woogac

`woogac` is a multi purpose utility tool. It is designed as a blackbox utility tool that can execute a manyfold of different tasks. It's main purpose for this task is to simulate one or many closed source third party command-line tools.

Usage

`woogac [FLAGS] [OPTIONS] [SUBCOMMAND]`

Flags

name	description
`-h`, `--help`	Prints help information
`-V`, `--version`	Prints version information
`-v`, `--verbose`	output verbosity

Options

name	description	default value	possible values	env
`--color <color>`	Output coloring	`auto`	`auto`, `always`, `never`	`WOOGAC_COLOR`
`--log-dir <log>`	path to log dir	-	-	`WOOGAC_LOG_DIR`

Subcommands

bundle

bundle assets and resources into one bundle archive.

Usage

`woogac bundle [FLAGS] [OPTIONS] <DIR>`

Flags

name	description
`-f`, `--force`	overwrite existing file
`-h`, `--help`	Prints help information
`-V`, `--version`	Prints version information

Options

name	description	default value	possible values	env
`--include <include-pattern>...`	file pattern to include	`*`	-	-
`--exclude <exclude-pattern>...`	file pattern to exclude	``	-	-
`-o`, `--output-dir <Dir>`	Output directory to use.	-	-	`PWD`
`--file-name <file_name>`	The file name of for bundle	`bundle.zip`	-	`WOOGAC_BUNDLE_FILENAME`

ARGS

name	description
`<DIR>`	directory with assets to bundle

Examples

Here are a few examples to show some basic use cases

create bundle from provided directory

`woogac bundle dir`

This will create a bundle with all files located in the provided directory. Subfolders will also be bundled. The bundle will be saved in the current working directory with the name `bundle.zip`

set file name for bundle

`woogac bundle --file-name custom_bundle.zip dir`

export bundle to a different directory

`woogac bundle --output-dir export/path dir`

This exports the bundle to the provided directory `export/path`. The directory must exist before executing.

set include filter for bundling

Giving the following directory structure

```
dir/
├── airplane_cast.png
├── airplane_cover.png
├── airplane_item.png
├── angel_cast.png
├── angel_cover.png
├── angel_item.png
├── awning_cast.png
├── awning_cover.png
├── awning_item.png
├── background.png
├── bag_cast.png
├── bag_cover.png
├── bag_item.png
```

`woogac bundle --include '*_item*' dir`

Will bundle only files matching the include glob pattern.

Multiple `--include` filter can be provided.

`woogac bundle --include '*_item*' --include 'bag_cover*' dir`

set exclude filter for bundling

Giving the following directory structure

```
dir/
├── airplane_cast.png
├── airplane_cover.png
├── airplane_item.png
├── angel_cast.png
├── angel_cover.png
├── angel_item.png
├── awning_cast.png
├── awning_cover.png
├── awning_item.png
├── background.png
├── bag_cast.png
├── bag_cover.png
├── bag_item.png
```

`woogac bundle --exclude '*_cast*' dir`

Will bundle all files except files which match the provided glob pattern.

set exclude/include filter for bundling for directories

The exclude and include glob filter work not only on filenames

```
dir/
├── 001-01
│   └── scene
│       ├── airplane_cast.png
│       ├── airplane_cover.png
│       ├── airplane_item.png
│       ├── angel_cast.png
│       ├── angel_cover.png
│       ├── angel_item.png
│       ├── awning_cast.png
│       ├── awning_cover.png
│       ├── awning_item.png
│       ├── background.png
│       ├── bag_cast.png
│       ├── bag_cover.png
│       └── bag_item.png
└── 001-02
    └── scene
        ├── angel_cast.png
        ├── angel_item.png
        ├── background.png
        ├── bag_cast.png
        ├── bag_cover.png
        ├── bag_item.png
        ├── ball_cast.png
        ├── ball_cover.png
        ├── ball_item.png
        ├── beads_cast.png
        └── beads_item.png
```

`woogac bundle --include '*001-02*' dir`

will only include files which contain `001-02` in their file-path.

checksum

calculates sha256 checksums for provided file[s].

