TileSet Tool for netpanzer V 0.2

Table of Contents

Functions

basic:

- convert between old .tls format to new
- convert between new to old .tls format

new format only:

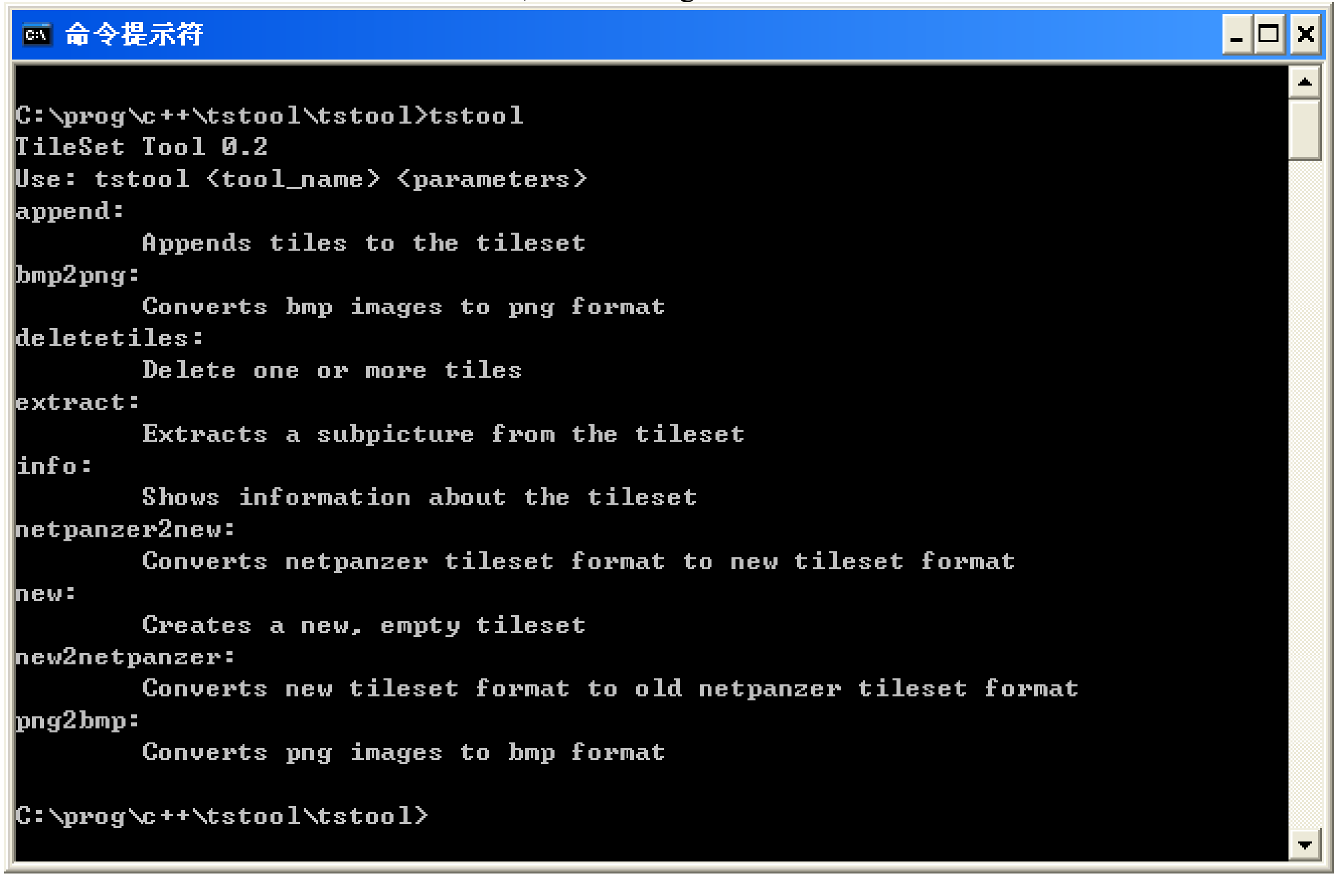
- extract a sub tileset from the tileset: this will extract the pictures that makes the tileset
- create new tileset
- append one tileset to another
- delete tiles from a tileset

old useless functions keep for no reason:

