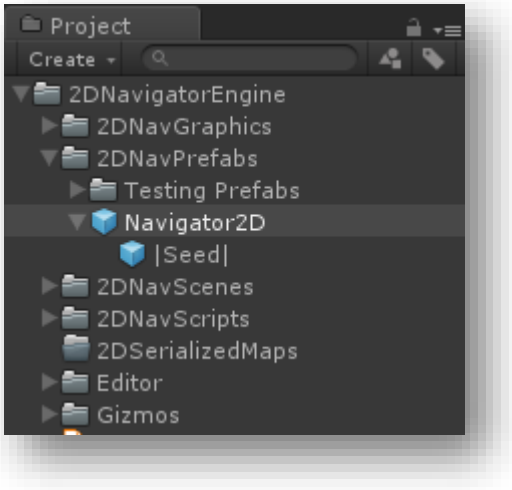
Navigator 2D

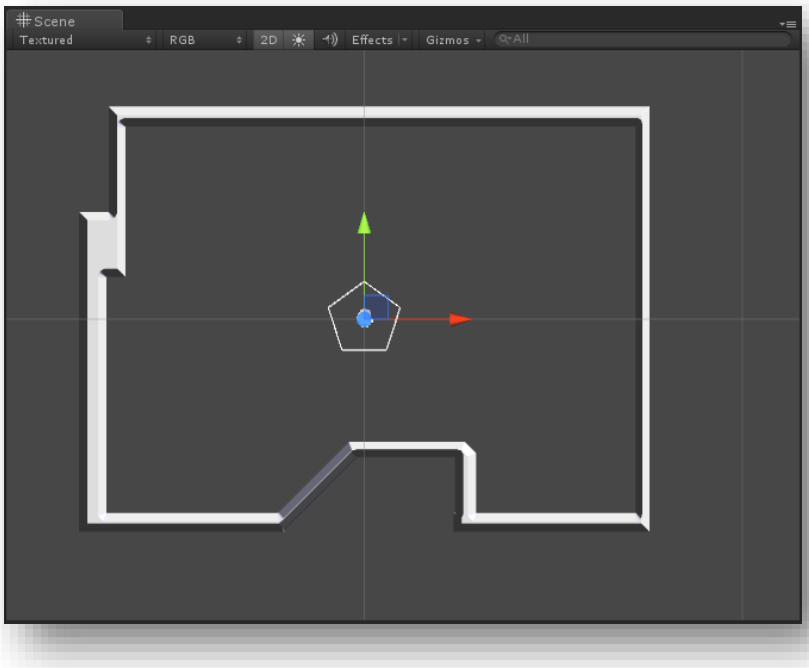
Creating the navigation polygon

Note: **Is mandatory create two new physic layer, one for flood fill, obstacles and zones (Recommended name Tile2D), and another for the navigation area (Recommended name Navigator2D)**