Usage

```
woogac checksum [OPTIONS] [Files]...
```

Flags

name	description
`-h`, `--help`	Prints help information
`-V`, `--version`	Prints version information

Options

name	description	default value	possible values	env
`--file-name <file_name>`	The file name for the report. The default name depends on the selected `--output-format`.	`checksums.txt` / `checksums.json`	-	`WOOGAC_CHECKSUM_FILENAME`
`-o`, `--output-dir <Dir>`	Output directory to use. This argument must point to a valid writable directory The report will be written to stdout, if this option is not provided.	-	-	-
`--output-format <output-format>`	The output format. The format is either plain text file. Each line contains the file path and the checksum seperated by a space character. Or a json hashmap"	`plain`	`json`, `plain`	`WOOGAC_CHECKSUM_OUTPUT_FOR`

ARGS

name	description
`<FILES>...`	input file[s] generate checksum for

Examples

Here are a few examples to show some basic use cases

output sha256 checksum info for a single file

`woogac checksum file.txt`

This will print the path and sha256 checksum of the provided file

```
/path/to/file.txt 5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144
```

output sha256 checksum for a multiple files

`woogac checksum file1.txt file2.txt`

This will print the path and sha256 checksum of the provided files. One file per line

```
/path/to/file1.txt 5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144
/path/to/file2.txt 884b32c07ee64f1acbd0e6bd6d63d0e8784146325f08464fcd29fe2d3adf04f1
```

output sha256 checksum info for a single file in json format

`woogac checksum --output-format json file.txt`

This will print the path and sha256 checksum of the provided image as a json hash

```
{
  "/path/to/file.txt": "5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144"
}
```

output sha256 checksum info for multiple files in json format

`woogac checksum --output-format json file1.txt file2.txt`

This will print the path and sha256 checksum of the provided files as a json hash

```
{
  "/path/to/file1.png": "5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144",
  "/path/to/file2.png": "884b32c07ee64f1acbd0e6bd6d63d0e8784146325f08464fcd29fe2d3adf04f1"
```

```
  }
```

redirect output to directory

`woogac checksum --output-dir . file.txt`

Saves the output in a file. By default the filename will be either `info.txt` or `info.json` depending on the selected `--output-format`.

redirect output from multiple files to directory with custom filename

`woogac checksum --output-dir . --file-name custom-info.txt file1.txt file2.txt`

crop

Crop image[s] from center with provided width and height.

Usage

`woogac crop [FLAGS] [OPTIONS] --height <height> --width <width> [Files]...`

Flags

name	description
`-f`, `--force`	Prints help information
`-h`, `--help`	Prints help information
`-V`, `--version`	Prints version information

Options

name	description	default value	possible values	env
`-o`, `--output-dir <Dir>`	Output directory to use. This argument must point to a valid writable directory.	-	-	`PWD`
`--format <format>`	output file format	`png`	`png`, `jpg`	`WOOGAC_CROP_FORMAT`
`-h`, `--height <height>`	Target height to crop image to. This value sets the height of the crop rectangle. If the value is bigger than the image height, the image height will be	-	-	-

	used instead.			
`-w`, `--width <width>`	Target width to crop image to. This value sets the width of the crop rectangle. If the value is bigger than the image width, the image width will be used instead.	-	-	-
`-p`, `--name-pattern <name_pattern>`	A name pattern to use for the output file. Each provided image file will be renamed according to this pattern. The command uses two variables which can be used to control the final file name: `{name}` *: the original file name without file extension* `{ext}` : the new file extension based on the provided `--format`	`{name}.{ext}`	-	`WOOGAC_CROP_NAME_PATTERN`
`--output-format <output-`	The output format. The format is either plain text file. Each line contains the `file-path` `width` and `height` and	`plain`	`json`, `plain`	`WOOGAC_CROP_OUTPUT_FORMAT`

`format>`	the `checksum` separated by a space character. Or a json hashmap"			

ARGS

name	description
`<FILES>...`	input file[s] to crop

Examples

Here are a few examples to show some basic use cases

crop image to 40px by 80px with the default format

```
woogac crop --width 40 --height 80 image.png
```

crop multiple images to 40px by 80px with the to jpg format

```
woogac crop --width 40 --height 80 --format jpg image1.png image2.png
```

save images in different directory after resize

```
woogac crop --width 20 --height 20 --output-dir export image1.png image2.png
```

rename images with pattern

```
woogac crop --width 20 --height 20 --name-patter '{name}_sm.{ext}' image1.png
image2.png
```

change report output format to json

```
woogac crop --width 20 --output-format json image1.png
```

force override of existing files

```
woogac pack --force image1.png image2.png
```

image-info

reads image informations for provided file[s].

