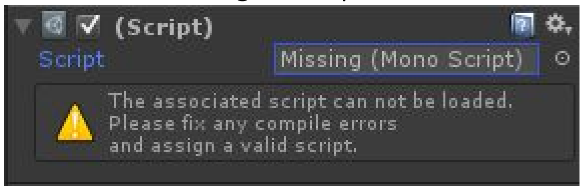
GO Map – 3D Map for AR Gaming
By Alan Grant

2.0 - 2.1 Upgrade

The 2.1 version of GoMap introduces a new folder structure preparatory to a some new plugins and addons that will come in the next months.
Upgrading from the 2.0 (or earlier version) with the unity updater could mess up your project and break/duplicate classes so my advice is to do the following steps:

- Delete the GoMap - 3D Map or AR gaming folder of your project
- import the new GoMap package from the asset store.
- Open your scene, or any of the demo scenes, and relink the LocationManager script in the LocationManager gameobject by clicking next to the "missing (mono script)" and choosing the LocationManager component.


(Script)
Script Missing (Mono Script)
The associated script can not be loaded.
Please fix any compile errors
and assign a valid script.

- Press play and the map will load just fine.
- If the character model is missing too, please just re-import it into the scene and link it to the MoveAvatar.cs and GoOrbit.cs scripts.

Documentation

GO Map is a real-time 3D map renderer for AR Gaming purposes.
It's really easy to use and customize, just build it on your device and go outside to try!
Or use it inside editor, it works great too =)

Every element of the map is customizable inside the editor without touching a single line of code.

The Following documentation is based on the "New! 3D Demo" scene. After reading this document you can safely delete anything inside the "demo" folder and make your own map.

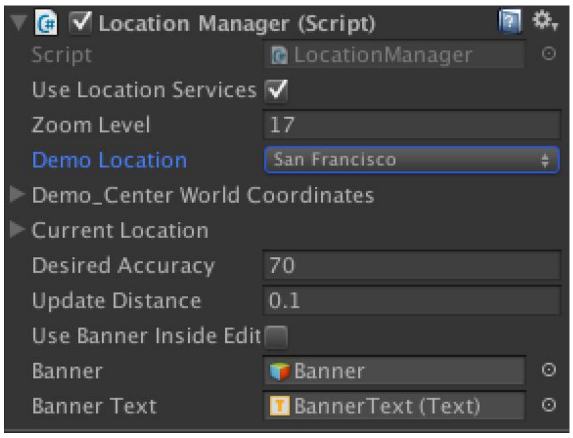
The scene contains 3 game objects:

- Location Manager
- Map

- Avatar

Location manager handles the GPS positioning (live or demo) and Map is the 3D map factory. Avatar is a game-like setup (character, clouds, gestures, land, etc) and it's completely (and recommended) editable / replaceable.

Location Manager
LocationManager.cs
Using the unity Input.Location class handles the gps positioning of the user's device, converting the GPS coordinates in unity world coordinates (2D x,z). It uses events and delegation to notify the map that the users has moved to refresh the map tiles.
You can simulate GPS movement while playing inside the editor using W,A,S,D keys.



Location Manager (Script)

Script	LocationManager
Use Location Services	✓
Zoom Level	17
Demo Location	San Francisco
Demo_Center World Coordinates	
Current Location	
Desired Accuracy	70
Update Distance	0.1
Use Banner Inside Edit	
Banner	Banner
Banner Text	BannerText (Text)

Parameters:
- (bool) useLocationServices: toggles between using GPS positioning or fixed location. It is always false when playing inside editor.
- (int) zoomLevel: the default map tiles zoomLevel. Greater is the zoomLevel (0-22) better is the polygons and lines detail.
- (enum) demoLocation: a list of Location presets to use when playing inside editor or without locationServices. If it's set to "custom" then the demo_Center World Coordinates is used (lat, lon, alt).
- (int) desiredAccuracy: the value in meters to accept the location input data.
- (float) updateDistance: the value in meters to consider location changes.

