Line Track Designer

Quentin Deschamps

Jan 30, 2024

CONTENTS:

CHAPTER

ONE

PRESENTATION

Line Track Designer is a tool to easily design line following tracks for robots. Tracks can be created in two different ways:

- with the command line interface (CLI)
- with the application programming interface in Python (API)

1.1 Introduction

This part explains how to install *Line Track Designer* and shows quickly how does the library work.

1.1.1 Installation

Using pip

You can install latest release of the *Line Track Designer* library using `pip3`:

```
pip3 install line-track-designer
```

Using setuptools

The second way to install the library is to clone the GitHub repository and to use *setuptools*:

```
git clone https://github.com/Quentin18/Line-Track-Designer.git
python3 setup.py install
```

Installing on Windows

On Windows, you can install the library with the two methods above. But, you probably will see the warning bellow:

```
The script linetrack.exe is installed in 'C:\Users\...' which is not on PATH.
```

To fix this warning, add the path indicated in the message to the PATH. You can follow the tutorial here.

Markdown editor

If you do not have any markdown editor, you can download Zettlr. It is very usefull to see markdown files built with *Line Track Designer*. In addition, to convert markdown files into PDF files, you need to install:

- Pandoc
- MiKTeX

Running tests

If you cloned the repository, the tests can be run using:

```
python3 setup.py pytest
```

1.1.2 Quickstart

Main idea

The main idea of *Line Track Designer* is to build tracks easily that you can **edit**, **save**, **share** and **print** with a printer.

To do that, different **tiles** are used and can be associated like a puzzle. This tiles originate from a PDF file and can be printed in A4 format (or US letter paper). They are squares of 200 mm, and are represented by a number between 2 and 33 which corresponds to the page number in the PDF file.

You can see all the tiles here: `linefollowtiles.pdf`

A track is represented by two matrix T and O:

- T contains the number of each tile of the track
- O indicates the orientation of each tile

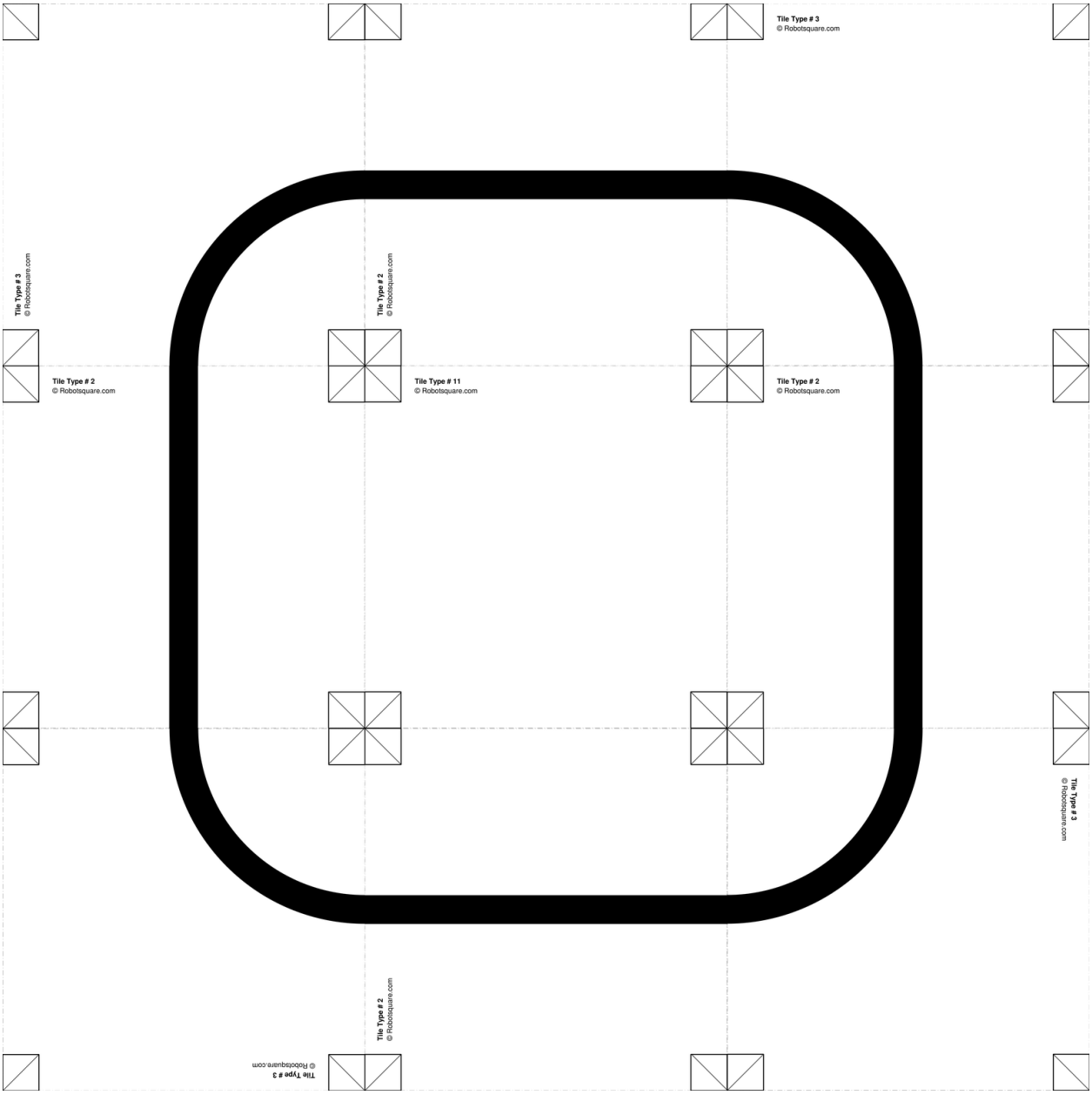
A correct orientation is 0, 1, 2 or 3 which corresponds to the number of times the tile is rotated by 90 degrees.