Usage

`woogac image-info [OPTIONS] [Files]...`

Flags

name	description
`-h` , `--help`	Prints help information
`-V` , `--version`	Prints version information

Options

name	description	default value	possible values	env
`--file-name <file_name>`	The file name for the report. The default name depends on the selected `--output-format` .	`info.txt` / `info.json`	-	`WOOGAC_IMAGE_INFO_FILENAME`
`-o` , `--output-dir <Dir>`	Output directory to use. This argument must point to a valid writable directory The report will be written to stdout, if	-	-	-

	this option is not provided.			
`--output-format <output-format>`	The output format. The format is either plain text file. Each line contains the `file-path` `width` and `height` and the `checksum` separated by a space character. Or a json hashmap	`plain`	`json`, `plain`	`WOOGAC_IMAGE_INFO_OUTPUT_FORMAT`

ARGS

name	description
`<FILES>...`	Input file[s] to fetch information for. One or more file paths to valid image files. Files must be files and not directories.

Examples

Here are a few examples to show some basic use cases

output image info for a single image

`woogac image-info image.png`

This will print the path, width and height and sha256 checksum of the provided image

```
/path/to/image.png 62 33 5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144
```

output image info for multiple images

`woogac image-info image1.png image2.png`

This will print the path, width and height and sha256 checksum of the provided images. One image per line

```
/path/to/image1.png 62 33 5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144
/path/to/image2.png 40 60 884b32c07ee64f1acbd0e6bd6d63d0e8784146325f08464fcd29fe2d3adf04f1
```

output image info for a single image in json format

`woogac image-info --output-format json image.png`

This will print the path, width and height and sha256 checksum of the provided image as a json hash

```
{
  "/path/to/image.png": {
    "size": {
      "width": 62,
      "height": 33
    },
    "checksum": "5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144"
  }
}
```

output image info for multiple images in json format

`woogac image-info --output-format json image1.png image2.png`

This will print the path, width and height and sha256 checksum of the provided images as a json hash

```
{
  "/path/to/image1.png": {
    "size": {
      "width": 62,
      "height": 33
    },
    "checksum": "5488521a9dfe1bad31164609d018fcd645a263c9e3108fd79823f3389628a144"
  },
  "/path/to/image2.png": {
    "size": {
      "width": 40,
      "height": 60
    },
    "checksum": "884b32c07ee64f1acbd0e6bd6d63d0e8784146325f08464fcd29fe2d3adf04f1"
  }
}
```

redirect output to directory

`woogac image-info --output-dir . image.png`

Saves the output in a file. By default the filename will be either `info.txt` or `info.json` depending on the selected `--output-format`.

redirect output from multiple images to directory with custom filename

```
woogac image-info --output-dir . --file-name custom-info.txt image1.png image2.png
```

pack

Pack images into one big texture Atlas.
The pack algorithm will try to fit all
provided images into the defined rectangle. The command fails of not all images fit.

Usage

`woogac pack [FLAGS] [OPTIONS] [Files]...`

Flags

name	description
`-f` , `--force`	Prints help information
`-h` , `--help`	Prints help information
`-V` , `--version`	Prints version information

Options

name	description	default value	possible values	env
`-o`, `--output-dir <Dir>`	Output directory to use. This argument must point to a valid writable directory.	-	-	`PWD`
`--format <format>`	output file format	`png`	`png`, `jpg`	`WOOGAC_RESIZE_FORMAT`
`-h`, `--height <height>`	Height of the texture atlas. This value determines the final height of the texture atlas.	`2048`	-	-
`-w`, `--width <width>`	Width of the texture atlas. This value determines the final width of the texture atlas.	`2048`	-	-
`--file-name <file_name>`	The file name of for the texture atlas without extension	`texture_atlas`	-	`WOOGAC_PACK_FILENAME`

ARGS

name	description
`<FILES>...`	Input file[s] to pack. One or more file paths to valid image files. Files must be files and not directories.