Banner:
The Location Manager object has a canvas in its hierarchy that is just an animated top bar showing the location status.
- (bool) useBannerInsideEditor: animates the canvas while playing inside the editor.
- (Panel) banner: the banner object. (disables animation if set to null)
- (Text) Banner text: the text object. (disables animation if set to null)

Coordinates.cs
As said before you don't have to edit any code to use or customize this map, but in case you would you should put attention to this particular class.
Coordinates.cs is widely used inside the package to wrap GPS coordinates and converting them to and from Vector3 coordinates.
It offers a series of useful methods to work with geographic data, use it if you want to add some Location logic to your app.

This is how you convert a lat lon to vector3 and place objects on map!

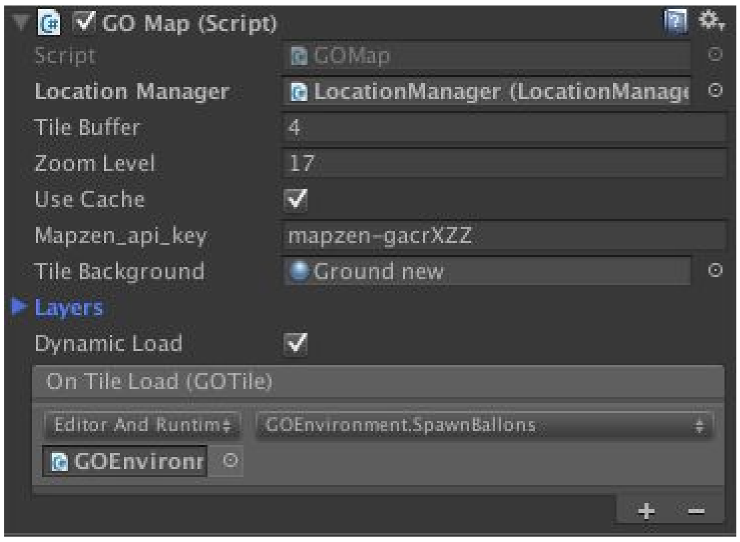
```
Coordinates coordinates = new Coordinates (lat, lng,0);
gameobject.transform.localPosition = coordinates.convertCoordinateToVector(0);
```

Map

GOMap.cs

The Map script holds map settings and customization, it is responsible for the creation of map tiles and their destruction while moving in the real world.

All map data is taken from the MapZen vector maps API, once downloaded is cached on the device and used to create the map geometries.



GO Map (Script)

Script	GOMap
Location Manager	LocationManager (LocationManage
Tile Buffer	4
Zoom Level	17
Use Cache	✓
Mapzen_api_key	mapzen-gacrXZZ
Tile Background	Ground new
Layers	
Dynamic Load	✓

On Tile Load (GOTile)

