

# TileMap Generator

## Documentation

Bartosz Cywiński

# TileMap Generator

|                                      |   |
|--------------------------------------|---|
| Chapter I : Description of a project | 3 |
| Short description of project         | 3 |
| Description of implementation        | 4 |
| Running a program                    | 4 |
| Creating a map                       | 5 |
| Most important methods and functions | 5 |
| Division for modules                 | 6 |
| Tests                                | 7 |
| Used libraries                       | 7 |
| Summary of a project                 | 7 |
| Chapter II : Instruction             | 7 |
| Creating new tilemaps                | 7 |
| Choosing mode                        | 7 |
| Entering size of map                 | 8 |
| Providing limits of central points   | 8 |
| Providing grounds                    | 9 |
| Visualizing and saving tilemap       | 9 |

## Chapter I: Description of a project

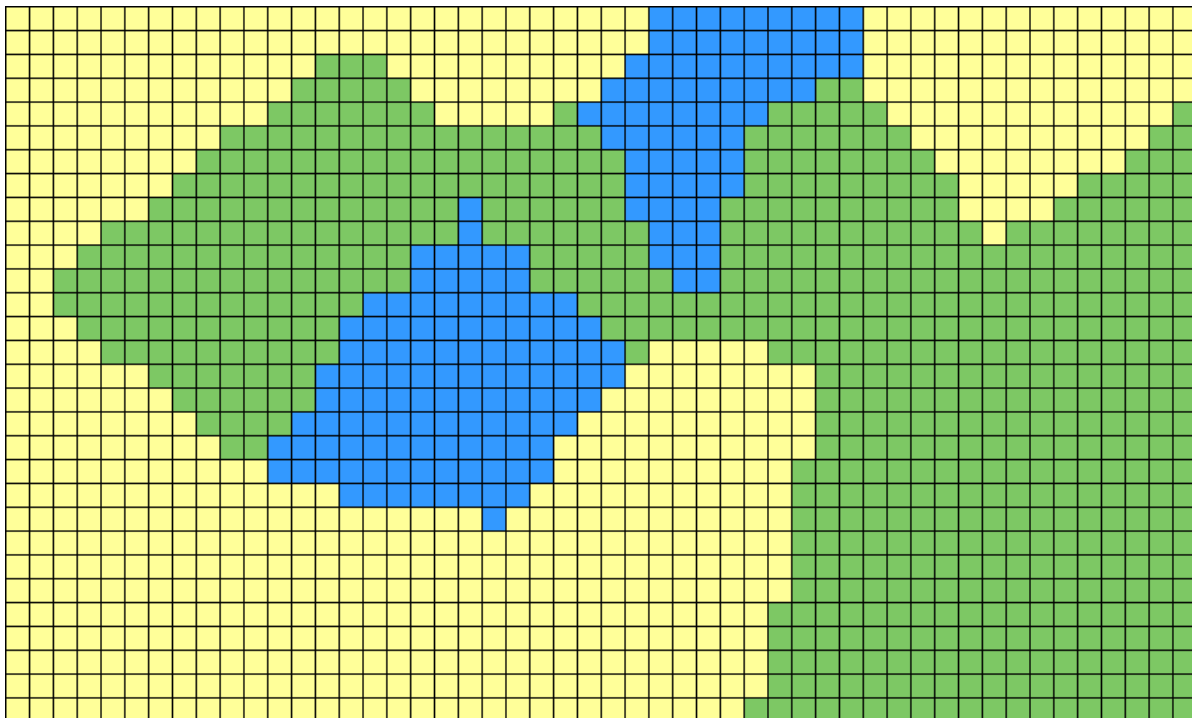
### Short description of project

Project concerns generator that makes simple 2D tile-maps. User who uses a program can provide:

- Size of map
- Type of grounds that will be applied to tile-map. There are seven basic grounds already available, but User is able to provide ground of custom color if he wants.
- Amount of "central points" of grounds (use of "central points" will be described in next point of documentation)

Moreover, User has got ability to load and visualize an existing map from file as well as saving a map User has just made.

