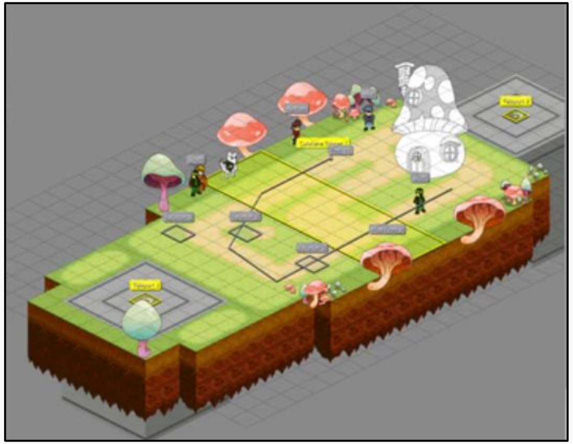
# Game dev: Map metadata

Ricard Pillosu - UPC

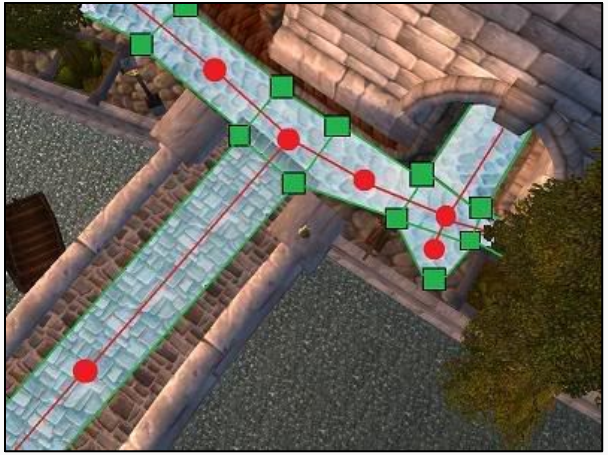
# Map metadata

- Every Tiled map may contain several layers of information
- Every layer can be graphic information or metadata
  - Audio: which audio environment to play
  - Spawn points for enemies / player
  - Doors / Item / Traps / Secrets placement
  - Invisible colliders to detect the player finished the level
  - Etc..



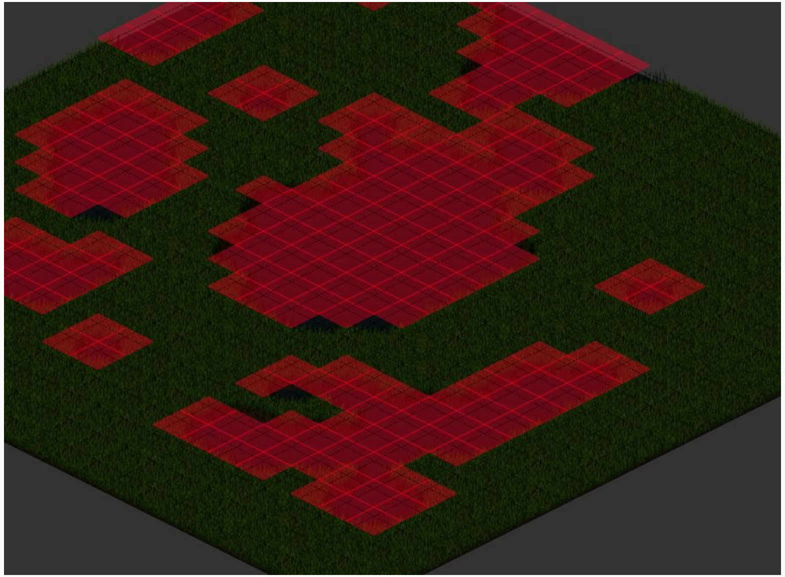
# What is a navigation mesh ?

