data in the buffer.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

class avifilelib.bmp.**BMP8Decoder**(*width*, *height*, *colors=None*)
Bases: *avifilelib.bmp.BMPDecoderBase*

Decoder for 8-bit bitmaps.

This class implements a decoder for the 8-bit bitmaps.

BIT_COUNT = 8

COMPRESSION = 0

decode_frame_buffer(*buffer*, *size*, *keyframe=True*)
Decode a frame from a bytes object.

Decodes a single frame from the data contained in *buffer*.

Parameters

- **buffer** (*bytes*) – A bytes object containing the frame data.
- **size** (*int*) – Size of the data in the buffer.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

class avifilelib.bmp.**BMPDecoderBase**(*width*, *height*, *colors=None*)
Bases: *avifilelib.decoder.DecoderBase*

Base class for bitmap decoders.

Parameters

- **width** (*int*) – Width of the image to be decoded.
- **height** (*int*) – Height of the image to be decoded.
- **colors** (*numpy.ndarray, dtype=uint8*) – N x 3 of red, green, and blue values, where N is 2^4 or 2^8 depending on the type of RLE compression.

COMPRESSION = (<BI_COMPRESSION.BI_RGB: 0>,)

classmethod for_avi_stream(*stream_definition*)
Attempts to find a decoder implementation for a stream.

Subclasses of *BMPDecoderBase* are selected by matching the value of *BIT_COUNT*.

Returns A subclass of *BMPDecoderBase*.

Return type object

image
Gets a copy of the image.

class avifilelib.bmp.**BMPFileHeader**(*type*, *size*, *reserved1*, *reserved2*, *offbits*)
Bases: object

BITMAPFILEHEADER structure.

Parameters

- **type** (*bytes*) – must be b'BM'
- **size** (*int*) – Size of the bitmap data
- **reserved1** (*int*) – Reserved value, must be zero (but not checked here)
- **reserved2** (*int*) – Reserved value, must be zero (but not checked here)
- **offbits** (*int*) – offset from the beginning of the header to start of the bitmap data

classmethod from_buffer(*buffer*)

classmethod from_file(*file*)

class avifilelib.bmp.**BMP_DRAW_ORDER**
Bases: enum.Enum

Bitmap drawing orders.

BOTTOM_UP = 1

TOP_DOWN = 2

1.1.3 avifilelib.data module

AVI Stream Data classes.

This module contains classes for handling the stream data within an AVI file. These classes include *AviMoviList* which represents the list structure containing stream data within the AVI file. The 'movi' list may optionally contain 'rec ' lists (represented by the *AviRecList* class). 'rec ' lists are used to group stream data chunks to indicate that they should all be read from disk at the same time. This library does not preload data, and therefore, does not take any special action based on the presence of 'rec ' lists within the 'movi' list. Further, upon the location and parsing of a 'rec ' list within an AVI file, *avifilelib* simply adds the data chunks contained in the 'rec ' list directly into the 'movi' list.

Finally, this module provides the *AviStreamChunk* class to represent a chunk of stream data within the AVI file. Note that the *flags* applicable to a stream chunk are identified in the *avifilelib.index.AviV1Index*, and therefore will not normally be available when AviStreamChunks are created.

class avifilelib.data.**AviMoviList**(*absolute_offset=0*, *data_chunks=None*)
Bases: object

Used to read the 'movi' list within an AVI file.

This class is used to facilitate the loading of stream data chunks contained with the 'movi' list and to make those chunks acessible for later decoding. Because the AVI index structure identifies chunks by their offset from the start of the 'movi' list, the absolute offset of the start of the 'movi' list is required in order to compute the relative offsets of the contained data chunks.

Parameters

- **absolute_offset** (*int*) – Absolute offset in bytes of the the start of the 'movi' list data section from the start of the underlying file.
- **data_chunks** (*list*) – A list of *AviStreamChunk* objects.

apply_index(*index*)
Apply flags and skipping as defined by an AVI Index.

Parameters **index** (*avifilelib.index.AviV1Index*) – An index containing AVI indexing information.

iter_chunks(*stream=None*)
Return an iterator over the chunks in a stream.

Parameters **stream** (*int*) – Stream identifier of the stream to be iterated over.

Returns A generator that iterates over the chunks in a stream.

