PyTiled Parser

Benjamin Kirkbride, Darren Eberly

Jan 16, 2023

CONTENTS

PyTiled Parser is a Python Library for parsing Tiled Map Editor maps and tilesets to be used as maps and levels for 2D games in a strictly typed fashion.

PyTiled Parser is not tied to any particular graphics library or game engine. It parses map files and returns arbitrary Python types(for example, `Path` objects for image files rather than a `Sprite` from a specific engine). This means it can be used to aide in implementing Tiled support into a wide variety of tools.

- Documentation available at: https://pytiled-parser.readthedocs.io/
- GitHub Project: https://github.com/pythonarcade/pytiled_parser
- PyPi: https://pypi.org/project/pytiled-parser/

The Arcade library has supporting code to integrate PyTiled Parser, as well as example code showing it's use.

CHAPTER

ONE

INSTALLATION

Simply install with pip:

```
pip install pytiled-parser
```

CHAPTER

TWO

QUICK LINKS

2.1 Programming Guide

This section will guide you through the implementation details of using PyTiled Parser. This is not an exact science, and will depend heavily on the library you are trying to integrate with. A lot of the techniques here will show some general concepts and sort of pseudo code, as well as compare to the Arcade implementation as it is the most complete implementation of this library to date.

Many liberties are taken when implementing with a game engine to suit that engines needs specifically. It is a double edged sword of using Tiled, it is very flexible and can be implemented in a lot of different ways. This comes at the cost of making a guide like this not as straight forward as it otherwise would be.

2.1.1 Loading A Map

Hello World

2.2 API Reference

This page documents the Application Programming Interface (API) for the PyTiled Parser library.

Throughout the API documentation you will see links labeled both TMX Reference and JSON Reference. These links go to the official Tiled documentation for that specific type. Sometimes there is not a one to one mapping, so it may lead to the closest thing in the Tiled format.

At some points certain classes modules may have links to other parts of the Tiled documentation which cover some of the concepts surrounding that module and it's usage within Tiled.

2.2.1 Parser

This module exposes the actual parsing functions. If you are creating an implementation, this is what you will actually pass a file to and receive back a PyTiled Parser Map or World class depending on what you're parsing.

pytiled_parser.parse_map

pytiled_parser.**parse_map**(*file: Path*) → *TiledMap*

Parse the raw Tiled map into a pytiled_parser type

Parameters
file – Path to the map file

Returns
A parsed and typed TiledMap

Return type
TiledMap

pytiled_parser.parse_world

pytiled_parser.**parse_world**(*file: Path*) → *World*

Parse the raw world file into a pytiled_parser type

Parameters
file – Path to the world file

Returns
A parsed and typed World

Return type
World

2.2.2 Common Types

This module provides some common types used throughout PyTiled Parser. These are all just NamedTuple classes provided to make sets of data more clear. As such they can be subscripted like a normal tuple to get the same values, or you can reference them by name. The values shown here are in the order they will be in the final tuple.

Color

class pytiled_parser.common_types.**Color**(*red: int*, *green: int*, *blue: int*, *alpha: int*)

Represents an RGBA color value as a four value Tuple.

red

Red value, between 0 and 255.

Type
int

green

Green value, between 0 and 255.

Type
int

blue

Blue value, between 0 and 255.

Type
int

alpha

Alpha value, between 0 and 255.

Type
int

OrderedPair

class pytiled_parser.common_types.**OrderedPair**(*x: float*, *y: float*)

Represents a two dimensional position as a two value Tuple.

x

X coordinate. Can be in either pixels or number of tiles.

Type
float

y

Y coordinate. Can be in either pixels or number of tiles.

Type
float

Size

class pytiled_parser.common_types.**Size**(*width: float*, *height: float*)

Represents a two dimensional size as a two value Tuple.

width

The width of the object. Can be in either pixels or number of tiles.

Type
float

height

The height of the object. Can be in either pixels or number of tiles.

Type
float

2.2.3 Properties

This module provides some common types used throughout PyTiled Parser. These are all just NamedTuple classes provided to make sets of data more clear. As such they can be subscripted like a normal tuple to get the same values, or you can reference them by name. The values shown here are in the order they will be in the final tuple.

Properties do not have a special class or anything associated with them. They are simply type aliases for built-in Python types.

pytiled_parser.Property

The `pytiled_parser.Property` type is a Union of the *float*, *str*, and *bool* built-in types, as well as *Path* class from pathlib, and the *pytiled_parser.Color* common type.

