mapnav V.1.2 Geolocation Toolkit

Thank you for purchasing MapNav! We have made a great effort to provide you with a powerful and easy to use geolocation engine and maps implementation in Unity, including GPS navigation, 2D/3D content integration (anchoring) and map/camera controls.
This manual will guide you through each and every step necessary to start your maps-based app or game. Please also have a look to the included Demo Scenes where you can find working examples of basic MapNav configurations and check the inline comments in our scripts for further information.

TABLE OF CONTENTS

GETTING STARTED

To start using MapNav you will first need to introduce two basic objects in your scene: a **map** (usually a plane containing the mapnav.js script) and a **player** (user visualization). In the first case, we highly recommend using the "Map" object included in the MapNav's prefabs folder (/MAPNAV/Prefabs). This map object is ready to use and includes all relevant scripts attached. As a player visualization you can also use one of the prefabs provided (2D and 3D pointer), but if you would like to use your own (2d pointer, 3d model, animated character, etc.) this is as easy as setting its Tag property to "*Player*". MapNav will then automatically recognize it as the main actor and set all the navigation system around it.
Note that the provided map object uses a diffuse shader for its main material, so any downloaded texture might appear considerably dark unless some kind of light source is being used in your scene (e.g. directional light). The reason for this is to enable the rendering of shadows from 3d objects located on the map, but if you are planning a 2D app/game you might want to use the "unlit/texture" shader instead. Finally, you will need to get a valid AppKey from MapQuest (or your maps provider) and use it in the MapNav main inspector (see "Maps AppKey" on next page).

MAPS IMPLEMENTATION

By default, MapNav uses the MapQuest Open Static Maps Service and OpenStreetMap* data to obtain the map tile for your current location. This texture is loaded into the Map object which is automatically scaled according to the tile size and zoom level, so world distances remain constant in your scene. Once the texture is not longer needed, it is correctly destroyed to avoid memory leaks.
Note that, when using MapNav, 1 unit in Unity world space equals 100m in the real physical world. (1:100 scale). You should take this scale factor into account, for example, when importing 3d models.
Alternatively, if you would like to use maps from another tiles provider (e.g. Google Static Maps) you can easily do so by modifying the corresponding url for the http call in the mapnav.js script (see inline comments). The new url should be compatible with the MapNav geolocation engine as long as it uses the *center* (lat,lon), *size*(e.g. 640X640) and *zoom* syntax to retrieve images. Also, you might need to populate the maptype array with the new map layers options available from your tiles provider (see the Maptype property below).
Please make sure you read and meet the terms of service and usage limitations of any maps provider you intend to implement. The MapQuest Platform Terms of Use (including Open Services) are available at http://developer.mapquest.com/web/info/terms-of-use.

MAIN INSPECTOR PANEL

The core functionality of our geolocation engine is provided by the MapNav script which is included in the "Map" prefab. If you are using any other object to render the map, please make sure that this script (/MAPNAV/Scripts/MapNav.js) is attached to it.
All the options and parameters can then be accessed from the custom Inspector Panel (see next page figure).