- convert bmp to png
- convert png to bmp

Usage

This tool is used from the command line, normal usage:



```
C:\prog\c++\tstool\tstool>tstool
TileSet Tool 0.2
Use: tstool <tool_name> <parameters>
append:
        Appends tiles to the tileset
bmp2png:
        Converts bmp images to png format
deletetiles:
        Delete one or more tiles
extract:
        Extracts a subpicture from the tileset
info:
        Shows information about the tileset
netpanzer2new:
        Converts netpanzer tileset format to new tileset format
new:
        Creates a new, empty tileset
new2netpanzer:
        Converts new tileset format to old netpanzer tileset format
png2bmp:
        Converts png images to bmp format

C:\prog\c++\tstool\tstool>
```

How to work with it

This version **doesn't support editing** the tileset in-place, so you need to do this (is not fast):

1. Make a script that will create a new tileset and append all the subtilesets to it
2. Edit your subtilesets (the image AND the move data)
3. Run your scripts to recreate the tileset
4. If want to see results on netpanzer < 0.8.6: convert to tls format, copy to netpanzer folder
5. Goto point 2

There is an example script "**create_summer.bat**" that does this (will create summercopy, without the conversion to netpanzer), and there is "**subpics.bat**" that will extract all the current netpanzer subtilesets so you can see/edit them (it expects the tileset name to be "newfor").

How to add a new pic

The easy way seems to be:

1. Find the dimensions (width, height) of your new pic "in tiles" (currently each tile is 32x32)
2. Extract a subtileset with those dimensions (in tiles).
3. Edit/modify the png image of the subtileset
4. Edit/modify the "move" values of the subtileset.json (ignore color)
5. Run your script to regenerate the tileset
6. Convert to tls format if desired

How to make your map editor work with the new tiles

No idea, so far I know of these editors:

- Flexlay: old editor, don't know if still works, maybe it has some config file that can be edited.
- New editor done by Wile64: newer, maybe it has some config file.

Hopefully in the distant future there will be a new web editor, let's find some js hackers.

How to convert tls format to new format:

```
tstool netpanzer2new <tlsfile> <palettefile> <new name> [row tiles]
```

- tlsfile: the path to the .tls file
- palettefile: the path to the .act file
- new name: the name for the new tileset
- row tiles: (optional, default 40) number of tiles to store in one row of the image

Full example (assumes summer12mb.tls and netp.act are in the current folder):

```
tstool netpanzer2new summer12mb.tls netp.act newfor
```

This will create 2 new files

- newfor.png: contains the images of each tile
- newfor.json: contains the data of the tileset

How to convert from the new format to the old tls format

```
tstool new2netpanzer <tileset> <outfile.tls> <outfile.act>
```

- tileset: the name of the tileset
- outfile.tls: the file to write the tls data (old format).
- outfile.act: the file to write the palette data (old format, useless).

Full example (assumes newts.png and newts.json are in the current folder):

```
tstool new2netpanzer newts othersummer.tls othersummer.act
```

This will create 2 new files

- othersummer.tls: contains the tileset in the old netpanzer format
- othersummer.act: contains the palette of the tileset for the old format, NOTE: this is actually of no use as we should try to keep the same palette as there is now.