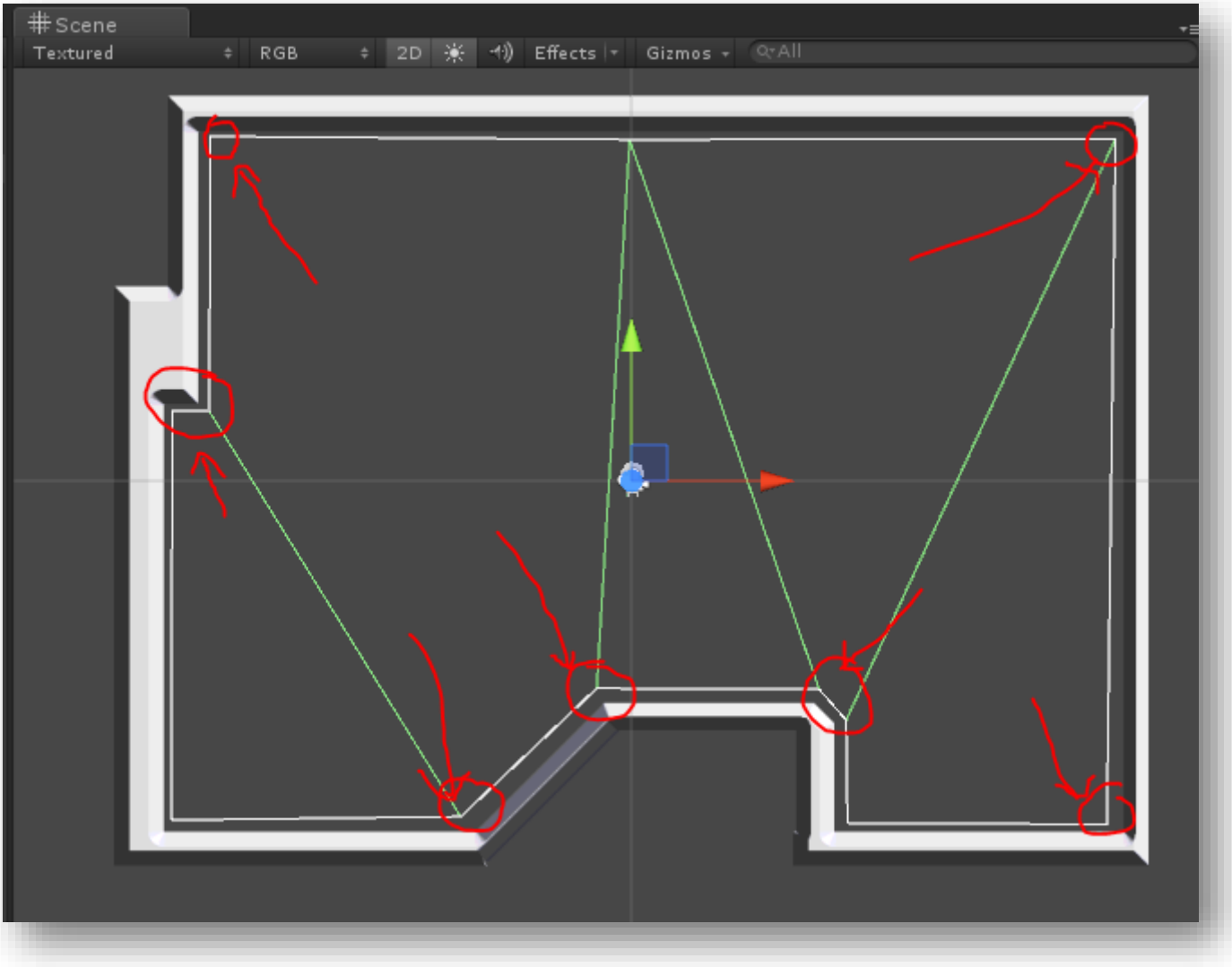
On unity editor, got to the **"KDNavigatorPrefab"** directory inside of you project (After importing navigator2D plugin). Drag the prefab named "Navigator 2D" into your hierarchy window. This prefab has a 2D polygon collider mandatory component attached on the root game object, and the C# script "KDNavigator2D" and one child game object attached named "**|Seed|**" (or seed for be faster). The seed must be always inside of the Navigator2D root game object. If you break this relation path finder 2D system will not work properly. As an advice try to not break the relation with the prefab "**Navigator2D**".



For creating the navigation area (zone of the 2D space that we are going to move using Navigator 2D engine) we must to select the game object on hierarchy named "Navigator2D". After that we will see on the Scene view window a white pentagon placed (generally) on the (0, 0, 0) point of 2D space.



Now, you can hold on **‘shift’** key and move mouse over of any white polygon lines of the pentagon. This is the polygon that is going to define full navigation area. In test scene, we had created a simple white wall to draw our navigation limits. For that we must to move all polygon points to the inside limits of our white wall.



