Navigation Maps (QSF)

Table Of Contents

Information about generating navigations maps for EMERGENCY 5, EMERGENCY 2016 and EMERGENCY 2017.

== AI Navigation Maps ==
For the movement of units from one point on the map to another point QSF uses path or route finding. Navigation maps (also called AI maps or KI maps) are the basis of the path finding routine. All kind of paths and obstacles are assigned to this maps. There are different navigation maps for different units and moved objects. QSF uses the street network for vehicles and persons. The free area is used for other units, for emergency vehicles it's a combination between street network and free area. Helicopter view is used by helicopters, diver map by divers and water streets is used by water vehicles. The bridge map is used for bridges and enables the movement on several layers. There are also special events which need additional navigation maps (e. g. "water during flood" in Hamburg).

Every navigaion map consits of pathes, which have a certain width. Units can move on this pathes. Obstacles will be recognized as areas without any pathes. Only static obstacles are important for the AI maps. You can create obstacles in the editor by placing bullet box collision components or blocker polygones with triangle mesh collision components. More information can by found in the article Virtual Objects. The width of the pathes will be used by the routefinder in order to check, if the unit is able to move on a certain path.

There are two types of navigation maps: Manually created AI maps (e.g. the street network) and automatic AI maps (e.g. the free area). In case of the manually created AI maps the paths assigned in the editor will be transmitted to the system. Paths can be created with the navigation tile tool and can be checked and connected with the street network tool. Attention: Editing failures can lead to wrong pathes, unused streets and driving through obstacles!

Automatic AI maps will be calculated by the editor. Every obstacle which is placed in the area of the AI map will be bypassed by paths. With the help of Voronoi diagramms the paths will be placed between two obstacles. For this step it's important to create all obstacles in a correct way and block unreachable parts of the map. In order to create the AI map it's also important to edit the ground map in a correct way. The ground map contains the height usable for the movements of the units and all important layers (water, height map) for the pathfinding routine. The terrain is always assigned to the ground map, objects will be assigned only of they have a walkable component. The regular height level of the terrain and most of the streets is walkable level 0. Level 1 will be used for bridges.

== Recalculation of navigation maps ==

Navigation maps can be created with the navigation map tool. Every time you change paths, obstacles and collisions you'll have to update the navigation maps. The navigation maps have to be created in the right order (hint: just calculate the navigation maps in the standard setup from buttom to the top).

Depending on the type of the navigation map the maps have to be calculated with the convert street network tool or Cal tool:

- Manually and directly created navigation maps can be transmitted with the concert street network tool, there aren't any settings (except Set). The transmission of the paths takes a few minutes.
- Automatic navigation maps can be created with Cal tool. You'll have to set the right settings. The calculation takes a few minutes.

The following parameters are important:

Corner1 and Corner2

Defines the area where the navigation map will be created. Only in this area paths will be created and obstacles calculated. The map border will also be recognized as obstacle. The height of the area (the y value of the coordinates) will be calculated with the help of the ground map, if TestAtTerrainHeight is activated. In the area between -0,2 and 2 meters every obstacle up to 2 meters above the ground will be assigned as obstacle. The value WalkableLevel sets which layer is used in the ground map.

CellSize

For the calculation with the Voronoi diagramm every obstacle will be assigned to a grid. With CellSize you can define the size of a cell. This value defines the resolution of the obstacles and small details of obstacles get lost.

LaneType

You shouldn't change this predefined values.

MapId

This is the identification number for our navigation map. This value should be set automatically. Attention: The numbers of the most important AI maps are predefined by QSF and shouldn't be changed! All units and the gameplay logics use the predefined AI maps.

AvoidedCollisionFilter and RequiredCollisionFilter
This setting defines, which collisions will be recognized and which not. Sett collision flags for further details.

WalkableLevel

This setting defines the WalkableLevel used by the AI map. If there is a layer above which isn't more than 1 meter apart, the area will be defined as blocker. This setting is used, when you mix several leayers with MixWithMapId. If this setting shouldn't be used, you'll have to set the WalkableLevel to -1.

MixWithMapId

MixWithMapId can mix two different AI maps from different layers, e.g. with a bridge map. This enables the vehicles and persons to walk and drive under bridges and to calculate correct pathes, so the unit won't jump from the bridge.
You'll need additional settings for the mix. You'll have to use special crossings, in order to assign paths for the connection and use bridge components. If you want to deactivate this function, set the value to -1.

IncludeWater
If you want to assign the water too (e.g. the diver map), the value has to be activated. Otherwise the water surfaces will be blocked in the AI map. Attention: The water polygon will still be an obstacle and the collision filter has to be set in a correct way.

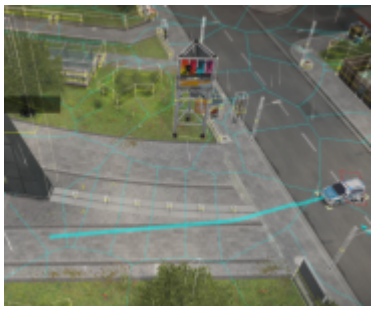
TestAtTerrainHeight
IF activated, the terrain and the walkable surfaces will be recognized for the calculation. Otherise the absolute height value will be used.

Show Obstacles, ShowEquidistantCells, mShowEquidistanceObstacleInfluence and DebugVisualizationTime
This are several debug functions which can be ignored. Show obstacles could be interesting in order to visualize all obstacles assigned to the AI map.

Set with own bar

The street network and the free area will be connected in the game, so emergency vehicles can use both. The value should be set correct (1 for street network, 0 for free area).

After finnishing the calculations you have to save the map. Don't forget this step! New navigation maps can be created and old ones can be deleted, but mostly there are changes in the code too.



The width of the paths won't be shown. By path smooting the path can use the whole width.

1 Bridge editing (difficult)

If bridge and several layers of navigation maps should work in a proper way, you'll have to do several complex editing steps! Every bridge on a higher walkable level must have paths on the junction. Just use a debug box with a street crossing compontent at this position. The paths can be created with the navigation tile tool. All paths have to be exactly between the junction of the walkable objects (e.g. a node on the bridge with WalkableLevel 1 and one on the screet with WalkableLevel 0). The correct number, position and width of the paths isn't easy to set. The path smoothing can make problems at the junction and can create zigzag lines. For this reason it's very important the define the junction at positions, which are equal to the natural and realistic positions of junction (e.g. the walkway). The width of the paths have to be big enough that all unit types can get trough it. All junction have to be covered by paths, otherwise the area will be recognized as obstacle. Without any pathes the bridge won't be reachable for units and the unit will search an ending point near the bridge. The bridge also won't be recognized even if the bridge is the shortest way to reach a destination. Every junction should be tested with persons and vehicles.

The second part of editing is about the creation of bridge components in order to identify bridges. QSF has two different pathfinding systems. The first system works with the help of navigation maps and is used for the

creation of paths. The second system is used in order to bypass obstacles dynamically, this system is completely grid-based. For this reason, the bridges and higher layers with WalkableLevel 1 or highter have to identifiable at any point. By placing debug boxes with bridge components you can give every bridge a certain ID. Every point of a bridge has to be nearer to a bridge compontent of the corresponding bridge than to the bridge component of another bridge. In case of bigger bridges you will need several bridge components to cover every part of the bridge. You also need collisions boxes or polygons underneath every bridge in order to prevent units breaking through with the second pathfinding system (the distance of the obstacle to the bridge has to be at least 0,2 meters (if you value of Corner 1 and Corner2 is between -0,2 and 2 meters), in order to prevent assigning obstacles onto the bridge instead of underneath the bridge.