Editor And Runtim | GOEnvironment.SpawnBallons

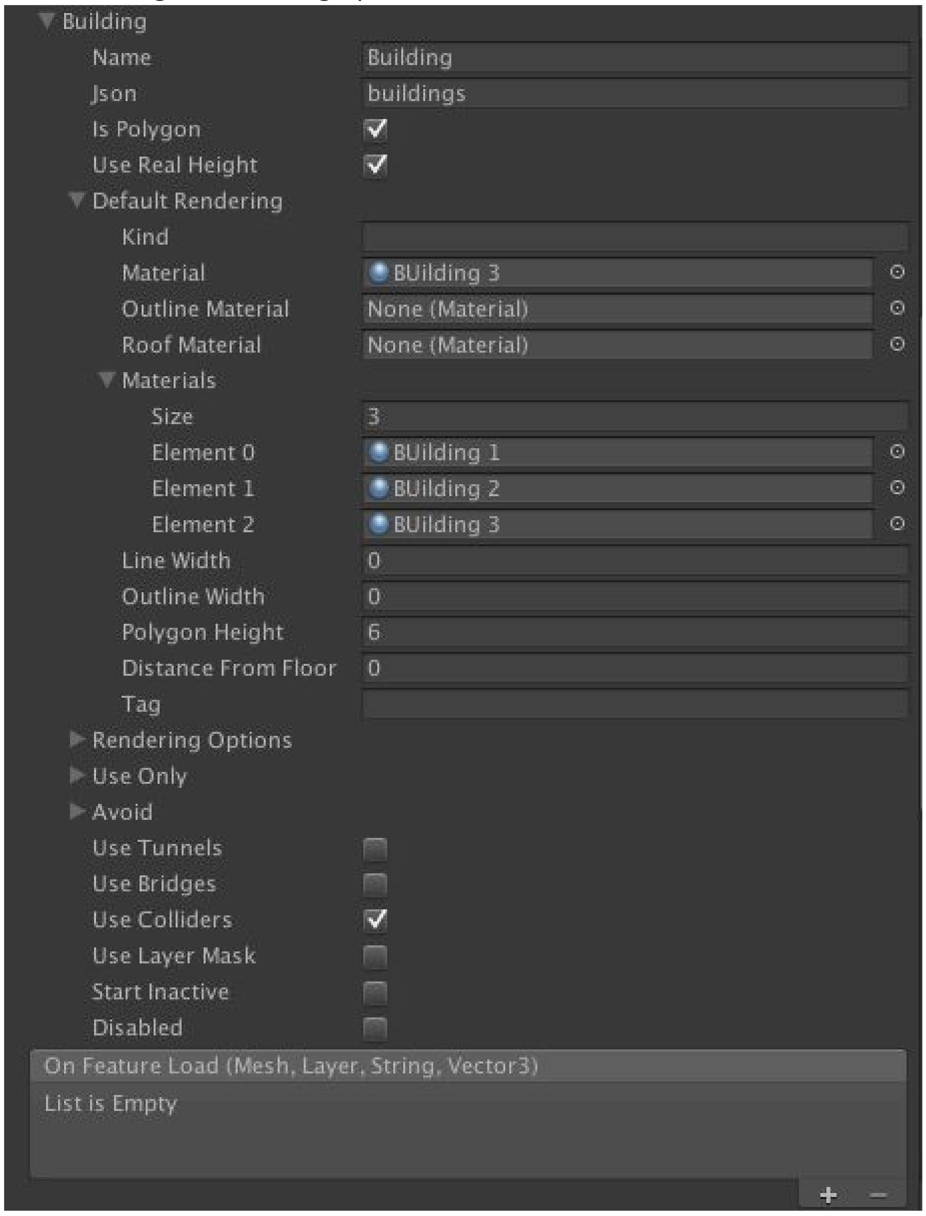
GOEnvironr

Parameters:

- (LocationManager) locationManager: the instance of location manager inside the scene.
- (int) zoomLevel: the default map tiles zoomLevel. Greater is the zoomLevel (0-22) better is the polygons and lines detail. If set to zero the LocationManager zoomLevel value is used.
- (int) TileBuffer: how many tiles the map will buffer around the user' location, this value has to be changed accordingly to the camera depth you want to use and to the zoomLevel value. A tile buffer value of 3 means that map will download 3 tiles in each direction from the current location (49 tiles).
- (bool) delayedFeatureLoad: toggles between creating map geometries all at once or delayed in time.
- (Material) tileBackground: if set the map will create a plane object (floor) with the specified material as a background for each tile.
- (List<Layer>) **Layers**: It is **the list of map layers used in the 3D rendering**. The demo scene uses most of the important map features present in the MapZen api, for the complete list please refer at their documentation: https://mapzen.com/documentation/vector-tiles/layers/
- (bool) dynamicLoad: when set to off the map WILL NOT LOAD. Set to off only if you have already built the map with the in-editor loader.
- (UnityEvent<GOTile>) OnTileLoad: This event is fired whenever a tile is created, you can use it to add some logic after a tile creation. In the scene it's used to target a method to spawn Ballons in the center of some tiles.

Layer (Map.cs)

A layer, "buildings" for example, is a group of objects representing a specific kind of map geometry. The layer class offers a wide range of graphic customization features along with downloading and filtering options.



