Map and Minimap

How to use Map and Minimap v1.01

There are a few steps you must follow to make this asset work. This document will explain these steps. If you ever feel lost, check at the Demo scene to understand how everything is setup.

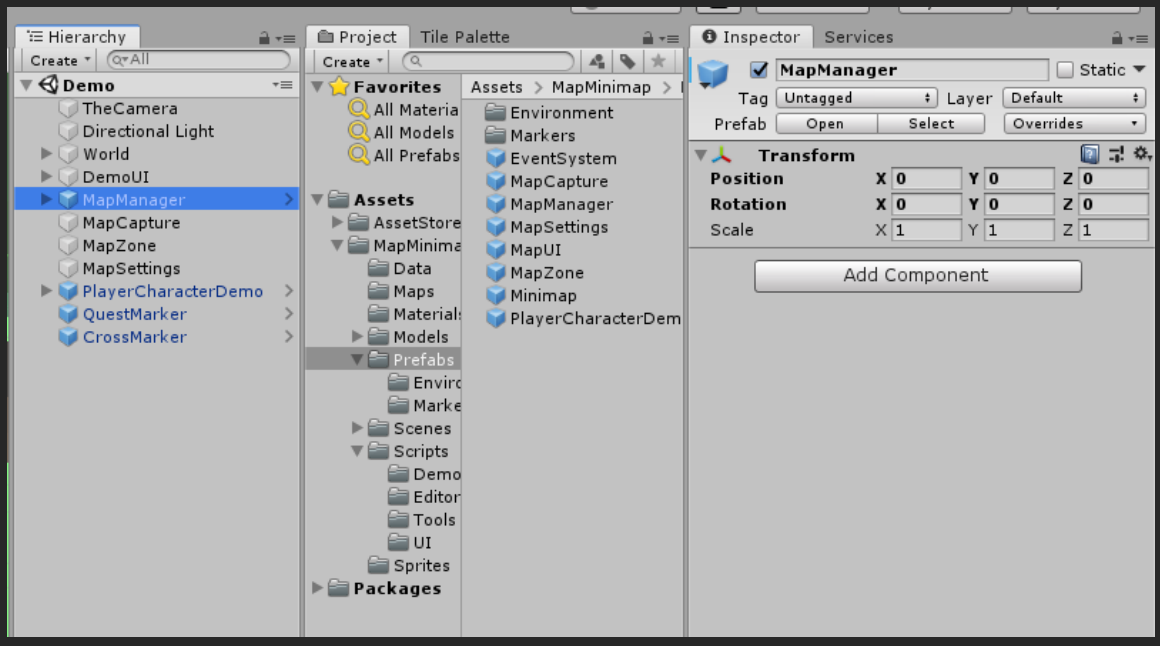
Getting Started

There are a few things you will need to start using the map or minimap:

- 1) Add the MapManager to your scene
- 2) Add a MapZone to the scene and have it cover the area that match your map.
- 3) Add the MapLevelSettings to your scene, and setup its properties (map and zone).
- 4) Add the MapIcon script to your player character and set it to 'Player'.

1) MapManager

Must be included once in your scene when you want to use this asset. It can be found in the prefabs folder.
This manager loads the UI (MapUI), handles controls, and has some generic functions for this asset.