For example, we consider the following matrix:

$$T = \begin{pmatrix} 3 & 2 & 3 \\ 2 & 11 & 2 \\ 3 & 2 & 3 \end{pmatrix}$$

$$O = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 2 & 1 & 3 \end{pmatrix}$$

The track associated to this two matrix is:



Tracks are stocked in **text files**. The two matrix are "superposed" like this:

```
3;1 2;1 3;0
2;0 11;0 2;0
3;2 2;1 3;3
```

Create your first track

This is how to build the track above with *Line Track Designer*:

1. Create en empty track `track.txt` with 3 rows and 3 columns using this command:

```
linetrack create track.txt 3 3
```

You will arrive in your default text editor with this content:

```
0;0 0;0 0;0
0;0 0;0 0;0
0;0 0;0 0;0
```

2. Change the numbers so that it looks like this:

```
3;1 2;1 3;0
2;0 11;0 2;0
3;2 2;1 3;3
```

3. Save the modifications and quit the text editor.
4. You can show the track as image using:

```
linetrack show track.txt
```

It will display the track in your picture reader.

5. Generate the markdown file associated to the track with:

```
linetrack savemd track.txt
```

The program asks the name of the track and its description. The files `track.png` and `track.md` will be created. You can convert the markdown file into PDF or HTML. This is the result: `track.pdf`

You can also create exactly the same files using a Python script with the API of *Line Track Designer*:

```
import numpy as np
from line_track_designer.track import Track


# Arrays for the track
tiles = np.array([
    [3, 2, 3],
    [2, 11, 2],
    [3, 2, 3]
])
orient = np.array([
    [1, 1, 0],
    [0, 0, 0],
    [2, 1, 3]
])

# Creation of the track
track = Track(tiles, orient, 'Test track')
```

(continues on next page)

(continued from previous page)

```
# Save the track
track.save_txt('track.txt')
# Make png file
track.save_img('track.png')
# Make markdown file
track.save_md('track.md', 'Easy track')
```

1.2 CLI Reference

The command line interface of *Line Track Designer* is available for Linux, macOS and Windows. It uses *Click*. To check if you successfully installed the library, you can entry in a command prompt:

```
linetrack --help
```

You will see the following content:

```
Usage: linetrack [OPTIONS] COMMAND [ARGS]...

Generate line following tracks for robots.

Options:
-v, --verbosity  Set the verbosity
--help           Show this message and exit.

Commands:
addcol    Add a column to track FILENAME.
addrow    Add a row to track FILENAME.
create    Create empty track FILENAME.
delcol    Delete column COL from track FILENAME.
delrow    Delete row ROW from track FILENAME.
doc       Open the documentation.
edit      Edit track FILENAME.
pdf       Open the PDF file containing the tiles.
printing  Print track FILENAME.
rotate    Rotate track FILENAME.
savemd    Save track FILENAME as MD file.
savepng   Save track FILENAME as PNG file.
show      Show track FILENAME as PNG file.
showtile  Show tile NUMBER.
write     Write track FILENAME in the command prompt.
```

It is the help menu of the CLI. You can see all the commands you can use.

You can open the documentation using this command:

```
linetrack doc [OPTIONS]
```

1.2.1 Creating a track

To **create a track**, you can use the `create` command:

```
linetrack create [OPTIONS] FILENAME NROW NCOL
```

FILENAME must be a text file. You need to indicate the number of rows and columns of the track. It creates a track with only blank tiles and open it so that you can edit it.

For example:

```
linetrack create track.txt 3 4
```

This command creates the file `track.txt` and open it with this content:

```
0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0
```

1.2.2 Editing a track

The `edit` command is usefull to **open a track with your default text editor** and modify it.

```
linetrack edit [OPTIONS] FILENAME
```

The file must be a text file corresponding to a track.

Moreover, you can use different commands to modify the track:

- `addcol` and `addrow`: **add a column/row** to a track

```
linetrack addcol [OPTIONS] FILENAME
linetrack addrow [OPTIONS] FILENAME
```

- `delcol` and `delrow`: **delete a column/row** from a track

```
linetrack delcol [OPTIONS] FILENAME COL
linetrack delrow [OPTIONS] FILENAME ROW
```

COL/ROW is the number of the column/row to delete.

- `rotate`: **rotate** a track

```
linetrack rotate [OPTIONS] FILENAME
```

The number of rotations can be indicated using the `-n` option.