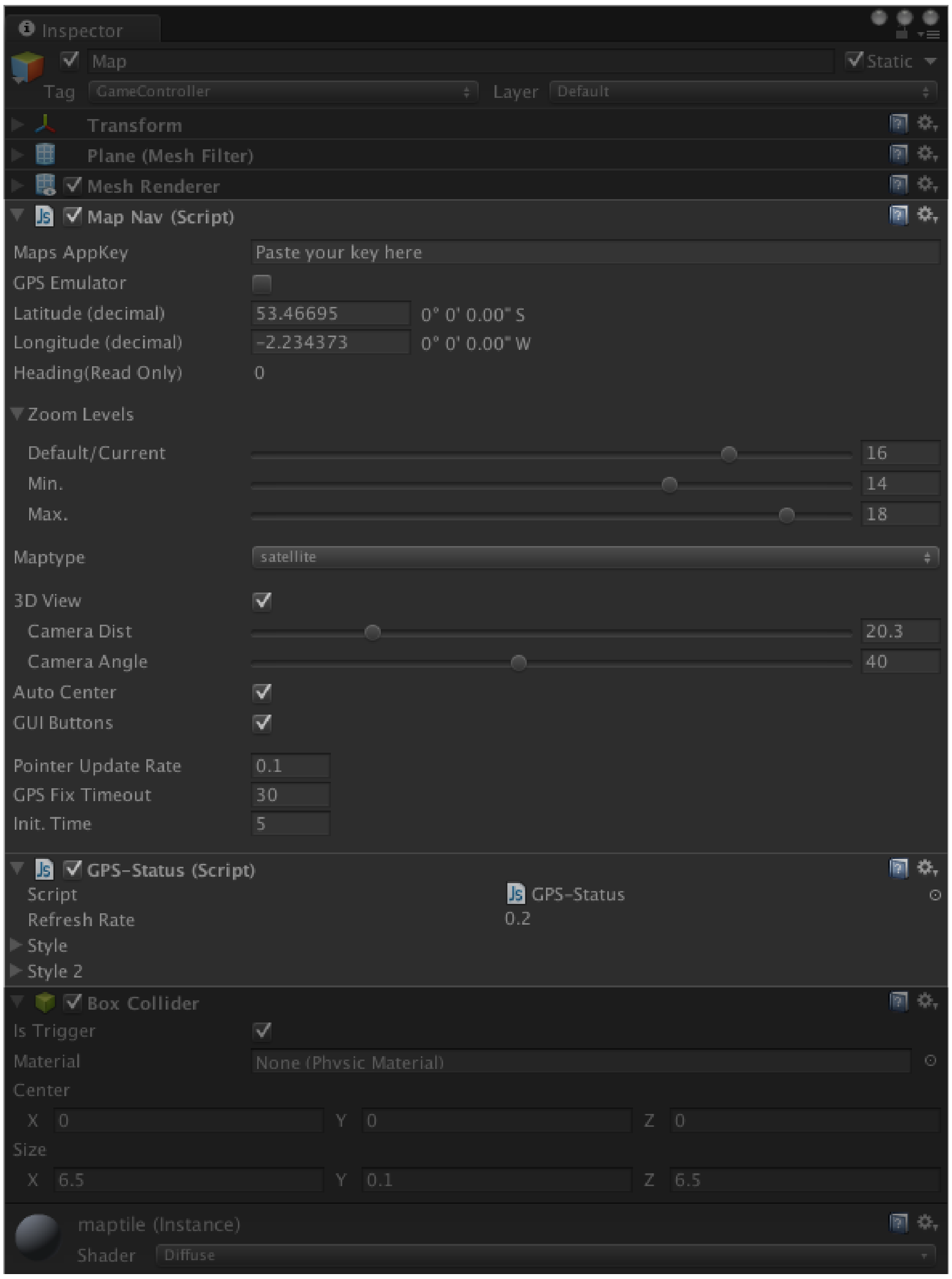
Maps AppKey. This field allows you to enter an AppKey code (API key) so your application can be identified by the maps servers. This is required in order to use MapQuest Open Static Maps and other maps services (e.g. Google Maps API).
To create a new MapQuest AppKey go to:
http://developer.mapquest.com/web/info/account/app-keys

GPS Emulator. Check this box if you are working on a PC/Mac standalone build, so a GPS Emulator is enabled to simulate location tracking. If building for mobile devices (Android/iOS) please disable this option just before you build your project. The real GPS sensor in your device will then be used.

Latitude/Longitude. Reads the current user position as geographical coordinates (WGS84). When using the GPS Emulator you can provide initial Latitude and Longitude values to set the location where your game/application will start.

Zoom Levels. Use these controls to set the default, min. and max. operative zoom levels for your application. Note that, even in some maps providers accept zoom levels up to 20, this level of detail might not be available in all geographical areas.

Maptype. This drop-down list includes all the possible map types that MapQuest Open Static Maps currently offers, i.e. *map, sat* and *hyb*. Choose the one that better suits your application. If you are using a different maps provider other than MapQuest, you can modify this list by editing the *maptype* (array) variable in the MapNav.js and MapNavInspector.cs scripts.