A property may be any one of these types.

pytiled_parser.Properties

The `pytiled_parser.Properties` type is a Dictionary mapping of *str* to *pytiled_parser.Property* objects.

When the map is parsed, all properties will be loaded in as a Property, and stored in a Properties dictionary with the name being it's key in the dictionary.

2.2.4 Tileset

This module provides an interface for Tilesets and the various objects they contain.

Also see Tiled's Docs for Editing Tilesets and the TMX Reference and JSON Reference

Tileset

class pytiled_parser.tileset.**Tileset**(*name: str*, *tile_width: int*, *tile_height: int*, *tile_count: int*, *columns: int*, *firstgid: int*, *type: str = 'tileset'*, *tile_render_size: str = 'tile'*, *fill_mode: str = 'stretch'*, *spacing: int = 0*, *margin: int = 0*, *tiled_version: Optional[str] = None*, *version: Optional[str] = None*, *image: Optional[Path] = None*, *image_width: Optional[int] = None*, *image_height: Optional[int] = None*, *transformations: Optional[Transformations] = None*, *class_: Optional[str] = None*, *background_color: Optional[Color] = None*, *tile_offset: Optional[OrderedPair] = None*, *transparent_color: Optional[Color] = None*, *grid: Optional[Grid] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool, Color]]] = None*, *tiles: Optional[Dict[int, Tile]] = None*, *wang_sets: Optional[List[WangSet]] = None*, *alignment: Optional[str] = None*)

A Tileset is a collection of tiles.

This class much more closely resembles the JSON format than TMX.

TMX Reference

JSON Reference

name

The name of this tileset.

Type
str

tile_width

The width of a tile in this tileset in pixels.

Type
int

`tile_height`

The height of a tile in this tileset in pixels.

Type
int

`tile_count`

The number of tiles in this tileset.

Type
int

`columns`

The number of tile columns in the tileset. For image collection tilesets it is editable and is used when displaying the tileset.

Type
int

`firstgid`

The global ID to give to the first Tile in this tileset. Global IDs for the rest of the tiles will increment from this number.

Type
int

`spacing`

The spacing in pixels between the tiles in this tileset (applies to the tileset image).

Type
int

`type`

Will always be *tileset*. Is statically included as a way to determine the type of a JSON file since Tiled does not have different extesnsions for map and tileset JSON files(as opposed to TMX/TSX files). This value will typically not be used.

Type
str

`spacing`

Spacing between adjacent tiles in the image in pixels. Defaults to 0.

Type
int

`margin`

Buffer between the image edge and the first tile in the image in pixels. Defaults to 0.

Type
int

`tiled_version`

The version of Tiled this tileset was saved with

Type
Optional[str]

`version`

The version of the JSON or TSX format this tileset was saved with. This will typically be the same as the tiled_version parameter, but they are tracked separately mostly for futureproofing.

Type
Optional[str]

`image`

The image file to be used for spritesheet tile sets.

Type
Optional[pathlib.Path]

`image_width`

The width of the *image* in pixels. Only set if the image parameter is. This value is taken from whatever Tiled outputs, the image size is not calculated by pytiled-parser.

Type
Optional[int]

`image_height`

The height of the *image* in pixels. Only set if the image parameter is. This value is taken from whatever Tiled outputs, the image size is not calculated by pytiled-parser.

Type
Optional[int]

`transformations`

What types of transformations are allowed on tiles within this Tileset

Type
Optional[*pytiled_parser.tileset.Transformations*]

`background_color`

The background color of the tileset. This will typically only be used by the editor, but could be useful for displaying a TileSet if you had the need to do that.

Type
Optional[*pytiled_parser.common_types.Color*]

`tileoffset`

Used to specify an offset in pixels when drawing a tile from the tileset. When not present, no offset is applied.

`transparent_color`

A color that should act as transparent on any tiles within the tileset. This would need to be taken into account by an implementation when loading the tile images.

Type
Optional[*pytiled_parser.common_types.Color*]

`grid`

Only used in case of isometric orientation, and determines how tile overlays for terrain and collision information are rendered.

Type
Optional[*pytiled_parser.tileset.Grid*]

properties

The properties for this Tileset.

Type
Optional[Dict[str, Union[float, pathlib.Path, str, bool, *pytiled_parser.common_types.Color*]]]

tiles

Dict of Tile objects with the Tile.id value as the key.