- It is a **logic** layer with a simplification of the map.
- It contains data per tile about navigation:
  - Is it blocked ?
  - Is it slow ?
  - Is it next to a cliff ?
  - *<insert anything that you need for your game>*



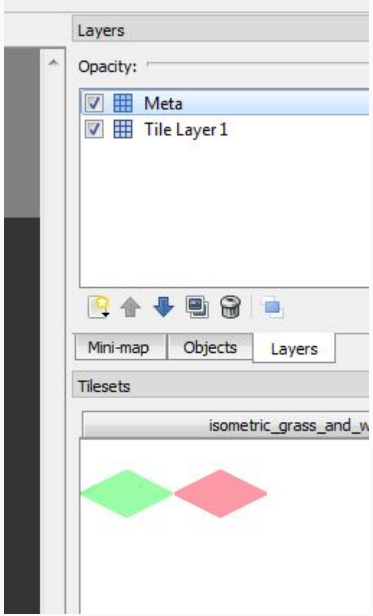
# Plan of attack!

- Create a new layer in Tiled for navigation data.
- Read it in the code to create the navigation mesh.
- For now, we will only have blocked (red) or not blocked (nothing).



# TODO 1

- Open Tiled :)
- Create a new *Tiled Layer*.
- Create a new tileset with *maps/meta.png* (64x64)
- Use the red to paint tiles with water.



Layers

Opacity:

☑ Meta

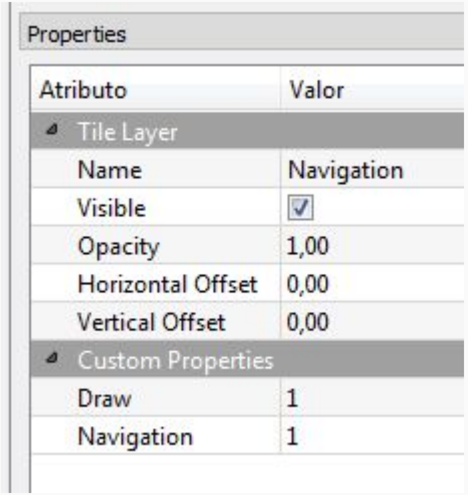
☑ Tile Layer 1

Tilesets

isometric_grass_and_w

# TODO 2

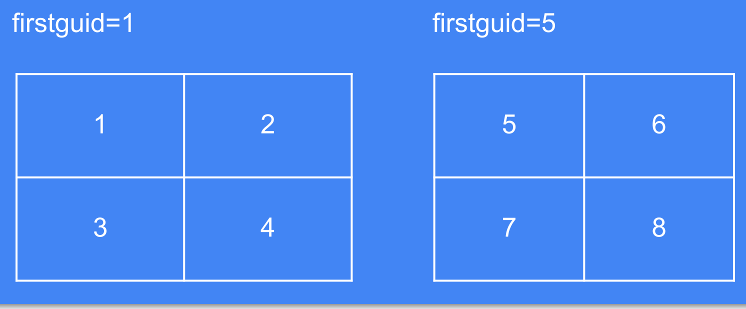
- Create a custom property, so you know this layer is for navigation data.
- And another one to know that this layer should not be drawn
- Save it as a new TMX and find where *Custom Properties* are.



Properties

| Atributo | Valor |
|---|---|
| Tile Layer | |
| Name | Navigation |
| Visible | ☑ |
| Opacity | 1,00 |
| Horizontal Offset | 0,00 |
| Vertical Offset | 0,00 |
| Custom Properties | |
| Draw | 1 |
| Navigation | 1 |