▼ Building

Name	Building
Json	buildings
Is Polygon	✓
Use Real Height	✓
▼ Default Rendering	
Kind	
Material	BUilding 3
Outline Material	None (Material)
Roof Material	None (Material)
▼ Materials	
Size	3
Element 0	BUilding 1
Element 1	BUilding 2
Element 2	BUilding 3
Line Width	0
Outline Width	0
Polygon Height	6
Distance From Floor	0
Tag	
▶ Rendering Options	
▶ Use Only	
▶ Avoid	
Use Tunnels	☐
Use Bridges	☐
Use Colliders	✓
Use Layer Mask	☐
Start Inactive	☐
Disabled	☐

On Feature Load (Mesh, Layer, String, Vector3)

List is Empty

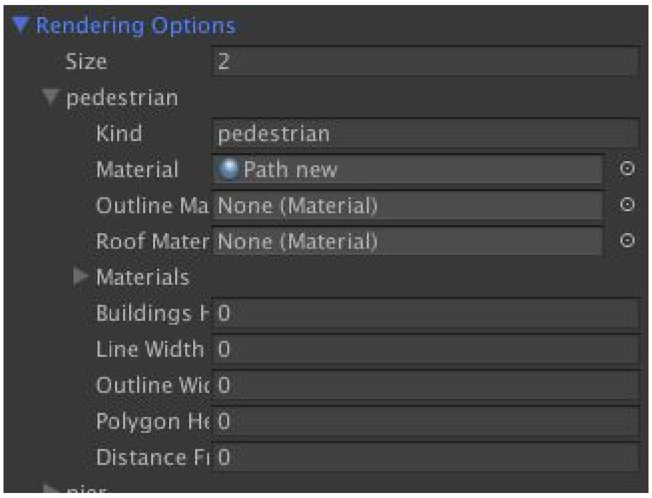
Parameters:

- (string) Name: the name showing inside the editor for the layer.
- (string) Json: the name used by the MapZen API.
- (bool) isPolygon: defines if layer's data is composed by map polygons or lines. (roads are defined as lines with a width).
- (bool) useRealHeight: enables the 3D building feature (polygons only).
- (RenderingOption) defaultRendering: contains all the default rendering options for the layer. Se the next paragraph for the "RenderingOption" class.
- (List<RenderingOption>) renderingOptions: overrides customization for special map elements.
- (string[]) useOnly: use this list to select to render only specified kinds of objects. A list with a size of 0 means that every kind is rendered.

- (string[]) avoid: use this list to select features kind not to render inside the map.
- (bool)useTunnels: flag on to show underground roads. (roads layer only).
- (bool)useBridges: flag on to show bridges. (roads layer only)
- (bool)useColliders: flag on to add colliders on objects.
- (bool)useLayerMask: flag on to add objects to unity layermasks, named after the GoMap layers. Note that you'll have to add layer masks manually to your project in order to make this feature work.
- (bool) Start inactive: the layer is downloaded and crated but set inactive (not visible).
- (bool) Disabled: the layer is not downloaded at all, so it is not created.

RenderingOption (Map.cs)

Layers contains object of various kind that you may want to render differently from the default style. If a rendering option is added the object will use the values specified in it instead of defaults. In the demo scene churches and schools have a different material and every road kind has a different width. While doing this kind of customization is strongly recommended to look at: https://mapzen.com/documentation/vector-tiles/layers/ for a full list of map "kinds".


Rendering Options
Size 2
pedestrian
Kind pedestrian
Material Path new
Outline Ma None (Material)
Roof Mater None (Material)
Materials
Buildings H 0
Line Width 0
Outline Wic 0
Polygon He 0
Distance Fr 0

Parameters:

- (string) Kind: the kind used by the MapZen API.
- (Material) material: the material used to render objects of that kind.
- (Material) outlineMaterial: material used for the roads outline.
- (Material) roofMaterial: the material used to render rooftops.
- (int) lineWidth: width of the lines (lines only).
- (int) outlineWidth: the witdh of the road outline.
- (int) polygonHeight: height of polygons (polygons only).
- (int) distanceFromFloor: the default y value.
- (int) tag: the unity tag used for this kind of objects. Note that you'll have to add tags manually to your project in order to make this feature work.
-

Avatar

MoveAvatar.cs, TwoXOrbitGesture.cs, GOOrbit.cs

As said before the Avatar group is just for demo purposes.

The game object hierarchy contains:

- Land: a plane that's used as main floor for the map.
- Clouds: a fake skybox (sphere) rotating around its center.
- Camera and light: the scene's main camera with a directional light attached.

- Robot: The 3D model used as the moving character.