Type
Optional[Dict[int, *pytiled_parser.tileset.Tile*]]

wang_sets

List of WangSets, this is used by the terrain system in Tiled. It is unlikely an implementation in a game engine would need to use these values.

Type
Optional[List[*pytiled_parser.wang_set.WangSet*]]

alignment

Which alignment to use for tile objects from this tileset.

Type
Optional[str]

class_

The Tiled class of this TileSet.

Type
Optional[str]

tile_render_size

The size to use when rendering tiles from this tileset. Can be either "tile" or "grid".

Type
str

fill_mode

The fill mode to use when rendering tiles from this tileset. Can be either "stretch" or "preserve-aspect-fit".

Type
str

Tile

class pytiled_parser.tileset.**Tile**(**, id: int, opacity: int = 1, x: int = 0, y: int = 0, width: Optional[int] = None, height: Optional[int] = None, class_: Optional[str] = None, animation: Optional[List[*Frame*]] = None, objects: Optional[*Layer*] = None, image: Optional[Path] = None, image_width: Optional[int] = None, image_height: Optional[int] = None, properties: Optional[Dict[str, Union[float, Path, str, bool, *Color*]]] = None, tileset: Optional[*Tileset*] = None, flipped_horizontally: bool = False, flipped_diagonally: bool = False, flipped_vertically: bool = False*)

Individual tile object.

This class more closely resembles the JSON format than TMX. A number of values within this class in the TMX format are pulled into other sub-tags such as <image>.

TMX Reference

JSON Reference

id

The local tile ID within it's tileset.

Type
int

opacity

The opacity this Tiled should be rendered with.

Type
int

type

An optional, arbitrary string that can be used to denote different types of tiles. For example "wall" or "floor".

animation

A list of [Frame][pytiled_parser.tileset.Frame] objects which makeup the animation for an animated tile.

Type
Optional[List[*pytiled_parser.tileset.Frame*]]

objects

An [ObjectLayer][pytiled_parser.layer.ObjectLayer] which contains objects that can be used for custom collision on the Tile. This field is set by the Tile Collision editor in Tiled.

Type
Optional[*pytiled_parser.layer.Layer*]

image

A path to the image for this tile.

Type
Optional[pathlib.Path]

image_width

The width of this tile's image.

Type
Optional[int]

image_height

The height of this tile's image.

Type
Optional[int]

properties

A list of properties on this Tile.

Type
Optional[Dict[str, Union[float, pathlib.Path, str, bool, *pytiled_parser.common_types.Color*]]]

tileset

The [Tileset][pytiled_parser.tileset.Tileset] this tile came from.

Type
Optional[*pytiled_parser.tileset.Tileset*]

flipped_horizontally

Should this Tile be flipped horizontally?

Type
bool

flipped_diagonally

Should this Tile be flipped diagonally?

Type
bool

flipped_vertically

Should this Tile be flipped vertically?

Type
bool

class_

The Tiled class of this Tile.

Type
Optional[str]

Transformations

class pytiled_parser.tileset.**Transformations**(*, *hflip: bool = False*, *vflip: bool = False*, *rotate: bool = False*, *prefer_untransformed: bool = False*)

Transformations Object.

This is used to store what transformations may be performed on Tiles within a tileset. Within Tiled this primarily used with wang sets and the terrain system, however, could be used for any means a game/engine wants to really.

TMX Reference

JSON Reference

hflip

Allow horizontal flip?

Type
bool

vflip

Allow vertical flip?

Type
bool

rotate

Allow rotation?

Type
bool

prefer_untransformed

Should untransformed tiles be preferred?

Type
bool

Frame

class pytiled_parser.tileset.**Frame**(*tile_id: int*, *duration: int*)

Animation Frame object.

This is only used as a part of an animation for Tile objects. A frame simply points to a tile within the tileset, and gives a duration. Meaning that tile would be displayed for the given duration.

TMX Reference

JSON Reference

tile_id

The local ID of a tile within the parent tile set object.

Type
int

duration

How long in milliseconds this frame should be displayed before advancing to the next frame.

Type
int

Grid

class pytiled_parser.tileset.**Grid**(*orientation: str*, *width: int*, *height: int*)

Contains info used in isometric maps.

This element is only used in case of isometric orientation, and determines how tile overlays for terrain and collision information are rendered.

TMX Reference

JSON Reference

orientation

Orientation of the grid for the tiles in this tileset (orthogonal or isometric).