# TODO 3



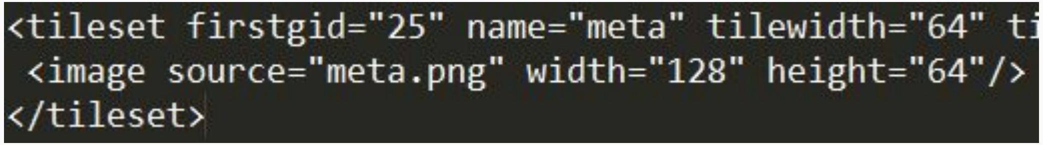
firstguid=1

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

firstguid=5

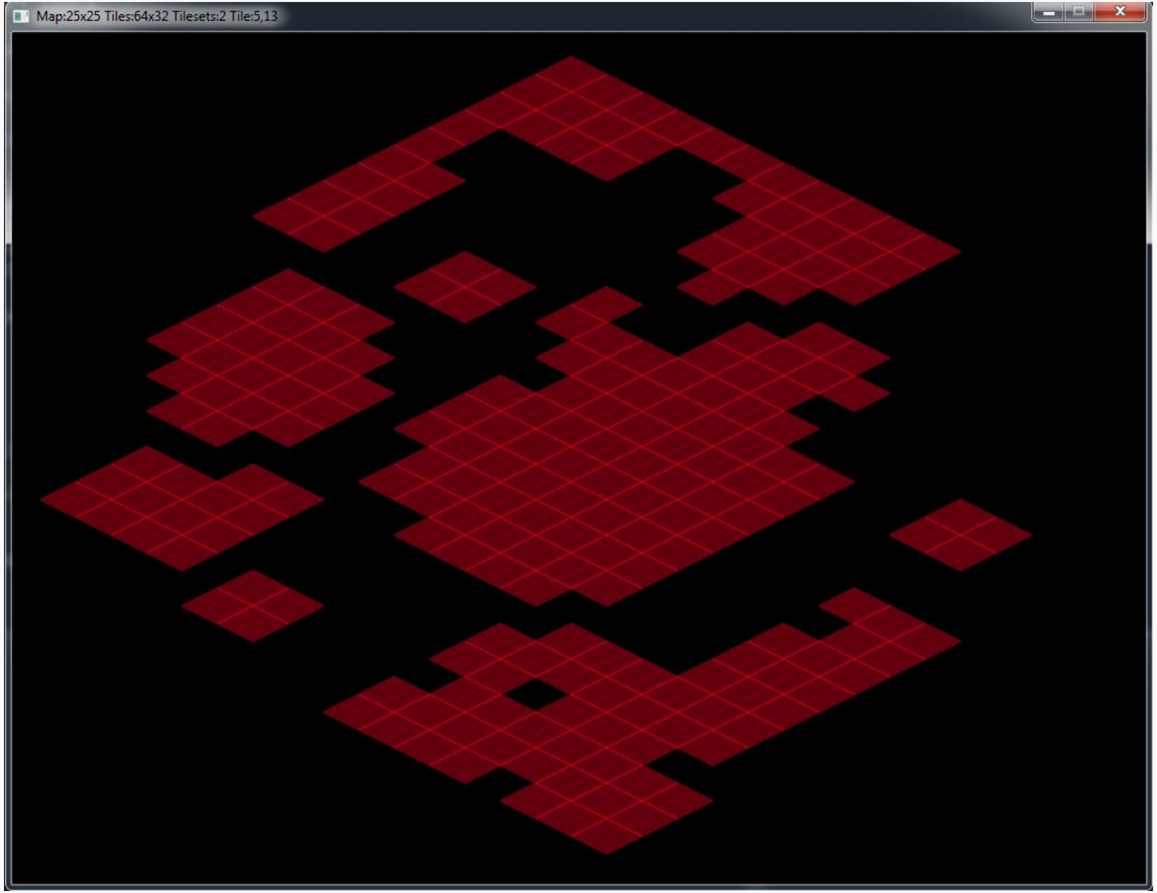
| | |
|---|---|
| 5 | 6 |
| 7 | 8 |

- Notice that we are using more than one tileset
- Our code reads them but we cannot draw them!
- The attribute **firstgid** sets the first tile id that this tileset contains
- Knowing this, fill the method that returns the proper tileset attached to any tile id.



```
<tileset firstgid="25" name="meta" tilewidth="64" ti
 <image source="meta.png" width="128" height="64"/>
</tileset>
```

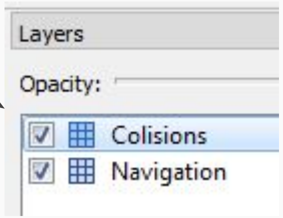
# TODO 3 : we can only draw one layer



Map:25x25 Tiles:64x32 Tilesets:2 Tile:5,13

# TODO 4

- Adapt the draw function to be able to draw all the layers.
- Just iterate all layers and draw
- You should see the isometric map, the navigation layer is below!
- Switch the layer order in Tiled