1.2.3 Showing a track

You can display a track in two different ways:

- **writing it in the command prompt** using the `write` command

```
linetrack write [OPTIONS] FILENAME
```

- **showing it in your picture viewer** using the `show` command

```
linetrack show [OPTIONS] FILENAME
```

For example, we consider the `track.txt` file with this content:

```
3;1 2;1 3;0
2;0 11;0 2;0
3;2 2;1 3;3
```

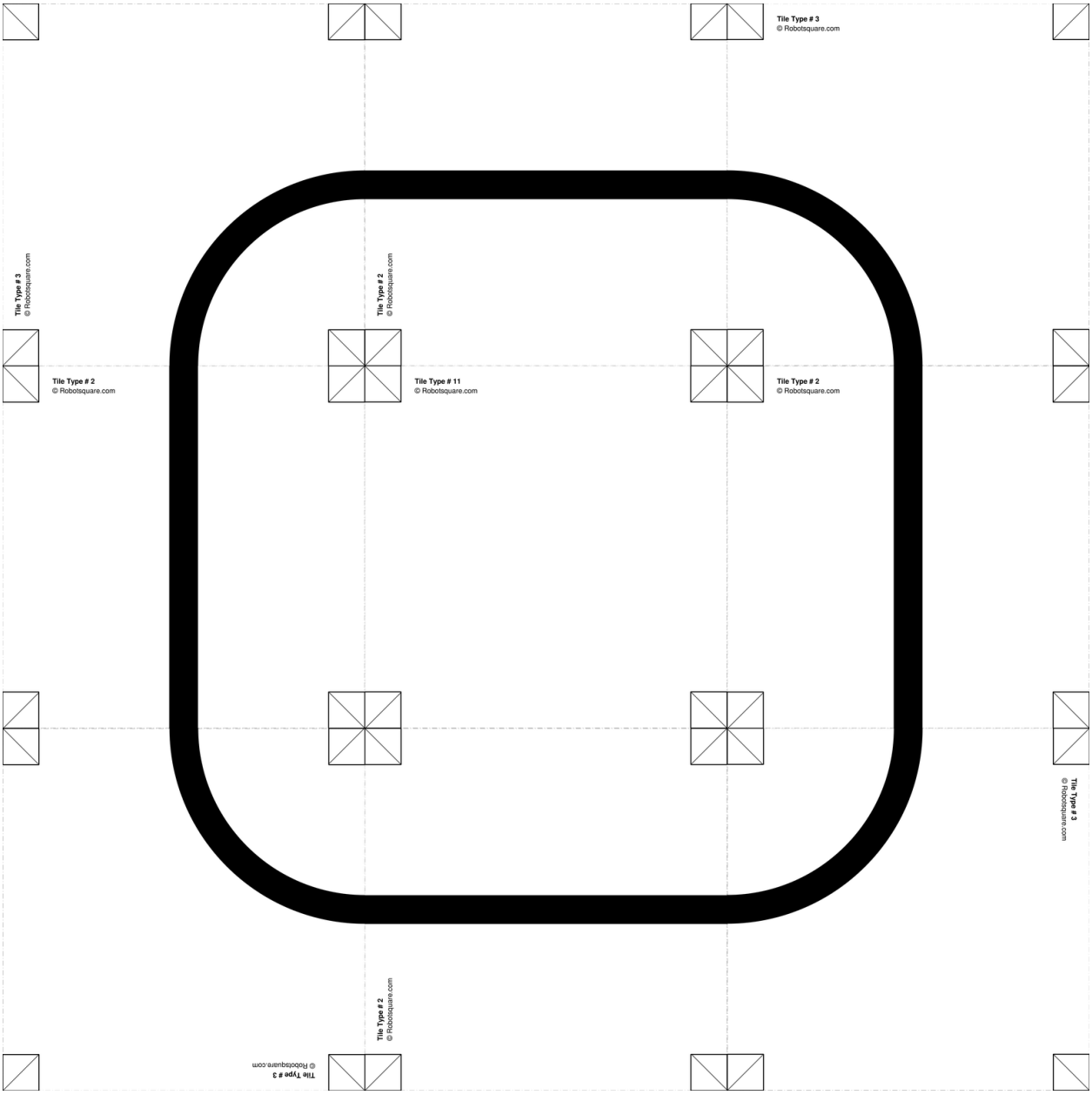
The first command will display its content in the command prompt:

```
linetrack write track.txt
```

With the second command,

```
linetrack show track.txt
```

We can see this PNG image:



1.2.4 Exporting a track

Once your track is finished, you can export it to easily share it.

First, you can generate and **save the PNG file associated to your track** using the `savepng` command:

```
linetrack savepng [OPTIONS] FILENAME
```

You can specify the name of the output PNG file using the `-o` or `--output` option. You can also open the PNG file using the `-s` or `--show` command.

For example:

```
linetrack savepng -o track_image.png track.txt
```

Then, you can **create a markdown file** to generate a little documentation about your track. To do that, you can use the `savemd` command:

```
linetrack savemd [OPTIONS] FILENAME
```

The following options are available:

```
-o, --output TEXT       Name of the MD file
-n, --name TEXT         Name of the track
-d, --description TEXT  Description of the track
```

The markdown file generated can be exported into PDF and HTML. You can see an example of a PDF file generated by this command here: `track.pdf`

Note: The `savemd` command will also generate the PNG file in the same folder than the markdown file.

1.2.5 Printing a track

Warning: This command can be used only on Linux and macOS.

Once your track is finished, you can **print** it in A4 format (or US letter paper) with your printer using the `printing` command:

```
linetrack printing [OPTIONS] FILENAME
```

1.2.6 Showing the tiles

You can see the tiles available using two commands:

- `showtile`: **show a tile** corresponding to the number given

```
linetrack showtile [OPTIONS] NUMBER
```

This command will open the PNG file corresponding to the tile in your picture viewer. You can indicate the orientation using the `-o` or `--orient` option.