Type
str

width

Width of a grid cell.

Type
int

height

Height of a grid cell.

Type
int

2.2.5 Layer

This module provides classes for all layer types

There is the base Layer class, which TileLayer, ObjectLayer, ImageLayer, and LayerGroup all derive from. The base Layer class is never directly used, and serves only as an abstract base for common elements between all types.

For more information about Layers, see Tiled's Manual

Layer

class pytiled_parser.layer.**Layer**(*, *name: str*, *opacity: float = 1*, *visible: bool = True*, *repeat_x: bool = False*, *repeat_y: bool = False*, *coordinates:* OrderedPair *= OrderedPair(x=0, y=0)*, *parallax_factor:* OrderedPair *= OrderedPair(x=1, y=1)*, *offset:* OrderedPair *= OrderedPair(x=0, y=0)*, *id: Optional[int] = None*, *class_: Optional[str] = None*, *size: Optional[*Size*] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool,* Color*]]] = None*, *tint_color: Optional[*Color*] = None*)

Base class that all layer types inherit from. Includes common attributes between the various types of layers. This class will never be returned directly by the parser. It will always return one of the full layer types.

TMX Reference

JSON Reference

name

The name of the layer object.

Type
str

opacity

Decimal value between 0 and 1 to determine opacity. 1 is completely opaque, 0 is completely transparent. Defaults to 1.

Type
float

visible

If the layer is visible in the Tiled Editor. Defaults to True

Type
bool

coordinates

Where layer content starts in tiles. Only used by infinite maps. Defaults to (0, 0).

Type
pytiled_parser.common_types.OrderedPair

parallax_factor

Used to determine parallaxing speed of a layer. Defaults to (1, 1).

Type
pytiled_parser.common_types.OrderedPair

offset

Rendering offset of the layer object in pixels. Defaults to (0, 0).

Type
pytiled_parser.common_types.OrderedPair

`id`

Unique ID of the layer. Each layer that is added to a map gets a unique id. Even if a layer is deleted, no layer ever gets the same ID.

Type
Optional[int]

`size`

Ordered pair of size of map in tiles.

Type
Optional[*pytiled_parser.common_types.Size*]

`properties`

Properties for the layer.

Type
Optional[Dict[str, Union[float, pathlib.Path, str, bool, *pytiled_parser.common_types.Color*]]]

`tint_color`

Tint color that is multiplied with any graphics in this layer.

Type
Optional[*pytiled_parser.common_types.Color*]

`class_`

The Tiled class of this Layer.

Type
Optional[str]

`repeat_x`

Repeat drawing on the X Axis(Currently only applies to image layers)

Type
bool

`repeat_y`

Repeat drawing on the Y Axis(Currently only applies to image layers)

Type
bool

TileLayer

`class pytiled_parser.layer.TileLayer(`*, name: str, opacity: float = 1, visible: bool = True, repeat_x: bool = False, repeat_y: bool = False, coordinates:* OrderedPair *= OrderedPair(x=0, y=0), parallax_factor:* OrderedPair *= OrderedPair(x=1, y=1), offset:* OrderedPair *= OrderedPair(x=0, y=0), id: Optional[int] = None, class_: Optional[str] = None, size: Optional[*Size*] = None, properties: Optional[Dict[str, Union[float, Path, str, bool,* Color*]]] = None, tint_color: Optional[*Color*] = None, chunks: Optional[List[*Chunk*]] = None, data: Optional[List[List[int]]] = None*)

The base type of layer which stores tile data for an area of a map.

Tiled Docs

TMX Reference

JSON Reference

chunks

List of chunks (only populated for infinite maps)

Type
Optional[List[*pytiled_parser.layer.Chunk*]]

data

A two dimensional array of integers representing the global

Type
Optional[List[List[int]]]

tile IDs for the layer

Type
only populaed for non-infinite maps

Chunk

class pytiled_parser.layer.**Chunk**(*coordinates:* OrderedPair, *size:* Size, *data: List[List[int]]*)

Chunk object for infinite maps. Stores *data* like you would have in a normal TileLayer but only for the area specified by *coordinates* and *size*.

Infinite Maps Docs

TMX Reference

JSON Reference

coordinates

Location of chunk in tiles.

Type
pytiled_parser.common_types.OrderedPair

size

The size of the chunk in tiles.

Type
pytiled_parser.common_types.Size

data

The global tile IDs in the chunk. A row-first two dimensional array.