Inspector

✓ Map ✓ Static

Tag GameController Layer Default

Transform

Plane (Mesh Filter)

✓ Mesh Renderer

✓ Map Nav (Script)

Maps AppKey	Paste your key here
GPS Emulator	
Latitude (decimal)	53.46695 0° 0' 0.00" S
Longitude (decimal)	-2.234373 0° 0' 0.00" W
Heading(Read Only)	0
Zoom Levels	
Default/Current	16
Min.	14
Max.	18
Maptype	satellite
3D View	✓
Camera Dist	20.3
Camera Angle	40
Auto Center	✓
GUI Buttons	✓
Pointer Update Rate	0.1
GPS Fix Timeout	30
Init. Time	5

✓ GPS-Status (Script)

Script	GPS-Status
Refresh Rate	0.2
Style	
Style 2	

✓ Box Collider

Is Trigger	✓
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Size	X 6.5 Y 0.1 Z 6.5

maptile (Instance)

Shader Diffuse

3D View. Enable this option if you are planning to add 3D objects on your map, so the camera can be adjusted (*distance* and *angle*) to allow a three-dimensional perspective view. If disabled (2D view), only the *height* of the camera can be set. In both cases, distance or height are measured from the object tagged as "*Player*". Please make sure that your camera is also tagged as "*MainCamera*".

Fixed Pointer Aspect. Fixes the scale of the "*Player*" object no matter what the zoom level is (2D mode only).

Auto Center. When enabled, the map texture will automatically be updated and the camera centered on the new player's position if the following conditions are met:

- 2D mode: The player moves away and is no longer visible on the screen (uses WorldToViewportPoint method)
- 3D mode: The player exits the box collider attached to the Map tile object. The size of the collider might be changed to set a different auto center boundary.

GUI Buttons. Activates a sample graphic user interface to control the following parameters: *maptype, zoom out, zoom in, refresh* (update map), and *info*. The info button enables a GPS status display showing geo-location data and console messages. This information will only be available if the "GPS Status script" (/MAPNAV/Scripts/GPS-Status.js) is also attached to the Map object. This GUI is only provided as a simple functional example. Please feel free to modify or extend it according to your own needs.

Pointer Update Rate (mobile only). This value sets how often the player's position will be updated using the location data provided by the GPS sensor.

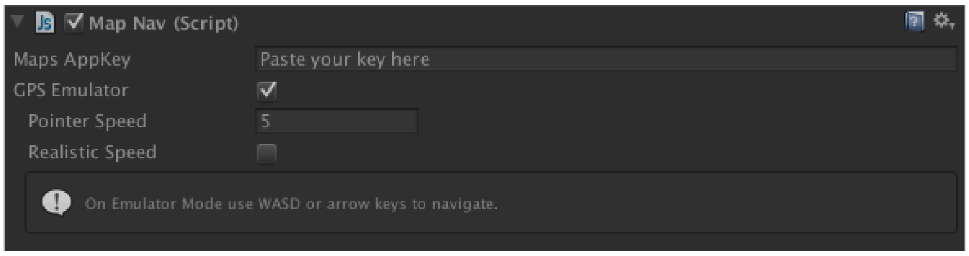
GPS Fix Timeout (mobile only). Maximum time in seconds before the user gets a warning to check if location services are enabled. Allow at least 15-20 seconds so the GPS sensor (if enabled) has enough time to get a GPS fix (it might take longer on some devices).

Init. Time (optional). A initialization time (delay) after a GPS fix has occurred and before the map is displayed, in order to increase the accuracy of the location data (number of satellites).

USING THE GPS EMULATOR

The GPS emulator allows us to implement the MAPNAV navigation system even if the target hardware/platform does not feature a GPS sensor (e.g. PC/MAC standalone applications). Therefore, the user position and movement is simulated using the keyboard input (WASD and arrow keys) and location data (latitude, longitude, etc.) is calculated according to our position on the map.

Two new options show up in the inspector when the GPS Emulator is enabled:


Map Nav (Script)

Maps AppKey	Paste your key here
GPS Emulator	✓
Pointer Speed	5
Realistic Speed	

On Emulator Mode use WASD or arrow keys to navigate.