- `pdf`: **open the PDF file containing the tiles** in your web browser

```
linetrack pdf [OPTIONS]
```

With this command, you can see the PDF file used by *Line Track Designer* and can choose the tiles you want to use in your track.

Note: On Windows, you can also use this command to open the PDF file and print the tiles you want manually.

1.3 API Reference

You can use the API of *Line Track Designer* to create tracks with Python scripts. This part explains all the classes and methods you can use.

The main module is **Track**.

To import *Line Track Designer*:

```
import line_track_designer
```

1.3.1 Track

With the **track** module, you can create, import, edit, save, and export tracks.

class track.**Track**(*tiles*, *orient*, *name='track'*)

Bases: object

Representation of a track. An instance of the **Track** class is composed of three fields:

- **tiles**: array which contains the number of each tile of the track
- **orient**: array which indicates the orientation of each tile
- **name**: name of the track

static read(*file*, *name='track'*)

Read a text file representing a track and return the track associated.

Parameters

- **file** (*str*) – filename
- **name** (*str*) – name of the track

Returns
the track associated to the file

Return type
Track

Raises

- ***LineTrackDesignerError*** – file not found
- ***LineTrackDesignerError*** – bad filename extension: requires .txt

static zeros(*nrow*, *ncol*, *name='track'*)

Create an empty track.

Parameters

- **nrow** (*int*) – number of rows
- **ncol** (*int*) – number of columns
- **name** (*str*) – name of the track

Returns
empty track (only zeros)

Return type
Track

static max_shape(*width*, *height*)

Return the maximum number of rows and columns of a track limited by a width and a height in mm.

Parameters

- **width** (*int*) – width in mm
- **height** (*int*) – height in mm

Returns
number of rows and columns

Return type
tuple of int

__init__(*tiles*, *orient*, *name='track'*)

Init a track. The arguments tiles and orient must be numpy arrays. For example:

```
import numpy as np
from line_track_designer.track import Track

tiles = np.array([
    [3, 2, 3],
    [2, 11, 2],
    [3, 2, 3]
])
orient = np.array([
    [1, 1, 0],
    [0, 0, 0],
    [2, 1, 3]
])

track = Track(tiles, orient, 'my track')
```

Parameters

- **tiles** (*numpy.array*) – array of tiles
- **orient** (*numpy.array*) – array of orientations
- **name** (*str*) – name of the track

Raises

- ***LineTrackDesignerError*** – tiles and orient must have the same shape
- ***LineTrackDesignerError*** – invalid values

property tiles

Get the array of tiles.

property orient

Get the array of orientations.

property name

Get the name of the track.

__str__()

Make the string format of the track. The tiles and orient matrix are superposed in one matrix. Each couple of values is separated by a semicolon.

With the last example, we obtain:

```
3;1 2;1 3;0
2;0 11;0 2;0
3;2 2;1 3;3
```

__repr__()

Make the repr format of the track. It's the same than the string format.

add_col()

Add a column to the track. This column is filled with 0.

add_row()

Add a row to the track. This row is filled with 0.

del_col(*col*)

Delete a column from the track.

Parameters
col (*int*) – index of the column to delete

del_row(*row*)

Delete a row from the track.

Parameters
row (*int*) – index of the row to delete

set_tile(*row*, *col*, *tile*, *orient*)

Set a tile of the track.

Parameters

- **row** (*int*) – index of the row of the tile
- **col** (*int*) – index of the column of the tile
- **tile** (*int*) – number of the tile
- **orient** (*int*) – orientation of the tile

Raises
LineTrackDesignerError – invalid tile/orient value

rotate(*k=1*)

Rotate the track. The argument k is the number of times the array is rotated by 90 degrees.

Parameters
k (*int*) – number of rotations (default: 1)

dimensions()

Return the dimensions in mm of the track.

Returns
width and height in mm

Return type
tuple of int

occurences()

Return the occurences of each tile used by the track. It returns a dictionary. The keys corresponds to the number of a tile and the values are the number of occurences.

Returns
occurences

Return type
dict

print_track()

Ask the printer to print the tiles to build the track.

export_img()

Export the track to image. It uses the PIL library.

Returns
image of the track

Return type
Image

show()

Displays the track with the PIL library. The image is in PNG format.

save_img(*file*)

Save the track as an image.

Parameters
file (*str*) – filename

Raises
LineTrackDesignerError – bad filename extension: use .png

save_txt(*file*)

Save the track as a text file. The content of the text file corresponds to the string format of the track.

Parameters
file (*str*) – filename

Raises
LineTrackDesignerError – bad filename extension: use .txt

save_md(*file*, *description=''*)

Save the track as a markdown file. It also creates the PNG image associated to the track. The md file contains the following informtions:

- name of the track
- PNG image of the track
- description of the track (optionnal)
- dimensions (in mm)
- tiles required to build the track

Parameters

- **file** (*str*) – filename (markdown file)
- **description** (*str*) – description of the track

Raises
LineTrackDesignerError – bad extension file: use .md

1.3.2 Tile

The **tile** module manages the tiles that are used to build a track.

Note: You can see all the tiles here: `linefollowtiles.pdf`

Warning: The tiles 10 and 32 can not be used by *Line Track Designer*.

`class tile.Tile(number)`

Bases: `object`

Representation of a tile. An instance of the **Tile** class is composed of four fields:

- **number**: number of the tile
- **name**: name of the tile
- **path**: path to the PNG image corresponding to the tile
- **image**: PIL.Image format associated to the tile

`SIDE = 200`

`static is_valid(number)`

Return True if the number corresponds to a valid tile. It is valid if the number is between 2 and 33, and the tiles 10 and 32 are invalid.