Type
List[List[int]]

ObjectLayer

class pytiled_parser.layer.**ObjectLayer**(*, *name: str*, *opacity: float = 1*, *visible: bool = True*, *repeat_x: bool = False*, *repeat_y: bool = False*, *coordinates:* OrderedPair = *OrderedPair(x=0, y=0)*, *parallax_factor:* OrderedPair = *OrderedPair(x=1, y=1)*, *offset:* OrderedPair = *OrderedPair(x=0, y=0)*, *id: Optional[int] = None*, *class_: Optional[str] = None*, *size: Optional[*Size*] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool,* Color*]]] = None*, *tint_color: Optional[*Color*] = None*, *tiled_objects: List[*TiledObject*]*, *draw_order: Optional[str] = 'topdown'*)

A Layer type which stores a list of Tiled Objects

Tiled Docs

TMX Reference

JSON Reference

tiled_objects

List of tiled_objects in the layer.

Type
List[*pytiled_parser.tiled_object.TiledObject*]

draworder

Whether the objects are drawn according to the order of the object elements in the object group element ('manual'), or sorted by their y-coordinate ('topdown'). Defaults to 'topdown'. See: https://doc.mapeditor.org/en/stable/manual/objects/#changing-stacking-order for more info.

ImageLayer

class pytiled_parser.layer.**ImageLayer**(*, *name: str*, *opacity: float = 1*, *visible: bool = True*, *repeat_x: bool = False*, *repeat_y: bool = False*, *coordinates:* OrderedPair = *OrderedPair(x=0, y=0)*, *parallax_factor:* OrderedPair = *OrderedPair(x=1, y=1)*, *offset:* OrderedPair = *OrderedPair(x=0, y=0)*, *id: Optional[int] = None*, *class_: Optional[str] = None*, *size: Optional[*Size*] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool,* Color*]]] = None*, *tint_color: Optional[*Color*] = None*, *image: Path*, *transparent_color: Optional[*Color*] = None*)

A layer type which stores a single image

Tiled Docs

TMX Reference

JSON Reference

image

The image used by this layer.

Type
pathlib.Path

transparent_color

Color that is to be made transparent on this layer.

Type
Optional[*pytiled_parser.common_types.Color*]

LayerGroup

class pytiled_parser.layer.**LayerGroup**(*, *name: str*, *opacity: float = 1*, *visible: bool = True*, *repeat_x: bool = False*, *repeat_y: bool = False*, *coordinates:* OrderedPair = *OrderedPair(x=0, y=0)*, *parallax_factor:* OrderedPair = *OrderedPair(x=1, y=1)*, *offset:* OrderedPair = *OrderedPair(x=0, y=0)*, *id: Optional[int] = None*, *class_: Optional[str] = None*, *size: Optional[*Size*] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool,* Color*]]] = None*, *tint_color: Optional[*Color*] = None*, *layers: Optional[List[*Layer*]]*)

A layer that contains layers (potentially including other LayerGroups, nested infinitely).

In Tiled, offset and opacity recursively affect child layers, however that is not enforced during parsing by pytiled_parser, and is up to the implementation how to handle recursive effects of LayerGroups

Tiled Docs

TMX Reference

JSON Reference

layers

list of layers contained in the group.

Type
Optional[List[*pytiled_parser.layer.Layer*]]

2.2.6 Objects

This module provides classes for all of the Tiled Object types.

There is the base TiledObject class, which all of the actual object types inherit from. The base TiledObject class is never directly used, and serves only as an abstract base for common elements between all types.

Some objects have no extra attributes over the base class, they exist as different classes anyways to denote the type of object, so that an implementation can load it in accordingly. For example, an Ellipse and a Point have no differing attributes from the base class, but obviously need to be handled very differently.

For more information about objects, see Tiled's Manual

Also see the TMX Reference and JSON Reference

TiledObject

class pytiled_parser.tiled_object.**TiledObject**(*, *id: int*, *coordinates:* OrderedPair, *size:* Size = *Size(width=0, height=0)*, *rotation: float = 0*, *visible: bool = True*, *name: str = ''*, *class_: str = ''*, *properties: Dict[str, Union[float, Path, str, bool,* Color*]] = {}*)

TiledObject object.

See:
https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#object

id_

Unique ID of the tiled object. Each tiled object that is placed on a map gets a unique id. Even if an tiled object was deleted, no tiled object gets the same ID.

gid

Global tiled object ID.

coordinates

The location of the tiled object in pixels.