Return type generator

classmethod load(*file_like*)
Create an AviMoviList structure.

This method creates an *AviMoviList* from the contents of an AVI 'movi' list. Note that this does not actually load the data associated with the contained stream chunks.

Parameters **file_like** (*file-like*) – A file-like object positioned at the start of a 'movi' list.

Returns An *AviMoviList* containing *AviStreamChunk* objects.

Return type *AviMoviList*

class avifilelib.data.**AviRecList**(*data_chunks=None*)

Bases: object

Used to read 'rec ' lists within an AVI file.

Note that this class is used only to facilitate the loading of stream data chunks contained with 'rec ' lists. *avifilelib* does not maintain the 'rec ' list structure after the stream data chunks have been identified. All stream data chunks contained within 'rec ' lists are reparented to belong to the 'movi' (represented by an *AviMoviList*) instead.

Parameters **data_chunks** (*list*) – A list containing *AviStreamChunk* objects.

classmethod load(*file_like*)

Create an AviRecList structure.

This method creates an *AviRecList* from the contents of an AVI 'rec ' list. Note that this does not actually load the data associated with the contained stream chunks.

Parameters **file_like** (*file-like*) – A file-like object positioned at the start of a 'rec ' list.

Returns An *AviRecList* containing *AviStreamChunk* objects.

Return type *AviRecList*

class avifilelib.data.**AviStreamChunk**(*stream_id*, *data_type*, *base_file*, *absolute_offset*, *size*, *flags=0*, *skip=False*)

Bases: object

A block of data representing a portion of an audio or video stream.

For a video stream, a stream chunk would typically represent a single frame. This class does not load the data into memory, but does provide an interface to read the data associated with the chunk from the file system.

Parameters

- **stream_id** (*int*) – Identifier of the stream.
- **data_type** (STREAM_DATA_TYPES) – Identifies the kind of data stored in the chunk.
- **base_file** (*file-like*) – File-like object from which the data should be read.
- **absolute_offset** (*int*) – Offset from the start of the 'MOVI' list.
- **size** (*int*) – Size of the chunk.
- **flags** (AVIIF) – Flags associated with the frame.
- **skip** (*bool*) – If *True* the chunk will be skipped when iterating over the chunks.

flags

Get the AVIIF flags for the chunks.

classmethod load(*file_like*)

Create an AviStreamChunk structure.

This method creates an *AviStreamChunk* from the contents of an AVI 'movi' or 'rec ' list. Note that this does not actually load the data associated with the stream chunk.

Parameters **file_like** (*file-like*) – A file-like object positioned at the start of a stream data chunk.

Returns An *AviStreamChunk* that may be used to read the data for this chunk.

Return type *AviStreamChunk*

read(*size=-1*)
Read *size* bytes from the underlying file.

seek(*pos*, *whence=0*)
Change the stream position to the given byte *pos*. *pos* is interpreted relative to the position indicated by *whence*. The default value for *whence* is *SEEK_SET*. Values for *whence* are:

- 0 – start of the chunk (the default); offset should be zero or positive
- 1 – current chunk position; offset may be negative
- 2 – end of the chunk; offset is usually negative

Returns the new absolute position relative to the start of the stream chunk.

Return type int

tell()
Return the current position in the chunk.

1.1.4 avifilelib.decoder module

Base class for video stream decoders.

This module defines the base class for video decoders and provides a utility function that is used by a number of the included decoders.

class avifilelib.decoder.**DecoderBase**(*width*, *height*)
Bases: object

Base class for video Decoder objects.

Parameters

- **width** (*int*) – Width of the image to be decoded.
- **height** (*int*) – Height of the image to be decoded.

decode_frame_buffer(*buffer*, *size*, *keyframe=True*)
Decode a frame from a bytes object.

This method has no implementation in DecoderBase and raises a NotImplementedError. This method must be implemented by subclasses of *DecoderBase*.

Parameters

- **buffer** (*bytes*) – A bytes object containing the frame data.
- **size** (*int*) – Size of the data in the buffer. Some formats write 2-byte aligned chunks, and therefore, the data size need not equal the length of *buffer*.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

decode_frame_chunk(*stream_chunk*, *keyframe=False*)
Decode a frame from a RIFF chunk.