Parameters
number (*int*) – number of a tile

Returns
Is a valid number

Return type
bool

`__init__(number)`

Init a tile.

Parameters
number (*int*) – number of the tile

Raises
LineTrackDesignerError – invalid tile number

`property number`

Get the number of the tile.

`property name`

Get the name of the tile.

`property path`

Get the path of the PNG image associated to the tile.

property image

Get the image associated to the tile.

__str__()

Make the sting format of the tile. It returns its name.

__repr__()

Make the repr format of the tile. It's the same than the string format.

show(*orient=0*)

Show the tile in your picture viewer.

Parameters
orient (*int*) – orientation of the tile (default: 0)

Raises
LineTrackDesignerError – invalid orient value

class tile.**Tiles**

Bases: object

Manage all the tiles. The tiles are stocked in the dictionary **dict_tiles**. The keys correspond to the number of the tile and the values are the Tile objects corresponding to this number.

__init__()

Init the tiles. It creates the dictionary **dict_tiles**.

property dict_tiles

Get the dictionary of tiles.

__str__()

Make the sting format of the tiles. It returns the names of the tiles.

__repr__()

Make the repr format of the tiles. It's the same than the string format.

get_tile(*number*)

Get a tile from its number.

Parameters
number (*int*) – number of the tile

Returns
tile associated to the number

Return type
Tile

Raises
LineTrackDesignerError – tile not found

static show()

Open the PDF file containing the tiles.

Raises
LineTrackDesignerError – unable to open the PDF file

1.3.3 Printer

With the **printer** module, you can print the tracks built with the library. It uses *CUPS*.

> **Warning:** This module can be used only on Linux and macOS.

class printer.**Printer**

Bases: object

Manage the printer to print a track. It is composed of three fields:

- **conn**: connection to the *CUPS* server
- **printer_name**: the name of the default printer
- **file_tiles**: the path to the PDF document with the tiles to print

Raises
LineTrackDesignerError – no printers found

> **Note:** If no printer is found, you need to add one in your devices.

__init__()

Init a Printer object.

property conn

Get the connection.

property printer_name

Get the name of the printer.

property file_tiles

Get the path of the PDF file.

__str__()

Make the string format of the Printer object. It returns the name of the printer.

__repr__()

Make the repr format of the Printer object. It's the same than the string format.

print_page(*copies*, *pages*, *title*, *media='a4'*)

Ask to the printer to print pages of the PDF file.

Parameters

- **copies** (*int*) – number of copies to print
- **pages** (*int*) – pages to print
- **title** (*str*) – name of the printing
- **media** (*str*) – format (default: 'a4')

Raises
LineTrackDesignerError – printing failed

1.3.4 Markdown

The **markdown** module is usefull to export tracks to markdown files. The **Markdown** class creates a markdown file and can add elements such as titles, images, and tables.

A markdown file can be edited using the *with* statement.

Note: You can read markdown files using *Zettlr*.

---

class markdown.**Markdown**(*filename*)

Bases: object

Create a markdown file. A Markdown object is composed of two fields:

- **filename** (str)
- **f** (file object)

__init__(*filename*)

Init a markdown file. It creates and opens the file.

Parameters
filename (*str*) – filename (markdown file)

property filename

Get the filename.

property f

Get the file object.

__enter__()

Enter in a with statement.

write(*text*, *break_before=True*, *break_after=True*)

Write text in the file. You can precise if you want a break line before and/or after the text.

Parameters

- **text** (*str*) – text to add
- **breack_before** (*bool*) – add a break line before the text if True
- **breack_after** (*bool*) – add a break line after the text if True

close()

Close the file.

add_title(*title*, *level*)

Add a title to the file. You can choose the level of the title.

Parameters

- **title** (*str*) – title to add
- **level** (*int*) – level of the title (must be between 1 and 6)

Raises
Exception – invalid level for a title in md

add_image(*file*, *name='Track'*)

Add an image to the file.

Parameters

- **file** (*str*) – filename of the image to add
- **name** (*str*) – label of the image

add_separator()

Add a separator to the file.

add_table(*array*, *head=True*)

Add a table to the file. You can precise if you want a header or not.

Parameters

- **array** (*numpy.array*) – table to add
- **head** (*bool*) – make a header if True

__exit__(*exc_type*, *exc_value*, *traceback*)

Exit and close the file.

1.3.5 Errors

The **error** module manages the errors of the library.

exception error.**LineTrackDesignerError**

Bases: Exception

Manage exception errors.

1.4 Examples

In this part, you find 4 tracks built with *Line Track Designer*. This pages have been created by the `savemd` command.

Moreover, you can find some examples that show how to use the API of *Line Track Designer* in the `examples` folder here.