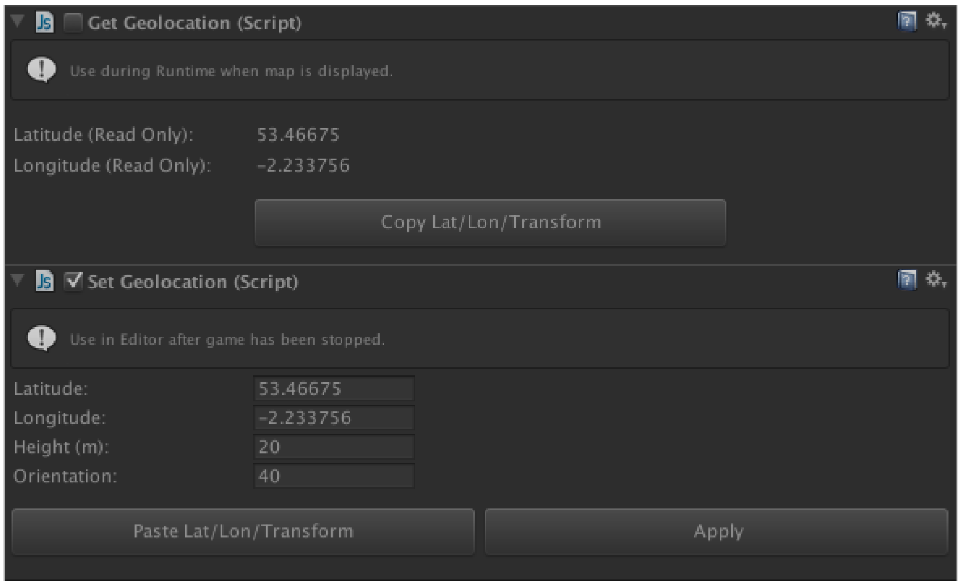
Pointer Speed (Emulator only). This value sets how fast the player moves on the map on user's keyboard input.

Realistic Speed (Emulator only). If disabled, the perceived pointer (player) speed is independent of the selected zoom level (unrealistic behavior). On the other hand, if this option is enabled, the player will seem to move slower on the screen when the zoom level is lower, and faster when we zoom in (realistic behavior).

Even if your target platform is mobile (Android, iOS or Windows Phone) it might be useful to use the GPS Emulator while developing your game or application. In that case, please remember to disable the emulator mode before the final build, so the real hardware GPS sensor is used instead when running the app in your device.

GEO-LOCATING 2D/3D GAME OBJECTS ON THE MAP

In order to populate the map with your own content, you will just need to drop any game objects into your scene and attach the GetGeolocation and SetGeolocation scripts to them.


Get Geolocation (Script)

Use during Runtime when map is displayed.

Latitude (Read Only):	53.46675
Longitude (Read Only):	-2.233756

Set Geolocation (Script)

Use in Editor after game has been stopped.

Latitude:	53.46675
Longitude:	-2.233756
Height (m):	20
Orientation:	40

The GetGeolocation script will allow you to adjust the position (x,y,z), orientation (y-axis only) and scale of your objects on the map while the scene is running, and temporally store these values for later use. This is necessary as the map texture for the current location can only be visualized while the scene is playing and the Unity editor does not keep any changes made during runtime.

After the scene is stopped, you can then use the SetGeolocation script to restore this data.

We can summarize this process in six simple steps:

Step 1. Drag a 2D/3D model into your scene and attach the GetGeolocation and SetGeolocation scripts to it.
Step 2. Start playing the scene.
Step 3. Once the map is visible, adjust the scale, orientation and position of your game object on the map and click on the "Copy Lat/Lon/Transform" button (GetGeolocation inspector). The console will print a message when the data has been successfully copied.
Step 4. Stop playing the scene.
Step 5. Click on the "Paste Lat/Lon/Transform" button in the SetGeolocation inspector and press "Apply" if you want to visualize the new values now while the scene is still stopped.
Step 6. Disable or delete the GetGeolocation script (optional) and make sure the SetGeolocation script is enabled before you play the scene again. Everything should now be in the right place!

Alternatively, if you already know the latitude, longitude and orientation for a particular object, you can also omit the GetGeolocation step and directly insert these values into the SetGeolocation inspector.

TIP: In order to find out the exact coordinates (latitude, longitude) of a particular location in the real world you might find useful the following web service:
http://gmaps-samples-v3.googlecode.com/svn/trunk/geocoder/getlatlng.html

GEO-LOCATING DYNAMIC (ANIMATED) OBJECTS

It is also possible to geolocate non-static game objects using the Get/Set-Geolocation method described in the previous section. An example of this can be found in the 3D Demo Scene included with MapNav (/MAPNAV/Demo Scenes/3D Scene/MapNav_3D_Demo) where the 3D bus model is animated to drive along a particular street. As you can see, the only difference is that we have created an empty **parent** object for this animated object, so the movement of our model is now relative to the position and orientation of its parent's transform. Therefore, the GetGeolocation and SetGeolocation scripts can be used normally on the parent object, ensuring that our animated model is perfectly geolocated within that street.

MapNav - Geolocation Toolkit

For support or further information please contact:
support@recursivearts.com