Parameters

- **stream_chunk** (avifilelib.avi.AviStreamChunk) – A data chunk that contains frame data.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

classmethod for_avi_stream(*stream_definition*)
Attempts to find a decoder implementation for a stream.

This method searches DecoderBase subclasses for a subclass capable of handling the stream format defined in *stream_definition*. Matches are made by comparing the *stream_definition.stream_header.compression* field to the *COMPRESSION* member of a *DecoderBase* subclass. The *COMPRESSION* member must be a member of the *avifilelib.enums.BI_COMPRESSION* enumeration, or a tuple of such values. Thus, if a particular compression method supports multiple subformats, it is recommended that a subclass base for that compression method be written, and the *for_avi_stream()* method of the subclass be overridden to handle further delegation.

Parameters **stream_definition** (avifilelib.avi.AviStreamDefinition) – Stream definition for the stream to be decoded.

Returns A subclass of *DecoderBase* capable of decoding a stream compressed according to *stream_definition*.

Return type object

height
Gets the height of the image.

image
Gets a copy of the image.

width
Gets the width of the image.

avifilelib.decoder.**chunkwise**(*iterable*, *count=2*, *fill_value=None*)
Iterate over *count*-size chunks of an iterable.

Parameters

- **iterable** (*iterable*) – An object that can be iterated over.
- **count** (*int*) – Size of the chunks to be returned.
- **fill_value** (*object*) – Value to be use as a filler if *iterable* is not divisible by *count*.

Returns An iterable object which yields *count*-tuples.

Return type zip_longest

1.1.5 avifilelib.definition module

Classes related to stream definitions.

This module contains classes related to stream defintions and headers.

class avifilelib.definition.**AviJunkChunk**
Bases: object

Consumes a Junk chunk.

classmethod load(*file_like*)
Consumes a junk chunk.

Parameters **file_like** (*file-like*) – A file-like object containing a 'JUNK' chunk.

class avifilelib.definition.**AviStreamData**(*raw_bytes*)
Bases: object

Data about a stream.

A stream defintion may contain additional data about a stream within a 'strd' chunk. No format is specified for the data. The data is stored in the *raw_bytes* member of the instance.

Parameters **raw_bytes** (*bytes*) – The data associated with the stream data chunk.

classmethod load(*file_like*)
Create a new *AviStreamData* instance.

Creates a new instance from a file-like object positioned at the start of a 'strd' chunk.

Parameters **file_like** (*file-like*) – A file-like object containing a 'strd' chunk.

Returns Stream data instance for this stream.

Return type *AviStreamData*

class avifilelib.definition.**AviStreamDefinition**(*stream_id*, *stream_header*, *stream_format*, *stream_data=None*, *stream_name=None*)
Bases: object

A container for the data related to stream definitions.

This class contains the information of the 'strl' list in an AVI file.

Parameters

- **stream_id** (*int*) – The id number of the stream.
- **stream_header** (*AviStreamHeader*) – Stream header data for this stream definition.
- **stream_format** (*AviStreamFormat*) – An *AviStreamFormat* (or subclass thereof) defining the format for the stream.
- **stream_data** (*AviStreamData*) – Optionally, an instance of *AviStreamData* (or subclass thereof).
- **stream_name** (*AviStreamName*) – Optionally, an instance of *AviStreamName* (or subclass thereof).

classmethod load(*stream_id*, *file_like*)
Create an *AviStreamDefinition*

This method creates a new *AviStreamDefinition* from the contents of an AVI 'strl' list.

Parameters

- **stream_id** (*int*) – The id number of the stream.
- **file_like** (*file-like*) – A file-like object positioned at the start of a 'strl' list.

Returns

Return type *AviStreamDefinition*

strd
Get the stream data chunk.

strf
Get the stream format.

strh
Get the stream_header.

strn
Get the stream name.

class avifilelib.definition.**AviStreamFormat**
Bases: object

Base class for stream format classes.

This class provides the base for concrete implementations of stream format classes.

classmethod load(*stream_header*, *file_like*)
Create an *AviStreamFormat* subclass

This method creates a new instance of a *AviStreamFormat* from the contents of an AVI 'strf' list. Subclasses are selected by matching the *FCC_TYPE* member of the class to the *fcc_type* member of the *stream_header*.

Parameters

- **stream_header** (*AviStreamHeader*) – Header associated with the stream.
- **file_like** (*file-like*) – A file-like object positioned at the start of a 'strh' list.