1.4.1 Test track

Description

Easy track

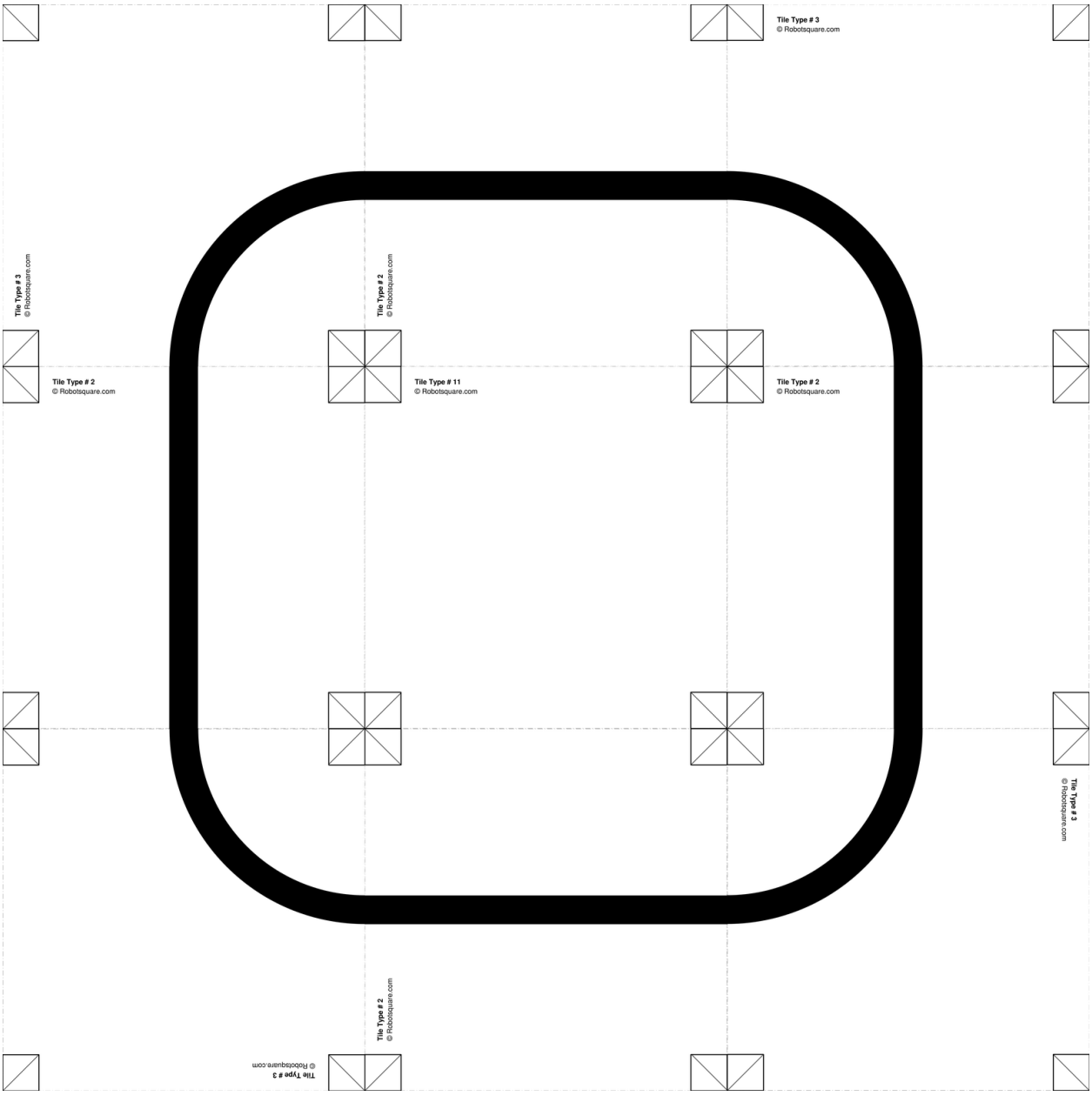


Fig. 1: Test track

Dimensions

Width	Height
600 mm	600 mm

Tiles

Tile number	Number of copies required
2	4
3	4
11	1

Built with Line Track Designer

1.4.2 Colors track

Description

A track with colors

Dimensions

Width	Height
1000 mm	1000 mm

Tiles

Tile number	Number of copies required
2	4
21	1
25	1
26	1
27	1
28	1