Type
pytiled_parser.common_types.OrderedPair

size

The width of the tiled object in pixels (default: (0, 0)).

Type
pytiled_parser.common_types.Size

rotation

The rotation of the tiled object in degrees clockwise (default: 0).

Type
float

opacity

The opacity of the tiled object. (default: 1)

name

The name of the tiled object.

Type
str

class_

The Tiled class of the tiled object.

Type
str

properties

The properties of the TiledObject.

Type
Dict[str, Union[float, pathlib.Path, str, bool, *pytiled_parser.common_types.Color*]]

Ellipse

class pytiled_parser.tiled_object.**Ellipse**(*, *id: int*, *coordinates:* OrderedPair, *size:* Size = *Size(width=0, height=0)*, *rotation: float = 0*, *visible: bool = True*, *name: str = ''*, *class_: str = ''*, *properties: Dict[str, Union[float, Path, str, bool,* Color*]] = {}*)

Elipse shape defined by a point, width, height, and rotation.

See: https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#ellipse

Point

class pytiled_parser.tiled_object.**Point**(*, *id: int*, *coordinates:* OrderedPair, *size:* Size = *Size(width=0, height=0)*, *rotation: float = 0*, *visible: bool = True*, *name: str = ''*, *class_: str = ''*, *properties: Dict[str, Union[float, Path, str, bool,* Color*]] = {}*)

Point defined by a coordinate (x,y).

See: https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#point

Polygon

class pytiled_parser.tiled_object.**Polygon**(*, *id: int*, *coordinates:* OrderedPair, *size:* Size = *Size(width=0, height=0)*, *rotation: float = 0*, *visible: bool = True*, *name: str = ''*, *class_: str = ''*, *properties: Dict[str, Union[float, Path, str, bool,* Color*]] = {}*, *points: List[*OrderedPair*]*)

Polygon shape defined by a set of connections between points.

See: https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#polygon

points

List of coordinates relative to the location of the object.

Type
List[*pytiled_parser.common_types.OrderedPair*]

Polyline

class pytiled_parser.tiled_object.**Polyline**(*, *id: int*, *coordinates:* OrderedPair, *size:* Size = *Size(width=0, height=0)*, *rotation: float = 0*, *visible: bool = True*, *name: str = ''*, *class_: str = ''*, *properties: Dict[str, Union[float, Path, str, bool,* Color*]] = {}*, *points: List[*OrderedPair*]*)

Polyline defined by a set of connections between points.

See:
https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#polyline

points

List of coordinates relative to the location of the object.

Type
List[*pytiled_parser.common_types.OrderedPair*]

Rectangle

class pytiled_parser.tiled_object.**Rectangle**(*, *id: int*, *coordinates:* OrderedPair, *size:* Size = *Size(width=0, height=0)*, *rotation: float = 0*, *visible: bool = True*, *name: str = ''*, *class_: str = ''*, *properties: Dict[str, Union[float, Path, str, bool,* Color*]] = {}*)

Rectangle shape defined by a point, width, and height.

See: **https://doc.mapeditor.org/en/stable/manual/objects/#insert-rectangle**
(objects in tiled are rectangles by default, so there is no specific documentation on the tmx-map-format page for it.)

Text

class pytiled_parser.tiled_object.**Text**(*, *id: int*, *coordinates:* OrderedPair, *size:* Size = *Size(width=0, height=0)*, *rotation: float = 0*, *visible: bool = True*, *name: str = ''*, *class_: str = ''*, *properties: Dict[str, Union[float, Path, str, bool,* Color*]] = {}*, *text: str*, *color:* Color = *Color(red=255, green=255, blue=255, alpha=255)*, *font_family: str = 'sans-serif'*, *font_size: float = 16*, *bold: bool = False*, *italic: bool = False*, *kerning: bool = True*, *strike_out: bool = False*, *underline: bool = False*, *horizontal_align: str = 'left'*, *vertical_align: str = 'top'*, *wrap: bool = False*)

Text object with associated settings.