Returns Instance of a *AviStreamFormat* subclass.

Return type object

class avifilelib.definition.**AviStreamHeader**(*fcc_type*, *fcc_handler*, *flags*, *priority*, *language*, *initial_frames*, *scale*, *rate*, *start*, *length*, *suggested_buffer_size*, *quality*, *sample_size*, *frame*)
Bases: object

An AVI Stream Header.

This class represents the AVISTREAMHEADER structure.

classmethod load(*file_like*)
Create an *AviStreamHeader*

This method creates a new *AviStreamHeader* from the contents of an AVI 'strh' list.

Parameters **file_like** (*file-like*) – A file-like object positioned at the start of a 'strh' list.

Returns

Return type *AviStreamHeader*

class avifilelib.definition.**AviStreamName**(*name*)
Bases: object

Name of the stream.

Parameters **name** (*str*) – Stream name

classmethod load(*file_like*)
Create a new *AviStreamName* instance.

Creates a new instance from a file-like object positioned at the start of a 'strn' chunk.

Parameters **file_like** (*file-like*) – A file-like object containing a 'strn' chunk.

Returns Stream data instance for this stream.

Return type *AviStreamName*

class avifilelib.definition.**BitmapInfoHeaders**(*size*, *width*, *height*, *planes*, *bit_count*, *compression*, *size_image*, *x_pels_per_meter*, *y_pels_per_meter*, *clr_used*, *clr_important*)
Bases: *avifilelib.definition.AviStreamFormat*

Stream format structure for video streams.

For video streams the stream format is a BITMAPINFO structure.

FCC_TYPE = 'vids'

UNPACK_FORMAT = '<I2i2H2I2i2I'

classmethod load(*stream_header*, *file_like*, *force_color_table=False*)
Create a new *BitmapInfoHeaders* instance from a RIFF file.

Parameters

- **stream_header** (*AviStreamHeader*) – Stream header structure for the stream
- **file_like** (*file-like*) – A file-like object positioned at the start of 'strf' chunk.
- **force_color_table** (*bool*) – Force an attempt to load a color table.

Returns The stream format instance for this stream.

Return type *BitmapInfoHeaders*

classmethod load_from_file(*file_like*, *force_color_table=False*)
Create a new *BitmapInfoHeaders* instance from a file.

Parameters

- **file_like** (*file-like*) – A file-like object positioned at the start of 'strf' chunk.
- **force_color_table** (*bool*) – Force an attempt to load a color table.

Returns The stream format instance for this stream.

Return type *BitmapInfoHeaders*

read_colortable(*file_like*, *force=False*)
Read and store a color table.

Parameters

- **chunk** (*file-like*) – The file-like object from which the color table should be read.
- **force** (*bool*) – Try and read a color table even if the stream format indicates that there were zero colors used.

class avifilelib.definition.**UnparsedStreamFormat**(*raw_bytes*)
Bases: *avifilelib.definition.AviStreamFormat*

An holder for a raw stream format structure.

This implementation does not parse the stream format.

Parameters **raw_bytes** (*bytes*) – A byte array with the stream format data.

classmethod load(*stream_header*, *file_like*)
Create an *UnparsedStreamFormat* instance

This method creates a new instance of *UnparsedStreamFormat* from the contents of an AVI 'strf' list.

Parameters

- **stream_header** (*AviStreamHeader*) – Header associated with the stream.
- **file_like** (*file-like*) – A file-like object positioned at the start of a 'strh' list.

Returns Instance of *UnparsedStreamFormat*.

Return type object

1.1.6 avifilelib.enums module

Enumerations associated with AVI Files.

The module provides enumerations used in the fields of AVI file data structures.

class avifilelib.enums.**AVIF**(**args*, ***kwds*)
Bases: aenum.IntFlag

AVI header flags.

COPYRIGHTED = 131072

HASINDEX = 16

ISINTERLEAVED = 256

MUSTUSEINDEX = 32

WASCAPTUREFILE = 65536

class avifilelib.enums.**AVIIF**(**args*, ***kwds*)
Bases: aenum.IntFlag

AVI index flags.

KEYFRAME = 16

LIST = 1

NO_TIME = 256

class avifilelib.enums.**AVISF**(**args*, ***kwds*)
Bases: aenum.IntFlag

AVI stream header Flags.