How to extract full pictures (new format only)

```
tstool extract <tileset> <first> <xtiles> <ytiles> <new ts>
```

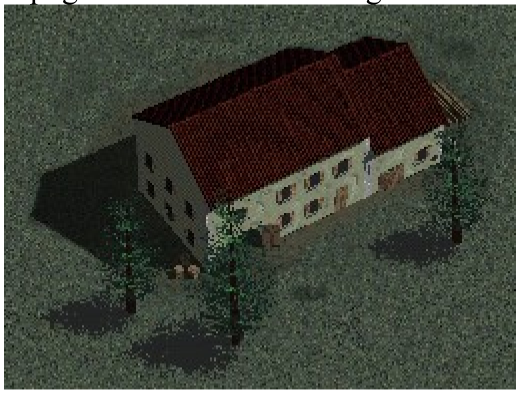
- tileset: the name of the tileset
- first: the **index** of the first tile to extract
- xtiles: how many tiles the new tileset set will have per row
- ytiles: how many rows of tiles the new tileset will have.
- new ts: the new tileset name

Full example (assumes newfor.png and newfor.json are in the current folder, directory 'data' exists and 'newfor' is a conversion of actual netpanzer tileset):

```
tstool extract newfor 576 8 6 "data/House-with-trees_01"
```

This will create 2 new files

- data/House-with-trees_01.png: contains the new image



- data/House-with-trees_01.json: contains the data

```
{
          "idheader": "data/House-with-trees_01",
          "version": 0,
          "tile_width": 32,
          "tile_height": 32,
          "tile_count": 48,
          "move": [
                1,  1,  4,  4,  4,  4,  4,  4,
                1,  4,  4,  4,  4,  4,  4,  4,
                4,  4,  4,  4,  4,  4,  4,  4,
                1,  4,  4,  4,  4,  4,  4,  4,
                1,  4,  4,  4,  4,  1,  1,  1,
                1,  1,  4,  4,  4,  1,  1,  1
          ],
          "color": [
               82, 82, 82, 76, 49, 46, 55, 82,
               82, 76, 35,  8, 19, 19, 26, 64,
               64, 23, 40, 19, 50,119, 93, 57,
               57, 20, 25, 82,119, 82, 46, 64,
               82, 46, 55, 38, 70, 82, 82, 82,
               82, 76, 51, 55, 82, 82, 82, 82
          ]
}
```

How to create a new tileset (new format only)

To create a new tileset, first you need to have ready a "subtileset", that is a png file and a json file with some format:

```
tstool new <name> <"text in header"> <subtileset> <tiles_per_row>
```

- name: the name of the new tileset
- "text in header": is some text that will be on the header of the file
- subtileset: the subtileset name
- tiles_per_row: how many tiles to store in one row of the image

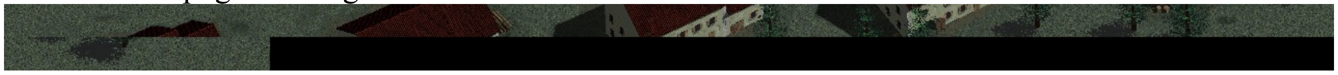
Notes: the "**color**" data from the subtileset will be ignored and recalculated for the new tileset

Full example (assumes the data comes from the "how to extract new pictures" example):

```
tstool new newts "some tileset for testing" "data/House-with-trees_01" 40
```

This will create 2 new files

- newts.png: the image data



- newts.json: the data

```
{
         "idheader": "some tileset for testing",
         "version": 1,
         "tile_width": 32,
         "tile_height": 32,
         "tile_count": 48,
         "move": [
               1,  1,  4,  4,  4,  4,  4,  4,  1,  4,  4,  4,  4,  4,  4,  4,  4,
         4,  4,  4,  4,  4,  4,  4,  1,  4,  4,  4,  4,  4,  4,  4,  1,  4,  4,
         4,  4,  1,  1,  1,
               1,  1,  4,  4,  4,  1,  1,  1
         ],
         "color": [
              93, 93, 82, 93, 93, 82, 82, 93, 82, 76, 76, 35,  3, 19, 35, 76, 82,
         65, 35,  8, 19, 19, 35, 64, 42, 12, 46, 65,162,137, 64, 57, 82, 42, 17,
         34, 76, 82, 55, 93,
              76, 40, 55, 28, 82, 82, 82, 82
         ]
}
```

How to append new tiles to a tileset (new format only)