The two only scripts you would reuse are MoveAvatar.cs (on the avatar gameobject) and the TwoXOrbitGesture.cs (on the main camera).

Move Avatar:
It moves the entire avatar group around the map (floor, clouds, character, camera). It really needs only a location manager instance to work, the (gameobject)AvatarFigure is just to orient the character facing the movement direction.

TwoXOrbitGesture, GOOrbit:
This script, attached to the main camera, provides orbit and zoom camera controls; both working with mouse and touch.
Orbit: moves the camera around the character at a certain distance always facing it. Activate this gesture moving the finger (or mouse) around the character.
Zoom: changes the distance between the camera and the character, as the max distance value is reached the angle between the camera and the character changes accordingly. Activate this gesture with pinch (two fingers) or scroll wheel.

Parameters:
- (gameobject) target: the orbit target.
- (float) distance: starting distance value.
- (float) orbitSpeed: multiplier for orbit gesture.
- (float) pinchSpeed: multiplier for pinch gesture.
- (float) YMin, YMax, DistanceMin, DistanceMax: tweak these values as you prefer to adjust the distance and y ranges.
- zoomCurve: This is only on the GOOrbit script, and it's made to design your own zoom camera movement.

First start guide

This part is going to explain how to create a (simple) map from scratch, tough it's recommended to start by **making a copy of the demo scene** and experimenting with it.

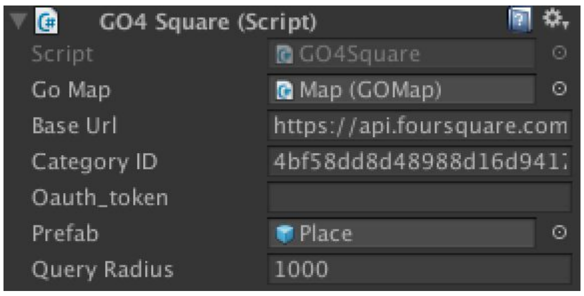
1. Create a new scene.
2. From the top menu GameObject select 3D Object/ GO Map, this will add a location manager and a map to your scene.
3. Play the scene, you will see buildings and roads of the central park area (New York).
4. Start customizing the map object inside the inspector as you prefer, add materials, edit sizes, add new layers, etc.

POI - Demo Scene
(GOPlaces.cs, GO4Square.cs)

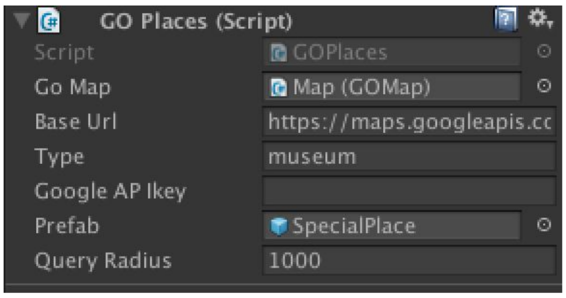
The POI Demo Scene is an example showing a map with some points of interest added. The scene uses Google Places and Foursquare APIs so in order to correctly fetch the POI data you will have to insert a Google Developer API key and/or a Foursquare oauth token.
<u>This scene is intended for demo purposes only,</u> the code in GOPlaces.cs and GO4Square.cs classes is short and well commented so use it to have an idea of how to fetch data and put it onto the map.

If you want to use these classes in your final product please read well the terms and condition of the Google Place and Foursquare services.

https://developers.google.com/places/web-service/intro
https://developer.foursquare.com/docs/venues/search


GO4 Square (Script)

Script	GO4Square
Go Map	Map (GOMap)
Base Url	https://api.foursquare.com
Category ID	4bf58dd8d48988d16d941
Oauth_token	
Prefab	Place
Query Radius	1000


GO Places (Script)

Script	GOPlaces
Go Map	Map (GOMap)
Base Url	https://maps.googleapis.c
Type	museum
Google AP Ikey	
Prefab	SpecialPlace
Query Radius	1000