AVISF_DISABLED = 1

AVISF_VIDEO_PALCHANGES = 65536

class avifilelib.enums.**BI_COMPRESSION**(**args*, ***kwds*)
Bases: aenum.IntFlag

AVI compression flags.

BI_BITFIELDS = 3

BI_CMYK = 11

BI_CMYKREL4 = 13

BI_CMYKRLE8 = 12

BI_JPEG = 4

BI_PNG = 5

BI_RGB = 0

BI_RLE4 = 2

BI_RLE8 = 1

class avifilelib.enums.**FCC_TYPE**
Bases: enum.Enum

AVI stream types.

AUDIO = 'auds'

MIDI = 'mids'

TEXT = 'txts'

VIDEO = 'vids'

class avifilelib.enums.**STREAM_DATA_TYPES**
Bases: enum.Enum

AVI stream chunk data types.

AUDIO_DATA = 'wb'

COMPRESSED_VIDEO = 'dc'

PALETTE_CHANGE = 'pc'

UNCOMPRESSED_VIDEO = 'db'

1.1.7 avifilelib.index module

AVI Index classes.

This module contains classes related to the index structures used in AVI files. At present, the module provides the *AviV1Index* class to represent the AVIOLDINDEX structure, and the *AviV1IndexEntry* class to represent entries in the index.

class avifilelib.index.**AviV1Index**(*index=None*)
Bases: object

A class to represent the *AVIOLDINDEX* structure.

Parameters **index** (*list*) – A list containing *AviV1IndexEntry* objects.

by_data_type(*data_type*)
Get a new index structure containing entries only for *data_type*.

Parameters **data_type** (*avifilelib.enums.AVIIF*) – The type of the data chunks that should be contained in the returned index.

Returns A new index containing entries only for *stream_id*.

Return type *AviV1Index*

by_stream(*stream_id*)
Get a new index structure containing only entries for *stream_id*.

Parameters **stream_id** (*int*) – The index number of stream for which an index should be returned.

Returns A new index containing entries only for *stream_id*.

Return type *AviV1Index*

classmethod load(*file_like*)
Create an *AviV1Index* structure.

This method creates an *AviV1Index* from the contents of an AVI 'idx1' list.

Parameters **file_like** (*file-like*) – A file-like object positioned at the start of a index structure.

Returns An *AviV1Index* that may be used to read the data for this chunk.

Return type *AviV1Index*

class avifilelib.index.**AviV1IndexEntry**(*chunk_id*, *flags*, *offset*, *size*)
Bases: object

A class to represent an *AVIOLDINDEX_ENTRY*.

Parameters

- **chunk_id** (*str*) – String version of the chunk identifier. This consists of two characters for the data type, and two characters for the stream id number.
- **flags** (avifilelib.enum.AVIIF) – Flags associated with a given chunk in the index.
- **offset** (*int*) – Offset in bytes from the start of the 'movi' list to the start of the data chunk.
- **size** (*int*) – Size of the data in the chunk.

classmethod load(*file_like*)
Create an *AviV1IndexEntry* structure.

This method creates an *AviV1IndexEntry* from the contents of an AVI 'idx1' list.

Parameters **file_like** (*file-like*) – A file-like object positioned at the start of a index entry.

Returns An *AviV1IndexEntry* containing data for an index entry.

Return type *AviV1IndexEntry*

1.1.8 avifilelib.riff module

exception avifilelib.riff.**ChunkFormatException**
Bases: Exception

Raised when the underlying chunk data does not match the expected format.

exception avifilelib.riff.**ChunkTypeException**
Bases: Exception

Raised when the underlying chunk data does not match the expected RIFF type.

class avifilelib.riff.**RIFFChunk**(*file*, *align=False*, *bigendian=False*, *inclheader=False*)
Bases: chunk.Chunk

A class for reading RIFF chunks.

A customized version of the chunk.Chunk class to be used for reading RIFF files. The main customization being that the *bigendian* parameter defaults to *False* rather than *True*. Additionally, the object will correctly handle RIFF 'LIST' chunks.