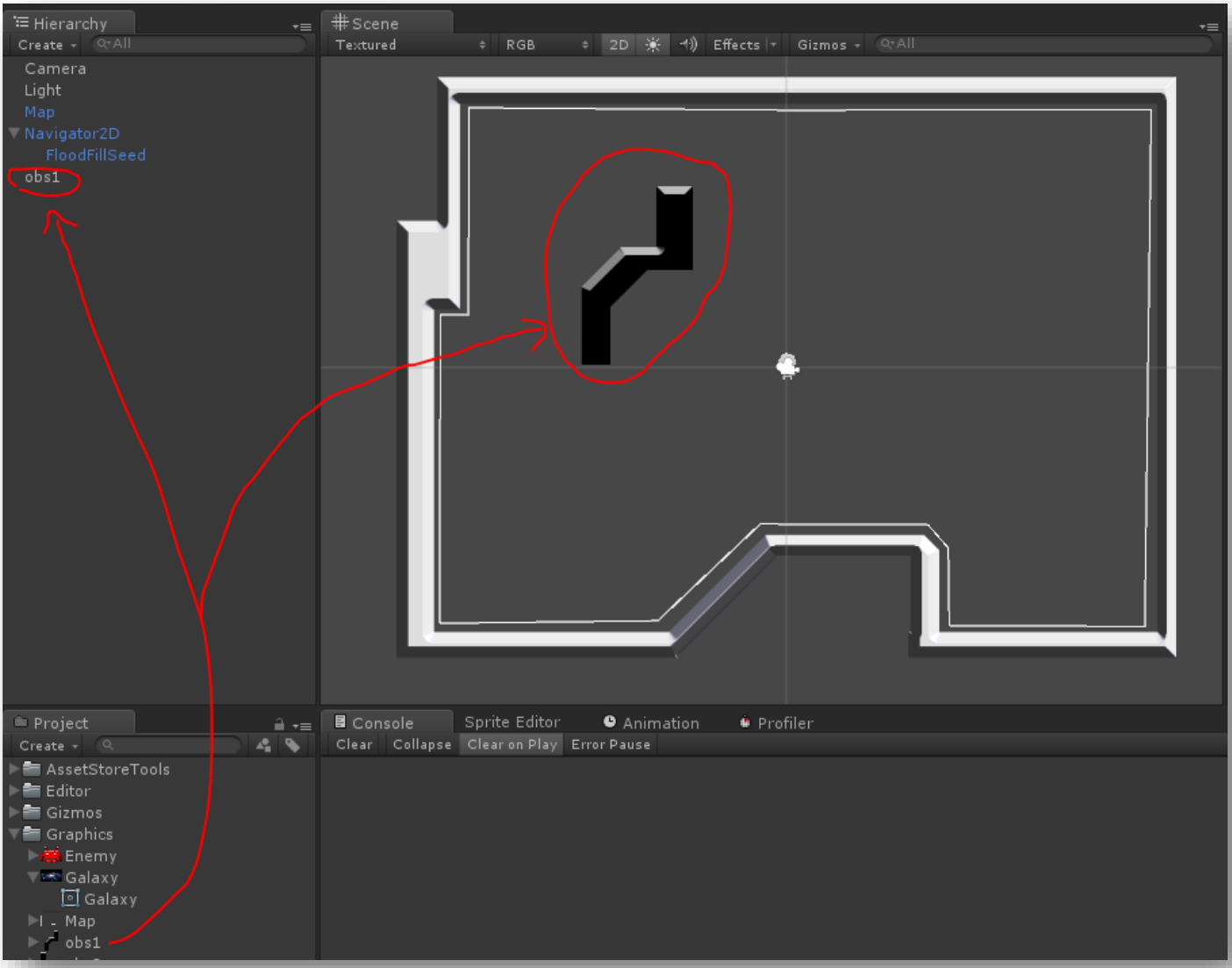
It´s recommended place the polygon limits not exactly over limits of the map or zone sprite. Doing that we ensure best results avoiding corners.

We just now created our navigation area. And our “Navigation Agents” (like in 3D navigation mesh), are going to be allowed to move only inside of this polygon.