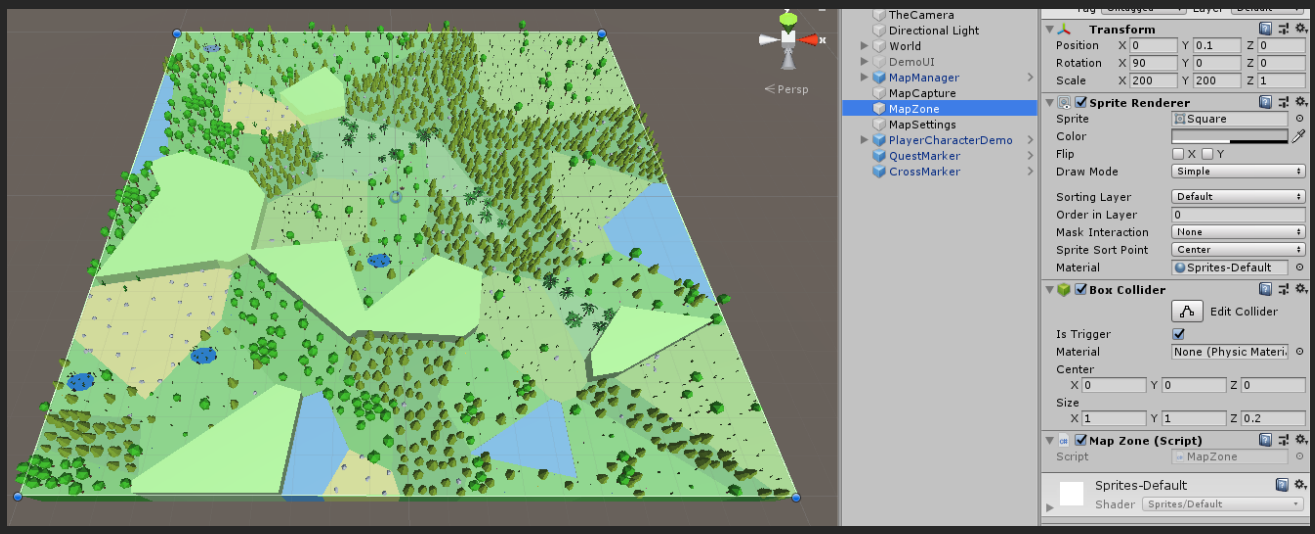


How to edit the Map UI?

You can do this by opening the MapUI prefab contained in the Prefabs folder for the full map.
For the minimap, open the Minimap prefab instead.

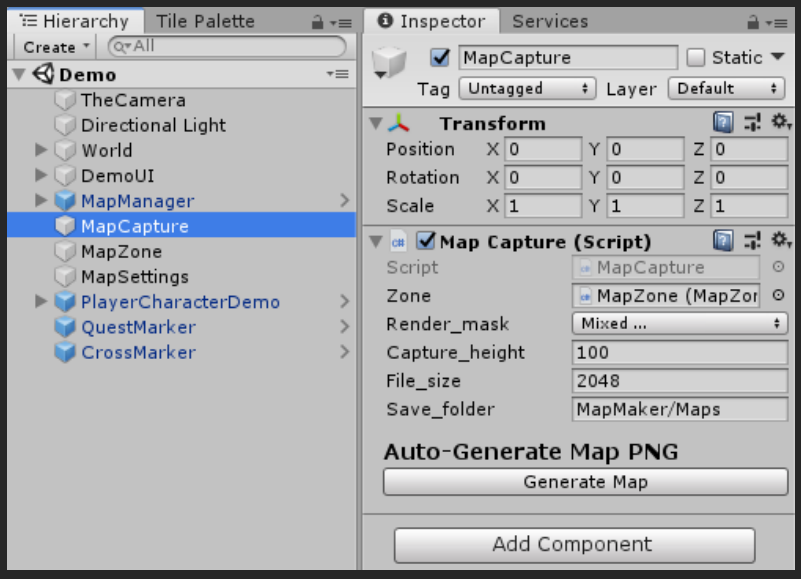
2) Map Zone

The map zone is necessary to have scene objects appear on the map as icon. It serves as a reference to know the boundaries of the map, and convert world positions to map positions. There can only be one map zone per scene. It's important that you adjust it's size to fit your play area. Map zone should always be in the X-Z axis, with Y+ being upward.



Map Capture

You may want to draw your own map image, but there is also a way to generate the map sprite automatically by using the MapCapture.cs script. Add the script to your scene, and in editor mode, click on "Generate Map" to generate a map based on the zone you setup. This auto-generated image may also serve as a template to draw your own map. This tool works in Editor mode only.



Zone: the map zone

RenderMask: You may want to hide some layers when generating the map.