Examples

Here are a few examples to show some basic use cases

pack provided images into the texture atlas

```
woogac pack image1.png image2.png image3.png image4.png image5.png
```

This will create two files in the working directory. A texture atlas named `texture_atlas.png` and the atlas manifest in json format `texture_atlas.json`

pack provided images into the texture atlas as with jpg format

`woogac pack --format jpg image1.png image2.png image3.png image4.png image5.png`

This will create two files in the working directory. A texture atlas named `texture_atlas.jpg`

export atlas to a different directory

`woogac pack --output-dir export/path image1.png image2.png image3.png image4.png image5.png`

This exports both texture atlas and manifest to the provided directory `export/path`. The directory must exist before executing.

set filename for exported atlas

`woogac pack --file-name atlas1 image1.png image2.png image3.png image4.png image5.png`

Both texture atlas and manifest will be named according to the provided name
`atlas1.png` / `atlas1.json`

set custom atlas width and height

`woogac pack --width 4096 --height 512 image1.png image2.png image3.png image4.png image5.png`

resize

Resize a set of image files.
This command allows to batch resize images.

Usage

`woogac resize [FLAGS] [OPTIONS] <Files>...`

Flags

name	description
`--exact`	Resize without preserving aspect ratio
`-f`, `--force`	Prints help information
`-h`, `--help`	Prints help information
`-V`, `--version`	Prints version information
`--use-percent`	Interpret width and height argument as percentage values. When this flag is set, Both `--with` and `--height` arguments will be interpreted as percent values. This allows to increase a series of images by the same factor. If `--use-percent` is set `--exact` is not set, then only one argument for `--width` or `--height` is needed.

Options

name	description	default value	possible values	env
`-o`, `--output-dir <Dir>`	Output directory to use. This argument must point to a valid writable directory.	-	-	`PWD`
`--format`	output file		`png`,	

`<format>`	format	`png`	`jpg`	`WOOGAC_RESIZE_FORMAT`
`-h`, `--height <height>`	Target height to resize image to. By default the resize is preserving the aspect ratio of the provided image. If the `--exact` flag is also set, then the command will resize the image exactly to the provided height. If this argument is not provided, the current image height will be used. If the `--percent` flag is set, the value will be interpreted as a percentage value. If this argument is not provided but the `--percent` flag is set the, the value 100 will be used.	-	-	-

`-w`, `--width <width>`	Target width to resize image to. By default the resize is preserving the aspect ratio of the provided image. If the `--exact` flag is also set, then the command will resize the image exactly to the provided width. If this argument is not provided, the current image width will be used. If the `--percent` flag is set, the value will be interpreted as a percentage value. If this argument is not provided but the `--percent` flag is set the, the value 100 will be used.	-	-	-
	A name pattern to use for the output file. Each			

`-p`, `--name-pattern <name_pattern>`	provided image file will be renamed according to this pattern. The command uses two variables which can be used to control the final file name: *`{name}`: the original file name without file extension* `{ext}`: the new file extension based on the provided `--format`	`{name}.{ext}`	-	`WOOGAC_RESIZE_NAME_PATTERN`
`--output-format <output-format>`	The output format. The format is either plain text file. Each line contains the `file-path` `width` and `height` and the `checksum` separated by a space character. Or a json hashmap	`plain`	`json`, `plain`	`WOOGAC_RESIZE_OUTPUT_FORMAT`

ARGS

name	description
`<FILES>...`	Input file[s] to resize. One or more file paths to valid image files. Files must be files and not directories.

Examples

Here are a few examples to show some basic use cases

resize image proportional to 40px by 80px with the default format

```
woogac resize --width 40 --height 80 image.png
```

resize multiple images proportional to 40px by 80px with the to jpg format

```
woogac resize --width 40 --height 80 --format jpg image1.png image2.png
```

save images in different directory after resize

```
woogac resize --width 20 --height 20 --output-dir export image1.png image2.png
```

rename images with pattern

```
woogac resize --width 20 --height 20 --name-patter '{name}_sm.{ext}' image1.png
image2.png
```

change report output format to json

```
woogac resize --width 20 --output-format json image1.png
```

resize image to exact width and height values

```
woogac resize --exact --width 20 --height 20 image1.png
```

resize with percentage values

```
woogac resize --use-percent --width 20 --height 20 image1.png
```

force override of existing files

```
woogac resize --force --width 20 --height 20 image1.png
```