Built with Line Track Designer

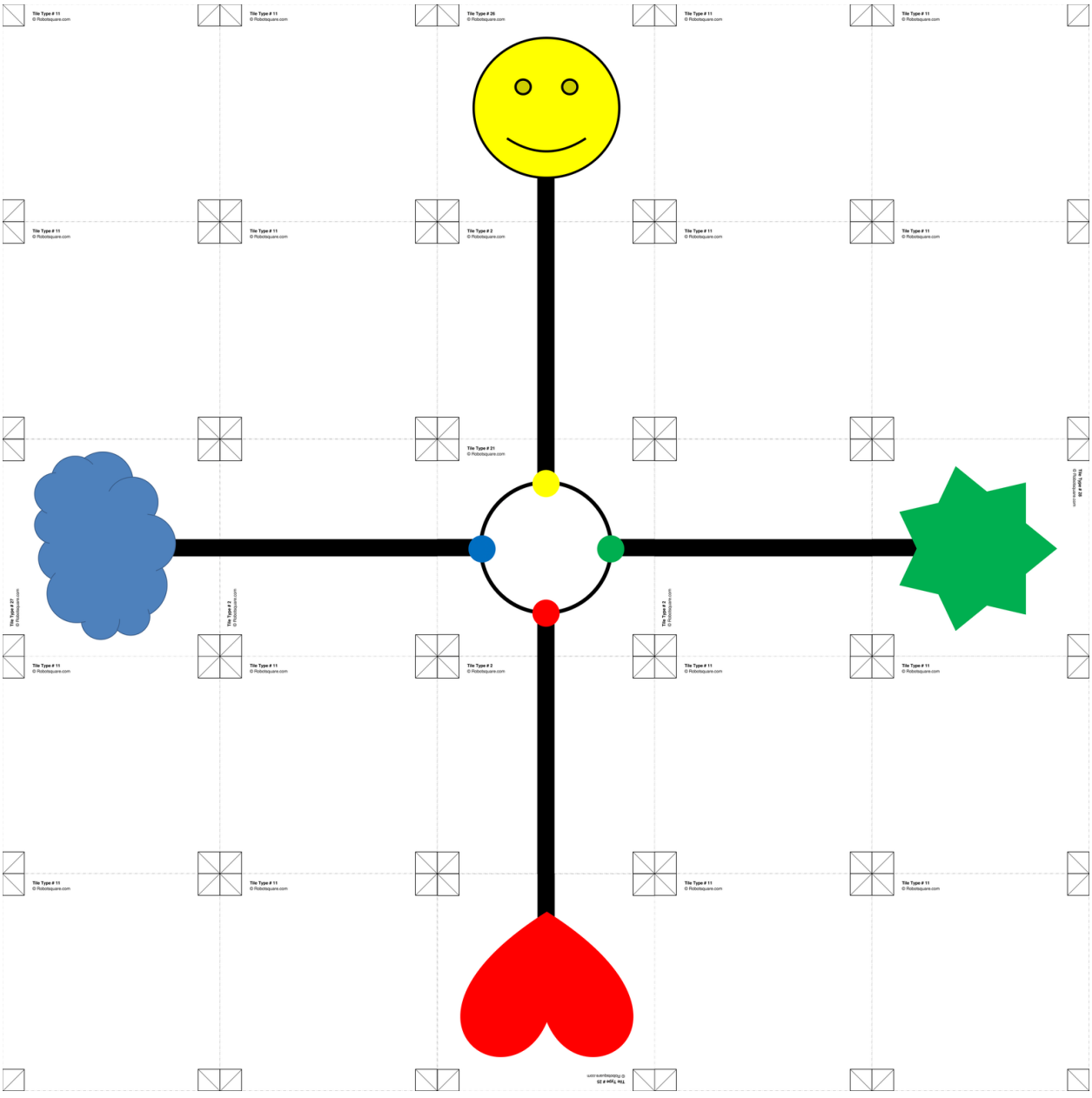


Fig. 2: Colors track

1.4.3 Cool guy track

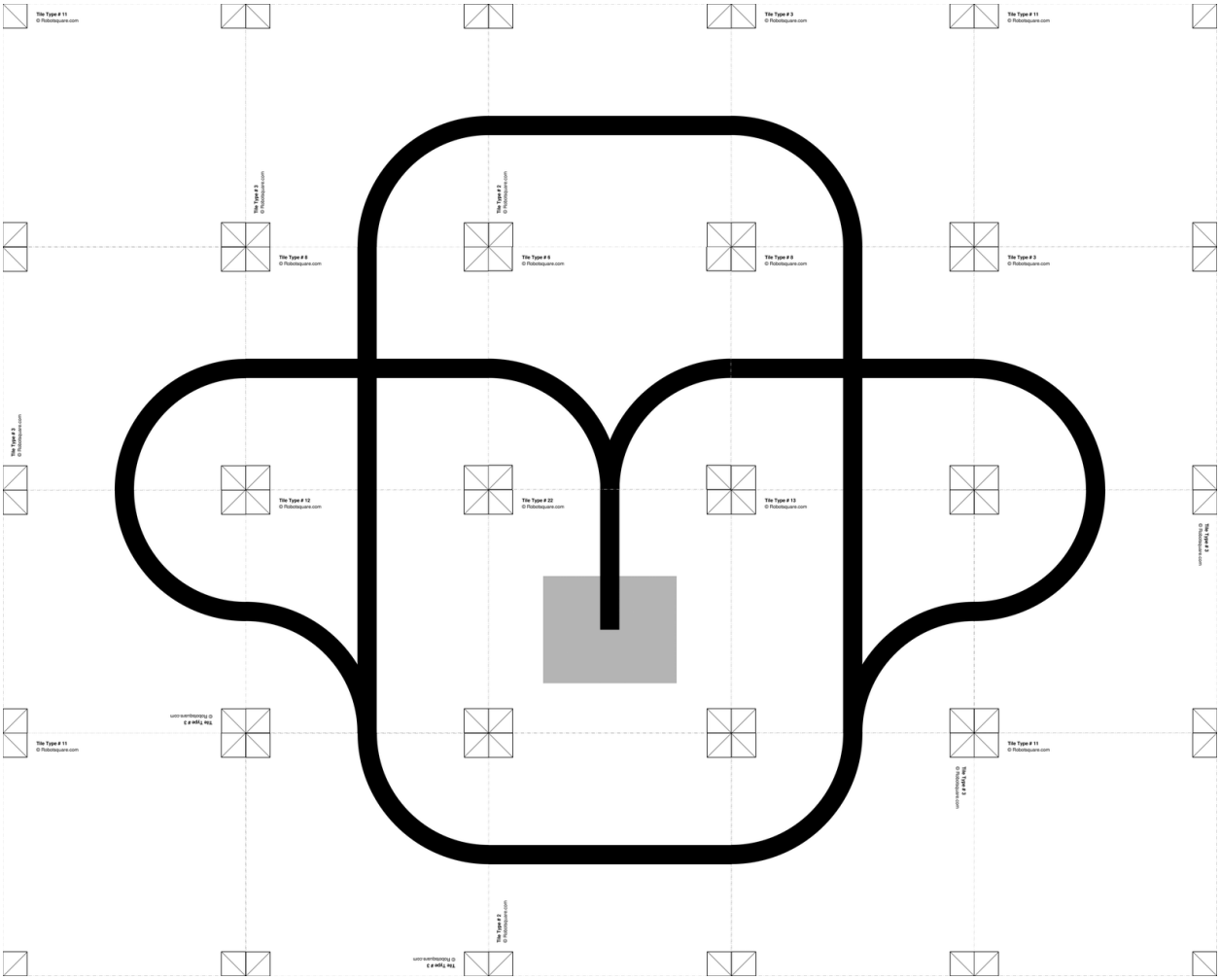


Fig. 3: Cool guy track

Description

It looks like a face…

Dimensions

Width	Height
1000 mm	800 mm

Tiles

Tile number	Number of copies required
2	2
3	8
6	1
8	2
12	1
13	1
22	1

Built with Line Track Designer

1.4.4 Hard track

Description