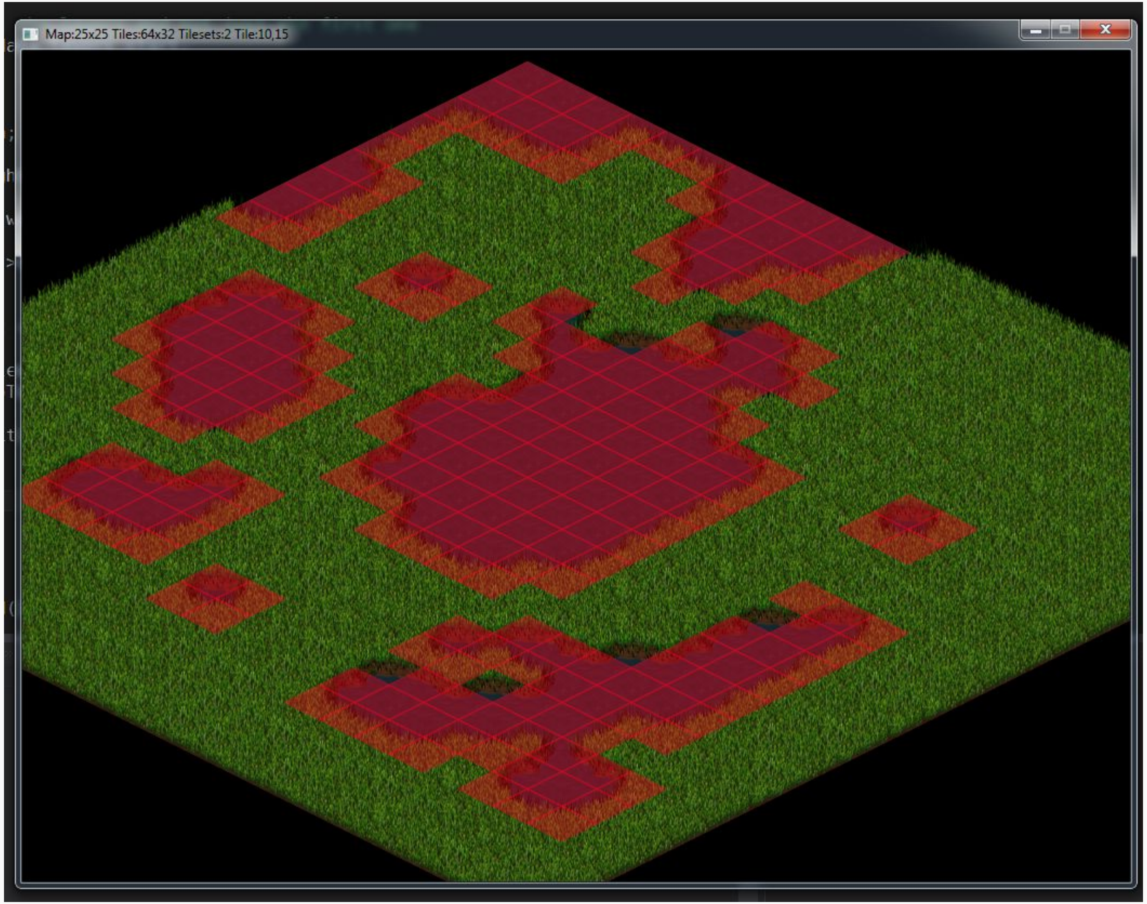
Layers

Opacity:

☑ Colisions

☑ Navigation

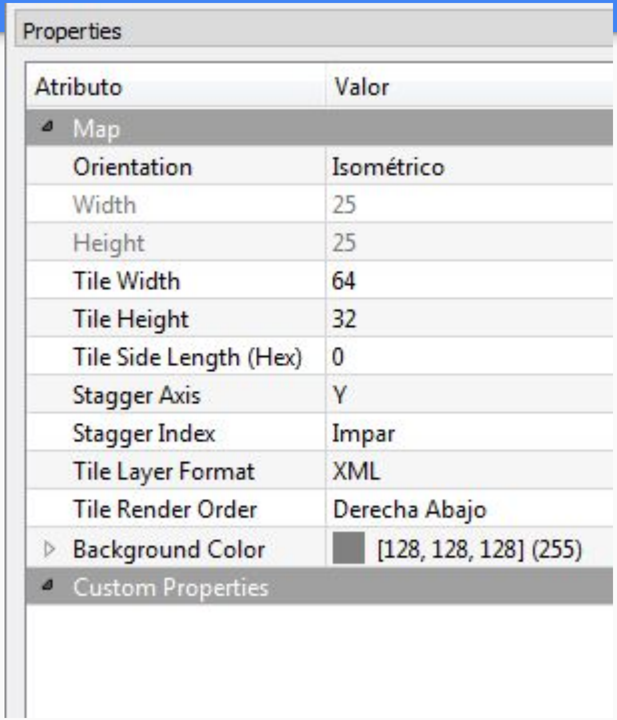
# TODO 4 : all layers drawn using multiple tilesets



Map:25x25 Tiles:64x32 Tilesets:2 Tile:10,15

# TODO 5

- In Tiled, many things might have custom properties:
  - Map
  - Each tile of a tileset
  - Layers
  - Terrains
- Create a *generic* structure in *j1Map.h* to hold custom properties



Properties

| Atributo | Valor |
|---|---|
| Map | |
| Orientation | Isométrico |
| Width | 25 |
| Height | 25 |
| Tile Width | 64 |
| Tile Height | 32 |
| Tile Side Length (Hex) | 0 |
| Stagger Axis | Y |
| Stagger Index | Impar |
| Tile Layer Format | XML |
| Tile Render Order | Derecha Abajo |
| Background Color | ■ [128, 128, 128] (255) |
| Custom Properties | |