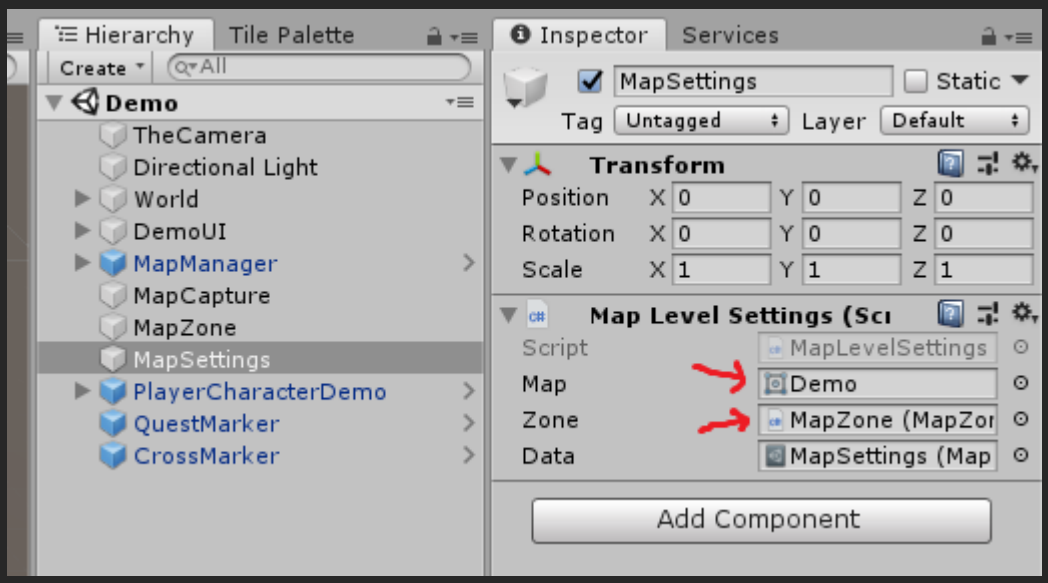
Capture Height: Position in Y from which the capture will be taken from. Just make sure it's high enough to capture everything.

File Size: Size in pixel of the file that will be created.

Save Folder: Folder to save the image, relative to the Assets folder.

3) Map Level Settings

Each scene will have their own MapLevelSettings, it's basically a reference to the image and zone you are using for this scene, make sure they are set properly!



Map Settings Data

Additional settings that are common to all scenes.

Zoom Max: how much you can zoom in on the map, for example a value of 4 will mean the maxed out zoom will be x4 closer than the full map view.

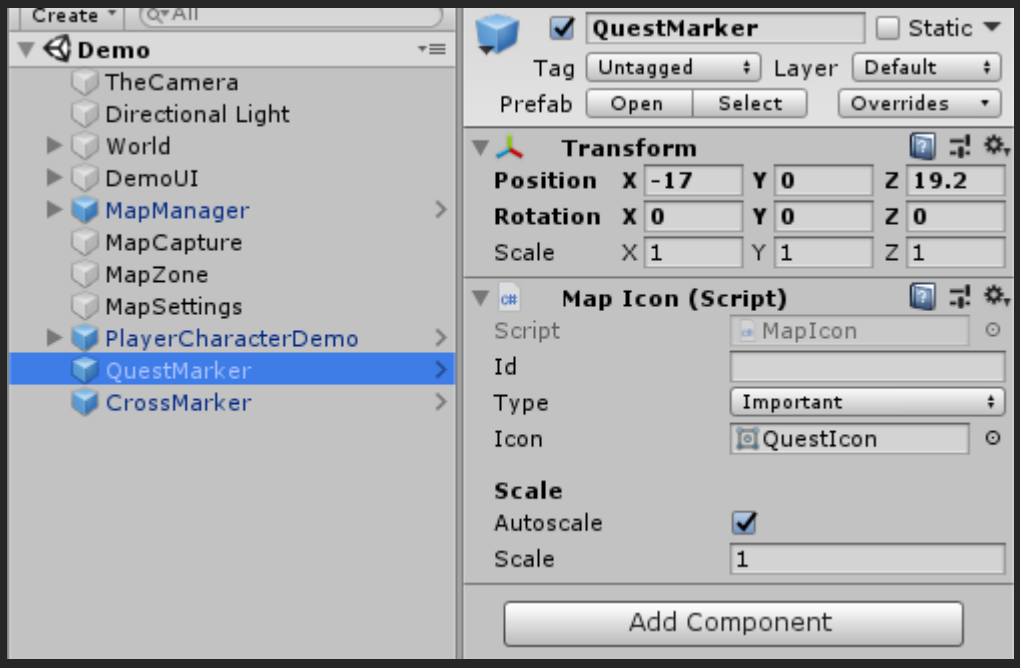
Icon Size: Generic default icon size in pixels. This settings is for all icons, there is also another setting on the MapIcon script to resize just one specific icon.

