# rledecode

1.0

Generated by Doxygen 1.8.6

Mon Apr 18 2016 12:10:05

# Contents

# Chapter 1

# Data Structure Index

## 1.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 3

# Data Structure Documentation

## 3.1 RLEFile Struct Reference

Represents a RLEplay file.

`#include <decompress.h>`

### Data Fields

- FILE * **fp**
- int cols

  *pointer into the .rle file*
- int rows

  *width of the image in pixels*
- int pixels

  *height of the image in pixels*

### 3.1.1 Detailed Description

Represents a RLEplay file.

The documentation for this struct was generated from the following file:

- decompress.h

# Chapter 4

# File Documentation

## 4.1 decompress.h File Reference

This file contains prototypes for functions that decompress RLEplay files.

```
#include <stdio.h>
#include <stdint.h>
```

### Data Structures

- struct RLEFile

  *Represents a RLEplay file.*

### Macros

- #define RED 0

  *byte offset for red channel*
- #define GREEN 1

  *byte offset for green channel*
- #define BLUE 2

  *byte offset for blue channel*

### Functions

- void decompress_file (FILE *infile, char *output_basename)

  *Decompresses the file passed in as `infile`.*
- void set_channel_value (uint8_t *buffer, uint8_t colour, int, uint8_t pixcount)

  *Sets the appropriate byte in.*
- void decode_channel (RLEFile *infile, uint8_t colour, uint8_t *)

  *Decodes a single channel into `buffer`.*

#### 4.1.1 Detailed Description

This file contains prototypes for functions that decompress RLEplay files.

**Author**

Robyn A. McNamara

**Date**

April 2016

### 4.1.2 Function Documentation

#### 4.1.2.1 void decode_channel ( RLEFile * *file,* uint8_t *colour,* uint8_t * *buffer* )

Decodes a single channel into `buffer`.

The buffer should exist before this function is called. To fully create the image, `decode_channel()` must be called on the red, green, and blue channels, passing the same buffer in at each call. Once this has been done, `buffer` will contain a stream of 24-bit RGB pixels, suitable for writing to a binary PPM.

**Parameters**

| | |
|---|---|
| *infile* | Reference to the RLEFile struct containing the input stream |
| *colour* | One of `RED`, `GREEN`, or `BLUE` |
| *buffer* | Buffer in which the channel will be set |

decompress.c: functions that decompress RLEplay files. – Robyn A. McNamara, April 2016

#### 4.1.2.2 void decompress_file ( FILE * *infile,* char * *output_basename* )

Decompresses the file passed in as `infile`.

If `output_basename` is non-null, it is interpreted as the desired basename for output PPMs; otherwise, the decompressed data is sent to `stdout` with the integer -1 between frames.

If output is to be stored to PPMs, they will be named <output_basename>-0001.ppm, <output_basename>-0002.-ppm, etc.

**Parameters**

| | |
|---|---|
| *infile* | Pointer to the input file, which has been opened but not validated |
| *output_-basename* | Desired basename for output PPMs, or NULL for output to `stdout` |

#### 4.1.2.3 void set_channel_value ( uint8_t * *buffer,* uint8_t *colour,* int , uint8_t *pixcount* )

Sets the appropriate byte in.

**Parameters**

| | |
|---|---|
| *buffer,given* | the `colour` and `pixcount`. |

Assumes `RED`, `GREEN`, and `BLUE` are appropriately defined.

**Parameters**

| | |
|---|---|
| *buffer* | Array containing pixel values in 8-bit RGB format |
| *colour* | One of `RED`, `GREEN`, or `BLUE` |
| *pixcount* | Index of the pixel whose channel is to be set |

## 4.2 rledecode.c File Reference

Decoder for RLEplay format video files.

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include "decompress.h"
```

### Functions

- void usage ()

  *Prints a usage message to stderr.*
- int **main** (int argc, char ∗argv[])

### 4.2.1 Detailed Description

Decoder for RLEplay format video files. This sample solution is intended as a basis for students to tackle Assignment 2 for FIT3042. It does not implement scaling or tweening as Assignment 2 does not rely on this functionality.

**Author**

Robyn A. McNamara

**Date**

April 2016

# Index