To append new tiles you will need a "subtileset":

```
tstool append <tileset> <subtileset>
```

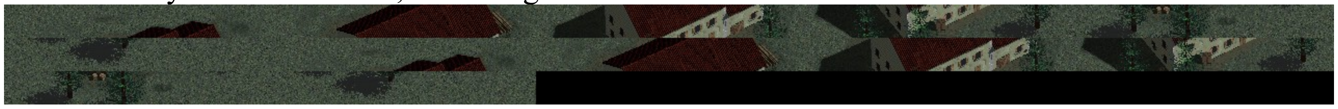
- tileset: name of the tileset where tiles will be appended
- subtileset: name of the tileset where tiles will be read

Notes: the "**color**" data from the subtileset will be ignored and recalculated for the tileset

Full example (assumes the data comes from the "how to create a new tileset" example):

```
tstool append newts "data/House-with-trees_01"
```

It will modify the newts tileset, now image is:



And the data is:

```
{
          "idheader": "some tileset for testing",
          "version": 1,
          "tile_width": 32,
          "tile_height": 32,
          "tile_count": 96,
          "move": [
                1,  1,  4,  4,  4,  4,  4,  4,  1,  4,  4,  4,  4,  4,  4,  4,  4,
          4,  4,  4,  4,  4,  4,  4,  1,  4,  4,  4,  4,  4,  4,  4,  1,  4,  4,
          4,  4,  1,  1,  1,
                1,  1,  4,  4,  4,  1,  1,  1,  1,  1,  4,  4,  4,  4,  4,  4,  1,
          4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  1,  4,  4,
          4,  4,  4,  4,  4,
                1,  4,  4,  4,  4,  1,  1,  1,  1,  1,  4,  4,  4,  1,  1,  1
          ],
          "color": [
               93, 93, 82, 93, 93, 82, 82, 93, 82, 76, 76, 35,  3, 19, 35, 76, 82,
          65, 35,  8, 19, 19, 35, 64, 42, 12, 46, 65,162,137, 64, 57, 82, 42, 17,
          34, 76, 82, 55, 93,
               76, 40, 55, 28, 82, 82, 82, 82, 93, 93, 82, 93, 93, 82, 82, 93, 82,
          76, 76, 35,  3, 19, 35, 76, 82, 65, 35,  8, 19, 19, 35, 64, 42, 12, 46,
          65,162,137, 64, 57,
               82, 42, 17, 34, 76, 82, 55, 93, 76, 40, 55, 28, 82, 82, 82, 82
          ]
}
```

How to delete tiles from a tileset (new format only)

```
tstool deletetiles <tileset> <first_tile> [to_delete:1]
```

- tileset: name of the tileset where tiles will be deleted
- first_tile: index of the first tile to be deleted
- to_delete: (optional, default to delete 1) number of tiles to delete

Full example (assumes the data comes from the “how to append new tiles to a tileset” example):

```
tstool append newts 5 80
```

It will modify the newts tileset, now image is:

And data is:

```
{
        "idheader": "some tileset for testing",
        "version": 1,
        "tile_width": 32,
        "tile_height": 32,
        "tile_count": 16,
        "move": [
             1,  1,  4,  4,  4,  1,  1,  1,  1,  1,  4,  4,  4,  1,  1,  1
        ],
        "color": [
            93, 93, 82, 93, 76, 82, 55, 93, 76, 40, 55, 28, 82, 82, 82, 82
        ]
}
```

Notes

Move values

The move values are used by the pathfinder to know which tile is walkable and which is not. Also, the walkable tiles might be “faster” or “slower” depending on the “move” value.

Currently the check is only “can move” and “cannot move”, the current used values are:

Move value	Result
0	Unit move freely
1	Unit move freely
2	Unit can move, but is slower (not really, unused)
3	Unit can move, but is even slower (not really, unused)
4	Non walkable tile, Unit cannot move here
5	Non walkable tile, Unit cannot move here