Fog: Toggle the fog on/off
Fog Reveal Radius: Distance around the player that is revealed.
Fog Precision: How many fog 'waypoint' will be stored in memory (as either revealed or not reveled), higher values will make the fog more accurate but may make performances slower.
Fog Update Rate: Interval in seconds of when the game check for fog reveal (in seconds). Lower value will require higher performance. A value of 0 means every frame.

4) Map Icon

MapIcon.cs is a script that you can attach to any object in your scene, to have it appear as an icon on the map. This is cool because if the object is moving in your scene, it will also move automatically on the map.

Player Icon: One Map Icon is mendatory: you must add this script to your player, with its type set to 'Player'. This is important because this will determine where to reveal the fog, and where to position the map by default when you open it.



ID: optional, allow you to retrieve a specific icon by script.
Type: Default, Player, or Important. Player is for your player character, you can only have one of those. Important will make the icon display OVER the fog, instead of under it.
Icon: The sprite icon.
Autoscale: if true, will appear bigger when zooming out on the map.
Scale: multiplies the scale in the settings data by this value. A value of 1 will have the scale unchanged.

If you want to add a new icon by script, you can call MapIcon.Create()
If you want to delete one, you can do MapIcon.Get(id) to retrieve one previously created, and then use the default unity function: Destroy(map_icon.gameObject) to destroy it.

5) Opening the map

You can setup your UI any way you like, you can also check the demo how it's done to open the map. By default, pressing on M will toggle it (you can change this in the Manager controls). You can also call the function OpenMap() in the MapManager: MapManager.Get().OpenMap();

To add a minimap, drag the Minimap prefab into your UI. (it's in the Prefabs/UI folder).

UI Scripts

MapUI
Top level UI script, just for toggle the panel of the full map on and off.

Minimap
Top level UI script for the minimap, will navigate the map position automatically.

MapViewer
Used by both the minimap and full map, it allows to navigate the map texture (move and zoom), and encapsulate it inside a frame. It must contain a child object with a MapRenderer
Move/Zoom speed: speed of move and zoom.
Locked: If true the map can't be navigated manually (only by script).

MapRenderer
This script will draw the map into a render texture, and then draw all the map icons.

Render Texture Size: Size in pixel of the texture where the map with icons is rendered during runtime. Higher resolution will give higher quality map but require more memory and performance.
Refresh_rate: Interval between each refresh, in seconds. A value of 0 means every frame.
Icon_scale: multiplies the icon size for this renderer. Useful if you want the minimap to have smaller icons than the full map for example.
Multi_resolution: When on, instead of trying to render the whole map everytime, the render frame will be adapted to current position and zoom, so that you get much higher resolution when zooming on the map.

FogRenderer
This script will draw the fog based on the fog 'waypoints' that are currently revealed. It's properties are similar to the map renderer. This script is optional.

MapData
Contains all values that are generated in game that should be stored to preserve a game session. For example, all the revealed 'fog waypoints' are stored there. There is a Save and Load functions to save this data to a file, but if you want to use these you must create your own Save UI or link it to your existing save system.

Improving the System

If you notice a missing feature, and you think that it would be really helpful for you. Please let me know about it. I really want to improve this system and make it great! And since I can't predict all the use cases, your feedback would really help me know what I should include in the future versions.

You can visit my youtube channel for tutorials:
https://www.youtube.com/channel/UC0LYig0AgPT9T5IN5DCUiqQ

If you have any questions or suggestions, join the discord, or send me an email:
Discord: https://discord.gg/JpVwUqG
Email: contact@indiemarc.com

Thank you!

Credits

Indie Marc (Marc-Antoine Desbiens)
Freelance Game Developer
indiemarc.com