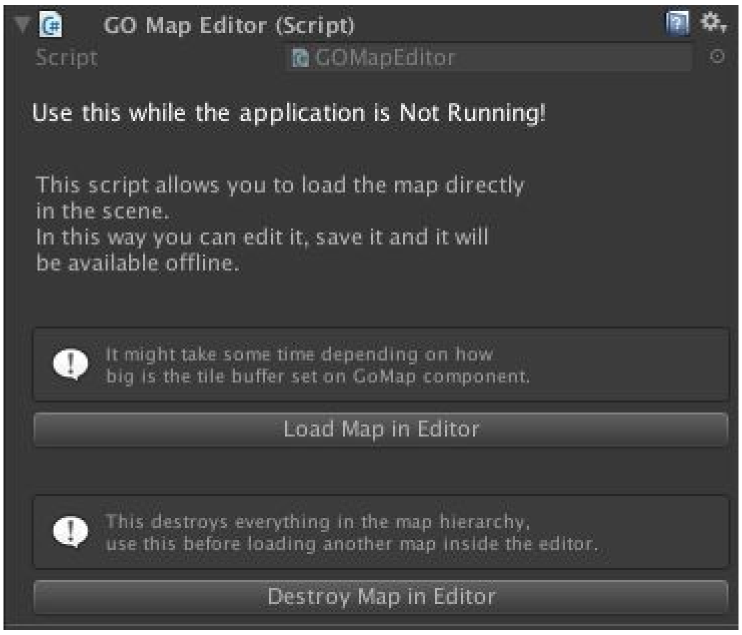
Parameters:

- (GOMap) : the map used in your scene.
- (string) baseUrl: url of the API.
- (string) categoryId: string identifier of the POI category in Foursquare. (https://api.foursquare.com/v2/venues/categories&oauth_token=)
- (string) Type: string identifier of the POI category in Google Places.
- (string) oauth_token: the token you get after a login in Foursquare.
- (string) googleAPIKey: the developer key you get after the Google Developer registration.
- (GameObject) prefab: The prefab you want to use for representing the map data.
- (queryRadius): The radius of the query, where the center is your current location. When the user approaches to the previous query margin the data is loaded again.

In-Editor map load
(GOMapEditor.cs)

With the version 2.0 you can preload the map inside your scene and not use it live with gps. This feature allows you to create awesome real world cities to set your game in.
To load a map in your scene just add this component to an already attached GOMap script and press the "Load Map in Editor" button. To destroy it and create a new one press the Destroy button.
Once the map is loaded you can edit it, save it (as a scene), and place your game in it.

Tip: After you have loaded the map remember to remove the GOMap script or **unflag the "dynamicLoad"** to prevent it reloading the map.


GO Map Editor (Script)

Script GOMapEditor

Use this while the application is Not Running!

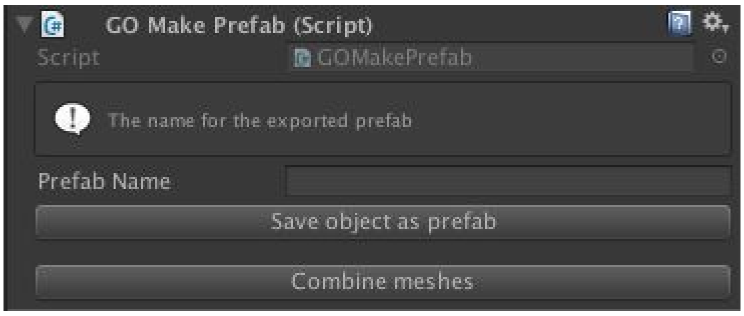
This script allows you to load the map directly
in the scene.
In this way you can edit it, save it and it will
be available offline.

It might take some time depending on how
big is the tile buffer set on GoMap component.

This destroys everything in the map hierarchy,
use this before loading another map inside the editor.

Combine and save meshes
(GOMakePrefab.cs)

With the version 2.0 you can combine and save meshes (like buildings) as prefabs. This could be very useful if you want to save some real building and use it in other contexts.
If you want to save a mesh that is already loaded attach the GOMakePrefab script to its GameObject **<u>give it a name</u>** and press the save button.
If you want to save a group of meshes (like a complex building) attach the GOMakePrefab script to a container GameObject insert a name and press save.
A new directory is automatically created when saving the first mesh at:
GO Map - 3D Map For AR Gaming/Exported


GO Make Prefab (Script)