See: **https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#text**
and https://doc.mapeditor.org/en/stable/manual/objects/#insert-text

text

The text to display

Type
str

color

Color of the text. (default: (255, 255, 255, 255))

Type
pytiled_parser.common_types.Color

font_family

The font family used (default: “sans-serif”)

Type
str

font_size

The size of the font in pixels. (default: 16)

Type
float

bold

Whether the font is bold. (default: False)

Type
bool

`italic`

Whether the font is italic. (default: False)

Type
bool

`kerning`

Whether kerning should be used while rendering the text. (default: False)

Type
bool

`strike_out`

Whether the text is striked-out. (default: False)

Type
bool

`underline`

Whether the text is underlined. (default: False)

Type
bool

`horizontal_align`

Horizontal alignment of the text (default: "left")

Type
str

`vertical_align`

Vertical alignment of the text (defalt: "top")

Type
str

`wrap`

Whether word wrapping is enabled. (default: False)

Type
bool

Tile

`class pytiled_parser.tiled_object.Tile(*, id: int, coordinates: OrderedPair, size: Size = Size(width=0, height=0), rotation: float = 0, visible: bool = True, name: str = '', class_: str = '', properties: Dict[str, Union[float, Path, str, bool, Color]] = {}, gid: int, new_tileset: Optional[Union[Element, Dict[str, Any]]] = None, new_tileset_path: Optional[Path] = None)`

Tile object

See: https://doc.mapeditor.org/en/stable/manual/objects/#insert-tile

`gid`

Reference to a global tile id.

Type
int

2.2.7 Map

This module provides the primary TiledMap class which represents a single map from Tiled.

TiledMap

class pytiled_parser.tiled_map.**TiledMap**(*infinite: bool*, *layers: List[Layer]*, *map_size: Size*, *next_layer_id: Optional[int]*, *next_object_id: int*, *orientation: str*, *render_order: str*, *tiled_version: str*, *tile_size: Size*, *tilesets: Dict[int, Tileset]*, *version: str*, *parallax_origin: OrderedPair = OrderedPair(x=0, y=0)*, *map_file: Optional[Path] = None*, *class_: Optional[str] = None*, *background_color: Optional[Color] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool, Color]]] = None*, *hex_side_length: Optional[int] = None*, *stagger_axis: Optional[str] = None*, *stagger_index: Optional[str] = None*)

Object for storing a Tiled map with all associated objects.

This object is the top level object for a map. It contains all layers within a map, as well as all Tiesets used by the map. When creating an implementation, this will be the primary class to work with to pull all data relating to a map.

TMX Reference

JSON Reference

infinite

If the map is infinite or not.

Type
bool

layers

List of layer objects by draw order.

Type
List[*pytiled_parser.layer.Layer*]

map_size

The map width in tiles.

Type
pytiled_parser.common_types.Size

next_layer_id

Stores the next available ID for new layers.

Type
Optional[int]

next_object_id

Stores the next available ID for new objects.

Type
int

orientation

Map orientation. Tiled supports “orthogonal”, “isometric”, “staggered” and “hexagonal”

Type
str

`render_order`

The order in which tiles on tile layers are rendered. Valid values are right-down, right-up, left-down and left-up. In all cases, the map is drawn row-by-row. (only supported for orthogonal maps at the moment)

Type
str

`tiled_version`

The Tiled version used to save the file. May be a date (for snapshot builds).

Type
str

`tile_size`

The size of a tile.

Type
pytiled_parser.common_types.Size

`tilesets`

Dict of Tileset where Key is the firstgid and the value is the Tileset

Type
Dict[int, *pytiled_parser.tileset.Tileset*]

`version`

The JSON format version.

Type
str

`background_color`

The background color of the map.

Type
Optional[*pytiled_parser.common_types.Color*]

`properties`

The properties of the Map.

Type
Optional[Dict[str, Union[float, pathlib.Path, str, bool, *pytiled_parser.common_types.Color*]]]

`hex_side_length`

Only for hexagonal maps. Determines the width or height (depending on the staggered axis) of the tile's edge, in pixels.

Type
Optional[int]

`stagger_axis`

For staggered and hexagonal maps, determines which axis ("x" or "y") is staggered.

Type
Optional[str]

stagger_index

For staggered and hexagonal maps, determines whether the "even" or "odd" indexes along the staggered axis are shifted.

Type
Optional[str]

class_

The Tiled class of this Map.

Type
Optional[str]

parallax_origin

The point on the map to center the parallax scrolling of layers on.

Type
pytiled_parser.common_types.OrderedPair

2.2.8 Wang Set

This module contains a number of classes related to Wang Sets.

Wang Sets are the underlying data used by Tiled's terrain system. It is unlikely this module will ever need touched for creating an implementation in a game engine. It is primarily data used by the editor during map creation.