Parameters

- **file** (*file_like*) – A file-like object (has *read()*, *seek()*, and *tell()* methods.
- **align** (*bool*) – Indicates whether the chunk should aligned to a 2-byte boundary.
- **bigendian** (*bool*) – Indicates whether the byte order of the data should be big endian or little endian.
- **inclheader** (*bool*) – Specifies whether the chunk size that will be read includes the size of the chunk header (name and size).

getlisttype()
Type of RIFF list.

Returns The four character type identifier of the RIFF list or *None* if the chunk is not a RIFF list.

Return type str

islist()
Indicates if the chunk contains a RIFF list.

Returns *True* if the chunk contains a RIFF list.

Return type bool

avifilelib.riff.**rollback**(*file_like*, *reraise=False*)
Context manager to recover from failed chunk creation.

This context manager can be used to wrap calls to methods that attempt to read a RIFF chunk but require the chunk to be of a specific type. If the method raises a *ChunkTypeException*, this context manager catches the *ChunkTypeException* and rewinds the *file_like* object to its position before the failed call.

Parameters

- **file_like** (*file-like*) – A file-like object (having at least *tell()* and *seek()* methods).
- **reraise** (*bool*) – If *True*, any *ChunkTypeException* raised while within the context manager will be reraised after *file_like* is rewound.

Yields *file-like* – the object *file_like* passed as a parameter.

1.1.9 avifilelib.rle module

Decoders for Microsoft RLE formats.

This package provides decoders capable of decoding frames encoded using the Microsoft RLE4 and RLE8 formats.

class avifilelib.rle.**RLE4Decoder**(*width*, *height*, *colors*)
Bases: *avifilelib.rle.RLEDecoderBase*

Decoder for RLE4 compression.

This class implements a decoder for the RLE4 compression algorithm.

Parameters

- **width** (*int*) – Width of the image to be decoded.
- **height** (*int*) – Height of the image to be decoded.

- **colors** (*numpy.ndarray, dtype=uint8*) – 16 x 3 of red, green, and blue values.

COMPRESSION = 2

decode_frame_buffer(*buffer*, *size*, *keyframe=True*)
Decode a frame from a bytes object.

Decodes a single frame from the data contained in *buffer*.

Parameters

- **buffer** (*bytes*) – A bytes object containing the frame data.
- **size** (*int*) – Size of the data in the buffer.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

class avifilelib.rle.**RLE8Decoder**(*width*, *height*, *colors*)
Bases: *avifilelib.rle.RLEDecoderBase*

Decoder for RLE8 compression.

This class implements a decoder for the RLE8 compression algorithm.

Parameters

- **width** (*int*) – Width of the image to be decoded.
- **height** (*int*) – Height of the image to be decoded.
- **colors** (*numpy.ndarray, dtype=uint8*) – 256 x 3 of red, green, and blue values.

COMPRESSION = 1

decode_frame_buffer(*buffer*, *size*, *keyframe=True*)
Decode a frame from a bytes object.

Decodes a single frame from the data contained in *buffer*.

Parameters

- **buffer** (*bytes*) – A bytes object containing the frame data.
- **size** (*int*) – Size of the data in the buffer.
- **keyframe** (*bool*) – Indicates to the decoder that this chunk contains a key frame.

Returns A two dimensional array of dimensions *height* by *width* containing the resulting image.

Return type numpy.ndarray

class avifilelib.rle.**RLEDecoderBase**(*width*, *height*, *colors*)
Bases: *avifilelib.decoder.DecoderBase*

Base class for RLE formats.

This class provides the foundation for run-length encoding decoders. Both the RLE4 and RLE8 formats require color paletes, and therefore this class accepts a color palate/table as an argument.

Parameters

- **width** (*int*) – Width of the image to be decoded.
- **height** (*int*) – Height of the image to be decoded.

- **colors** (*numpy.ndarray, dtype=uint8*) – N x 3 of red, green, and blue values, where N is 2^4 or 2^8 depending on the type of RLE compression.

COMPRESSION = (<BI_COMPRESSION.BI_RLE4: 2>, <BI_COMPRESSION.BI_RLE8: 1>)

colors
Get the color table.

classmethod for_avi_stream(*stream_definition*)
Attempts to find a decoder implementation for a stream.

Subclasses of *RLEDecoderBase* are selected by matching the value of *COMPRESSION*.

Returns A subclass of *RLEDecoderBase*.

Return type object

1.2 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index