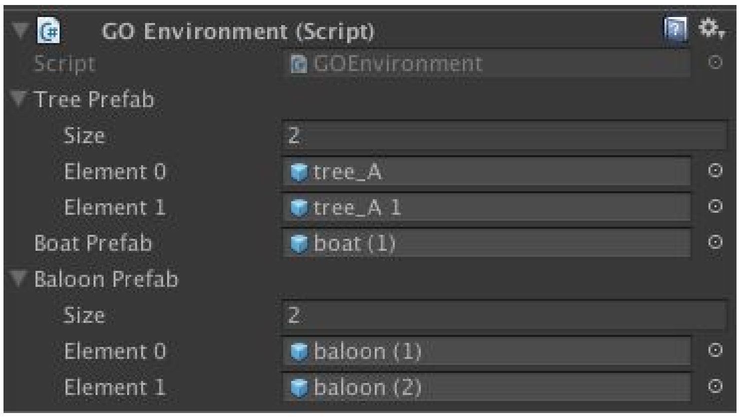
Script GOMakePrefab

The name for the exported prefab

Prefab Name

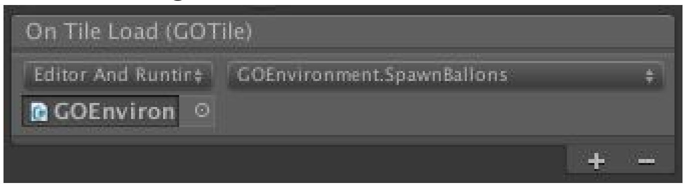
Environment, how to use map events
(GOLayer , GOMap.cs , GOEnvironment.cs)

GOMap generates two kinds of events that you can use to add some other logic to the map creation. The GOEnvironment script is just a way to use them, use it as inspiration for build other stuff or do your custom game logic.


GO Environment (Script)

Script	GOEnvironment
Tree Prefab	
Size	2
Element 0	tree_A
Element 1	tree_A 1
Boat Prefab	boat (1)
Baloon Prefab	
Size	2
Element 0	baloon (1)
Element 1	baloon (2)

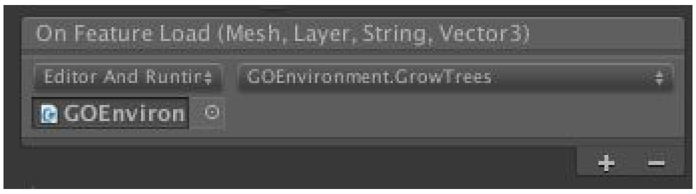
OnTileLoad
This event is fired every time a tile is created. A GOTile parameter is passed in input so you can know everything about the area that is being created.
In the GOEnvironment example this event is used to spawn a Balloon in some of the tile centers at a random height.


On Tile Load (GOTile)
Editor And Runtime — GOEnvironment.SpawnBallons
GOEnviron

OnFeatureLoad
This event is available for every layer separately and it is fired every time a new map feature is created. A Feature is a map geometry with a mesh representing it.
The parameter passed are:

- The mesh created
- the GOLayer object that fired the event
- the “kind” of the feature (see this for list of object kinds)
- the center of the mesh

In the GOEnvironment example this event is used to add trees in every park or garden and to place boats in some of the water areas.


On Feature Load (Mesh, Layer, String, Vector3)
Editor And Runtime — GOEnvironment.GrowTrees
GOEnviron

FAQ

- **There’s an error on FileHandler.cs, can’t compile**: Unity is set to build on WebPlayer which is not supported. Build for any supported platform (iOS, Android, Standalone) and it will fix it.

- **How can I add a GameObject at some GPS position on the map?**: It's very easy, read the POI section here in the documentation and try the "POI - Demo Scene" to have an example or just use the "dropPin(double lat, double lng, GameObject go)" method in GOMap class.

  The basic GPS-Vector3 conversion is just this.

  ```
  Coordinates coordinates = new Coordinates (lat, lng,0);
  gameobject.transform.localPosition = coordinates.convertCoordinateToVector(0);
  ```

- **The map seems to have stopped loading data, why?**: It's probably because you have cached some tiles with a different layer content than the one you're using right now. Try disabling the cache (useCache bool on GOMap.cs) and run again, then put the cache back