Exemplary tile-map generated by program with three types of grounds looks like this:



## Description of implementation

### Running a program

After typing `python run.py` User will be prompted to provide information about tile-map he wants to make. There are four options of creating a map, depending on how many things User wants to specify in his map.

- First mode is to make most basic map consisting only green land and water.
- Second mode is to make map with grounds provided in program:
  - water,
  - land,
  - sand,
  - forest,
  - stone,
  - ice,
  - snow.
- Third mode is to make map with grounds provided by User. Program will prompt for names and RGB tuples of grounds.
- Fourth mode combines mode second and third.

There is also fifth mode to load an existing map from file. For every mode except fifth, User can specify limit of central points as well as save a map.

```

WELCOME IN THE TILE MAP GENERATOR!

*****
SHORT INSTRUCTION:
In Tile Map Generator you can create your own Tile Map with
desired size and types of grounds.
-----
TYPES OF GROUNDS TO CHOOSE FROM:
- 'water',
- 'land',
- 'sand',
- 'forest',
- 'stone',
- 'ice',
- 'snow'
-----
---> Type '1' to create new map with basic grounds
---> Type '2' to create new map with chosen grounds
---> Type '3' to create new map with Your own grounds
---> Type '4' to create new map with both chosen and Your own grounds
---> Type '5' to load and visualize an existing map from file
*****
After creating a new Map, you will be asked if you want to save it.

>>>Type number in: █

```

### Creating a map

After information are provided by User, class TileMap is made in steps:

1. Create NumPy zeros array in size of a map.
2. Randomly choose number of central points. Amount may be limited by User's input.
3. Randomly choose points in zeros array, converted to list, where central points will be applied.
4. For every zero in zeros array, find closest central point, and change zero to point's RGB tuple.
5. Create image from array with RGB tuples.
6. Expand image 16 times, so that pixels will be better visible.
7. Draw a black grid in image that separates every pixel from another.

### Most important methods and functions

**colors\_to\_use()** : Gets RGB tuples of grounds that will be used in tile-map stored in dictionary

**central\_pts\_to\_array()** : Gets point with two coordinates in list of zeros using **choose\_random\_coordinate()** method. Choose one of grounds' colors that will be used in that tile-map and apply to that point if that point is zero in that moment. Also makes sure that every chosen ground will appear in map.

**amount\_central\_pts()** : Staticmethod that chooses amount of central points in tile-map. Central points are "source" points of every land. Every zero in zeros array is being changed based on distance to closest central point. Amount is being chosen based on User's inputted limit of central points. Method also eliminates cases where map would not look realistic, but only in cases where User do not specify both low limit and high limit of number of central points.

**measure\_distances()** : Measures distances for every zero in zeros array to every central point and gets minimal value of that distances

**apply\_ground()** : Based on distances measured in **measure\_distances()**, applies RGB tuple of closest ground for particular zero. If more than one points are same close to zero, method chooses one randomly.

**expand\_image()** : Expands image made from array of RGB tuples 16 times

**img\_with\_grid()** : Draws black grid on expanded image

**visualize()** : Visualizes map

**save()** : Saves image in file

**load()** : Loads image from file

## Division for modules

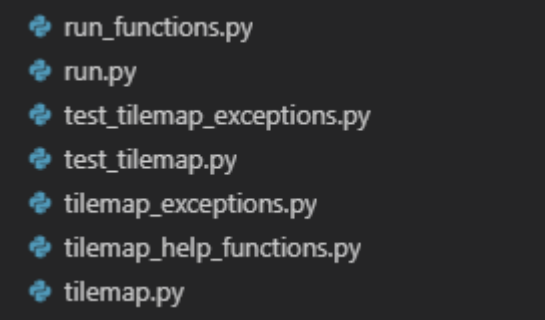
**tilemap.py** - class TileMap

**tilemap\_help\_functions.py** - functions that do not concert class TileMap directly.

**tilemap\_exceptions.py** - exceptions of TileMap class, include cases when User enters incorrect input for constructing tile-map.