A very hard track

Dimensions

Width	Height
600 mm	1000 mm

Tiles

Tile number	Number of copies required
2	2
3	6
4	1
6	1
13	1
15	2
17	1
22	1

Built with Line Track Designer

GitHub repository: https://github.com/Quentin18/Line-Track-Designer/

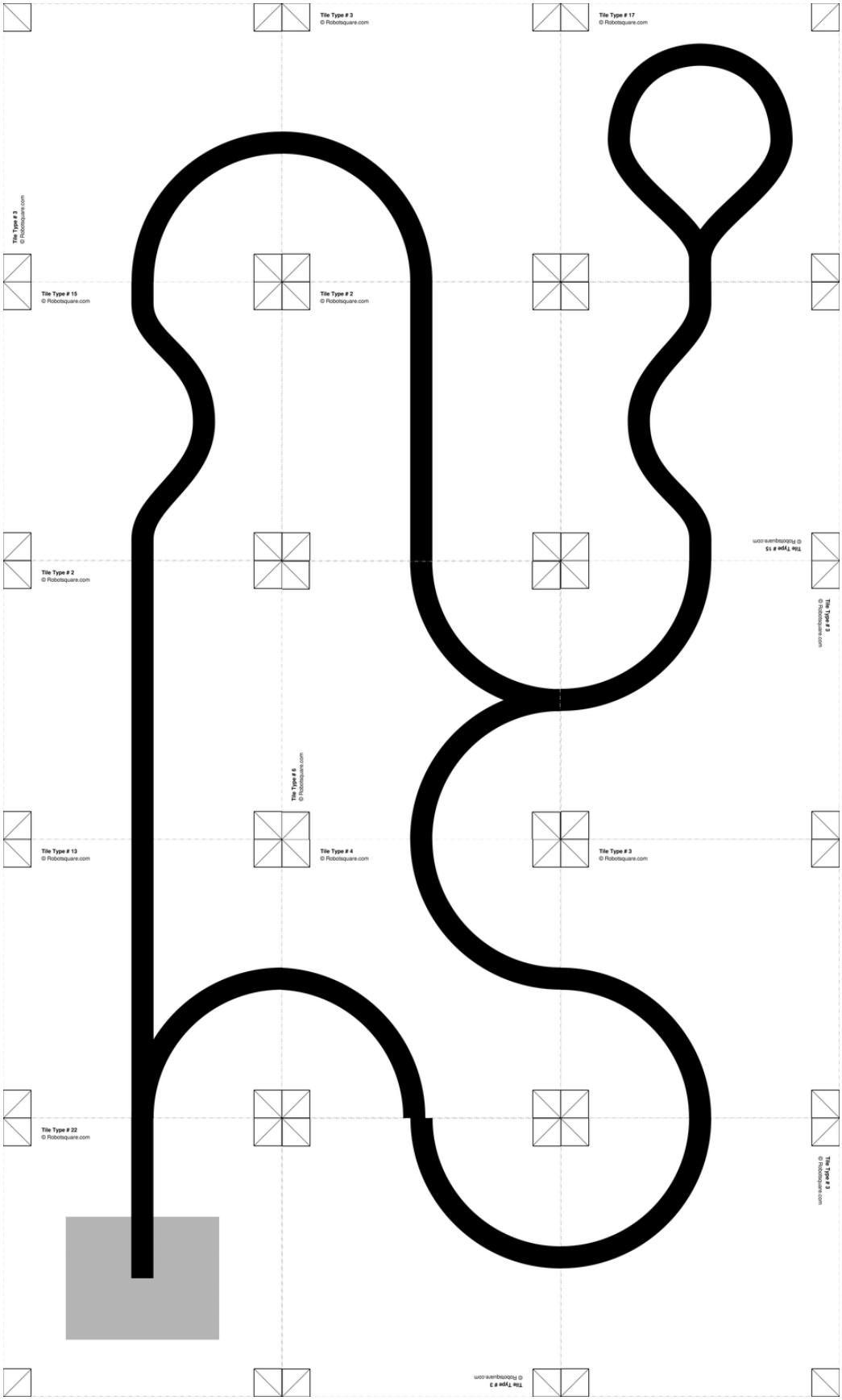


Fig. 4: Hard track

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

C

D

E

F

G

I

L

M

N

O

P

R

S

T

W

Z