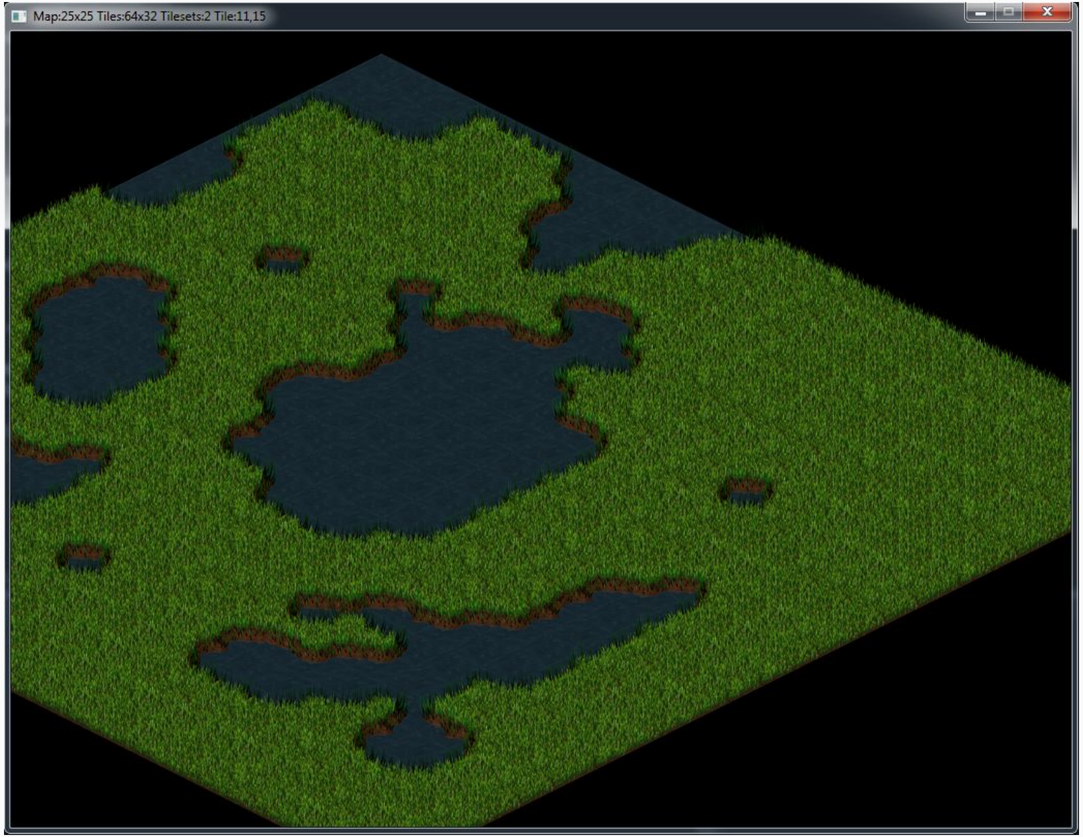
# TODO 6

- Fill the method to load custom properties
- Should receive a xml node
- Would it be better to be a method in the Custom Property class/struct ?
- Add the calls to this method so we load custom properties for each layer

# TODO 7

- Add a method to custom properties struct.
- It should be to request any value.
- It must accept a default value in case it is not found (in the lines of pugui xml).
- Now use it to ask each layer if your “Draw” property is true.
- Test if it works changing the values in Tiled and checking the result.

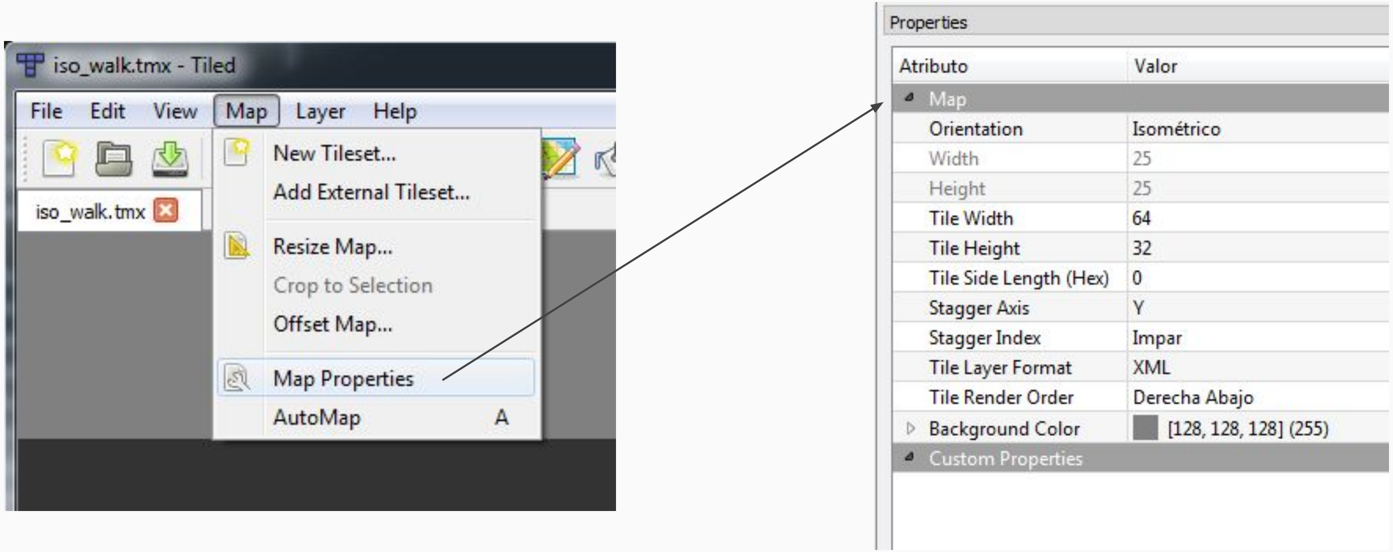
# TODO 7: Not spectacular, but now we have metadata loaded!



Map:25x25 Tiles:64x32 Tilesets:2 Tile:11,15

# Homework

- Enable the code to read custom properties from the Map



Properties

| Atributo | Valor |
|---|---|
| Map | |
| Orientation | Isométrico |
| Width | 25 |
| Height | 25 |
| Tile Width | 64 |
| Tile Height | 32 |
| Tile Side Length (Hex) | 0 |
| Stagger Axis | Y |
| Stagger Index | Impar |
| Tile Layer Format | XML |
| Tile Render Order | Derecha Abajo |
| Background Color | [128, 128, 128] (255) |
| Custom Properties | |