See Tiled's docs about terrain and also the TMX Reference and the JSON Reference

WangSet

class pytiled_parser.wang_set.**WangSet**(*name: str*, *tile: int*, *wang_type: str*, *wang_tiles: Dict[int, WangTile]*, *wang_colors: List[WangColor]*, *class_: Optional[str] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool, Color]]] = None*)

A complete Wang Set defining a list of corner and edge [WangColors][pytiled_parser.wang_set.WangColor], and any number of [WangTiles][pytiled_parser.wang_set.WangTile]

[TMX Reference](https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#wangset)

[JSON Reference](https://doc.mapeditor.org/en/stable/reference/json-map-format/#wang-set)

name

Name of the WangSet

Type
str

properties

The properties for this wang set

Type
Optional[Dict[str, Union[float, pathlib.Path, str, bool, *pytiled_parser.common_types.Color*]]]

class_

The Tiled class of this Wang Set

Type
Optional[str]

tile

Tile ID of the tile representing this Wang Set

Type
int

wang_colors

A list of [WangColors][pytiled_parser.wang_set.WangColor]

Type
List[*pytiled_parser.wang_set.WangColor*]

wang_tiles

A list of [WangTiles][pytiled_parser.wang_set.WangTile]

Type
Dict[int, *pytiled_parser.wang_set.WangTile*]

wang_type

A string noting the type of this wang set

Type
str

WangColor

class pytiled_parser.wang_set.**WangColor**(*color: Color*, *name: str*, *probability: float*, *tile: int*, *class_: Optional[str] = None*, *properties: Optional[Dict[str, Union[float, Path, str, bool, Color]]] = None*)

A color that can be used to define the corner and/or edge of a Wang tile

[TMX Reference](https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#wangcolor)

[JSON Reference](https://doc.mapeditor.org/en/stable/reference/json-map-format/#wang-color)

color

An RGBA color used to identify this Wang color

Type
pytiled_parser.common_types.Color

name

The name for this color

Type
str

probability

The probability used when randomizing tiles

Type
float

properties

The properties for this wang color

Type
Optional[Dict[str, Union[float, pathlib.Path, str, bool, *pytiled_parser.common_types.Color*]]]

class_

The Tiled class of this Wang Color

Type
Optional[str]

tile

Tile ID of the tile representing this color

Type
int

WangTile

class pytiled_parser.wang_set.**WangTile**(*tile_id: int*, *wang_id: List[int]*)

Defines a Wang tile by linking a tile in the tileset to a Wang ID.

[TMX Reference](https://doc.mapeditor.org/en/stable/reference/tmx-map-format/#wangtile)

[JSON Reference](https://doc.mapeditor.org/en/stable/reference/json-map-format/#wang-tile)

tile_id

The ID of the tile this WangTile maps to

Type
int

wang_id

The wang color indexes for this tile. This is a list of IDs referring to colors within the wang set. In the order of top, top right, right, bottom right, bottom, bottom left, left, top left.

Type
List[int]

2.2.9 World

This module provides an implementation for the World files from Tiled.

See Tiled's docs for Worlds for more info about worlds and what they do.

The functionality within PyTiled Parser is to load the world and outline the size and position of each map, and provide the path to the map file. Loading a world does not automatically load each map within the world, this is so that the game or engine implementation can decide how to handle map loading.

WorldMap

class pytiled_parser.world.**WorldMap**(*map_file: Path*, *size:* Size, *coordinates:* OrderedPair)

Represents a map within a world.

This object can be accessed to load in a map after loading the world.

This class translates to each object within the *maps* list of a World file

map_file

A Path object to the map file, can be passed to the pytiled_parser.parse_map function later.

Type
pathlib.Path

size

The size of the map in pixels

Type
pytiled_parser.common_types.Size

coordinates

The position of the map within the world in pixels

Type
pytiled_parser.common_types.OrderedPair

World

class pytiled_parser.world.**World**(*maps: List[*WorldMap*]*, *only_show_adjacent: bool = False*)

Represents a world file.

maps

The list of maps within the world. These are not fully parsed TiledMap objects, but rather WorldMap objects which can be used to later parse each individual map.

Type
List[*pytiled_parser.world.WorldMap*]

only_show_adjacent

Largely only used by the Tiled editor to determine if only maps adjacent to the active one should be displayed. This could be used to determine implementation behavior as well though to toggle auto-loading of adjacent maps or something of the sort.

Type
bool

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

T

U

V

W

X

Y