**run\_functions.py** - functions that construct tile-map from User's input, parsing it and checking if is correct.

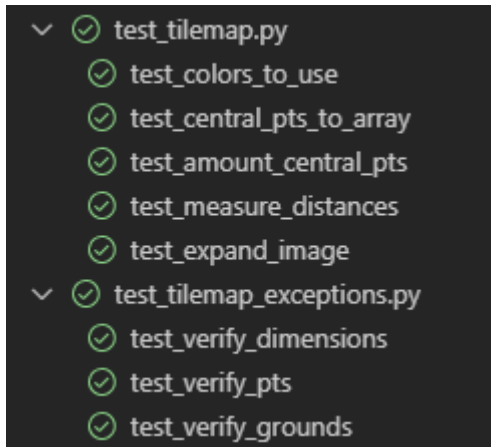
**run.py** - includes main() function and is used to run TileMap Generator



```
run_functions.py
run.py
test_tilemap_exceptions.py
test_tilemap.py
tilemap_exceptions.py
tilemap_help_functions.py
tilemap.py
```

## Tests

Generator of tile-maps in many cases is random, so test is not so numerous. They mainly test if arrays and lists are well constructed and if program works correctly with User's input.



## Used libraries

**NumPy** - used for creating arrays.

**Pillow** - used for creating maps from array and for showing them.

**random** - used for getting random points in array and for choosing random color from list of chosen grounds' colors.

**math** - sqrt function

## Summary of a project

In my opinion, TileMap Generator which I made, despite creating simple 2D tile-maps, gives User many opportunities to adjust them to his usage. By option to specify number of central points and type of grounds, User has got more control over how every tile-map will look like. Running a program shows every option of generating a map clearly so that it is easy to create a map in desired way. This project brought many experiences to me, I learned many new things about programming in Python as well as thinking about possible solutions to a problem.

## Chapter II: Instruction

### Creating new tile-maps

#### Choosing mode

After running a program (*run.py* file) if you want to create a new tile-map, you need to choose one of four modes (1-4).

- mode '1' is used to create a map with only two basic grounds ('land' and 'water')

- in mode '2' User can choose one or more grounds from provided one is by program
- in mode '3' User can provide name of grounds and its RGB tuples by his own
- mode '4' User is basically combination of mode '2' and '3' - User can both choose grounds and provide one is on his own

### Entering size of map

First User will be prompted for both width and height of desired map. Input should be an integer. Both width and height need to be at least two times greater than amount of provided grounds.

### Providing limits of central points

After choosing one of four modes, User always will be asked if he wants to limit either minimal or maximal number of central points. If he does wants to limit, he should type *y* or *yes*, otherwise he should type *n* or *no*.

#### If User wants to provide low limit of central points:

1. It cannot be greater than (if high limit not provided):

```
max((h_lim, (len(grounds)+ len(own_grounds))))
```

where:

```
h_lim = max(width,height)/2 if max(width,height) % 2 == 0 else (max(width,height)-1)/2
```

grounds - chosen grounds,

own\_grounds - grounds provided by User,

width - width of map,

height - height of map,

2. It cannot be greater than high limit (if provided),

3. It cannot be less than:

```
(len(grounds) + len(own_grounds))
```

#### If User wants to provide high limit of central points:

1. It cannot be less than (if low limit not provided):

```
max(l_lim, (len(grounds) + len(own_grounds)))
```

where:

```
l_lim = min(width,height)/2 if min(width,height) % 2 == 0 else (min(width,height)-1)/2
```

2. It cannot be less than:

```
(len(grounds) + len(own_grounds))
```

3. It cannot be less than low limit (if provided)



### Providing grounds

If User has chosen mode '2', '3' or '4', he will be prompted to enter grounds. Grounds that are being chosen from provided to User by program, should be specified in one line, separated by ", ". In mode '3' and '4' program will prompt User to first enter name than RGB tuple of custom ground.

```
Enter name of custom ground: name_of_ground  
Enter RGB tuple of custom ground: []
```

Then it will ask if User wants to enter another custom ground.

```
Do you want to enter another ground?([y]/n) []
```

### Visualizing and saving tile-map

After User enters grounds, he will be asked if map should be visualized, after that he will be asked if map should be saved in file. If User chooses to save a map, he will be asked for name of a file to save in.

```
Do you want to visualize your Tile Map?([y]/n)
```

```
Do you want to save your Tile Map?([y]/n)
```

```
Enter name of a file:
```