Adding Obstacles

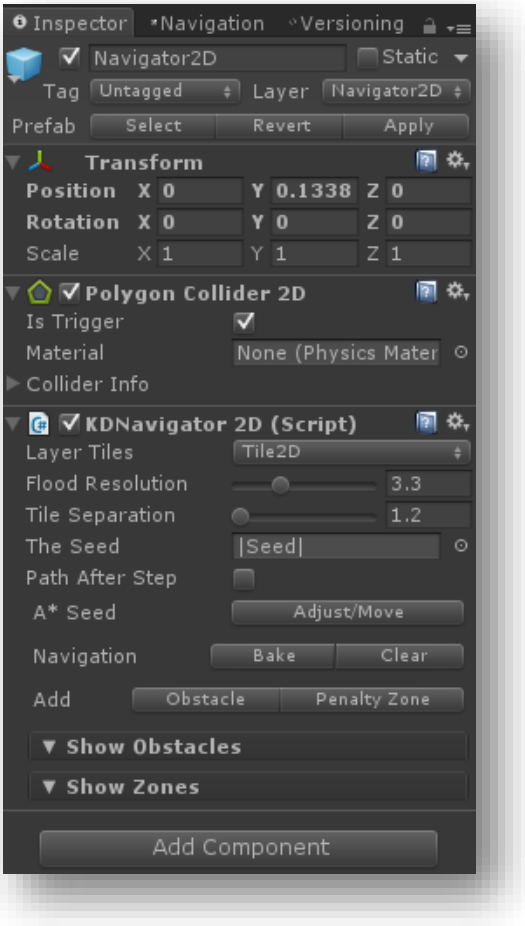
Of course Navigator 2D is a pathfinder engine for find routes from one 2D point to another. For that we must to avoid all obstacles inside of our navigation polygon.

After putting our obstacle inside of the scene (for example in our case, a simple Tetris obstacle)

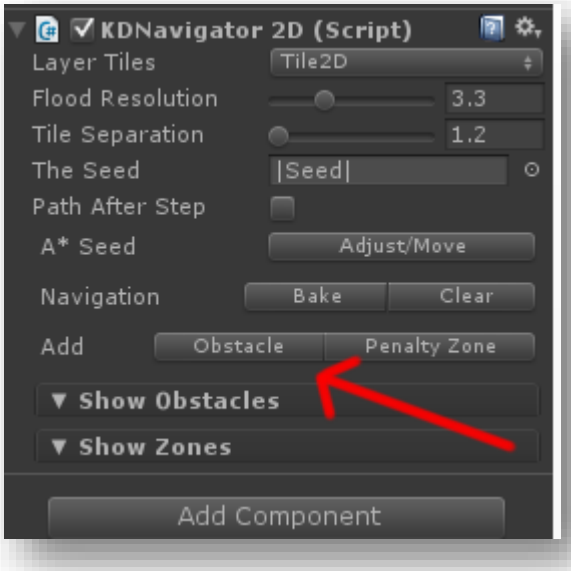


Navigator 2D

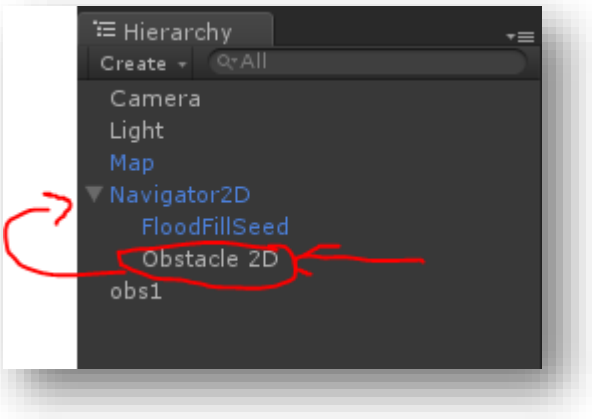
For create an obstacle we must to select **‘Navigator2D’** object. On the inspector window we see this options.



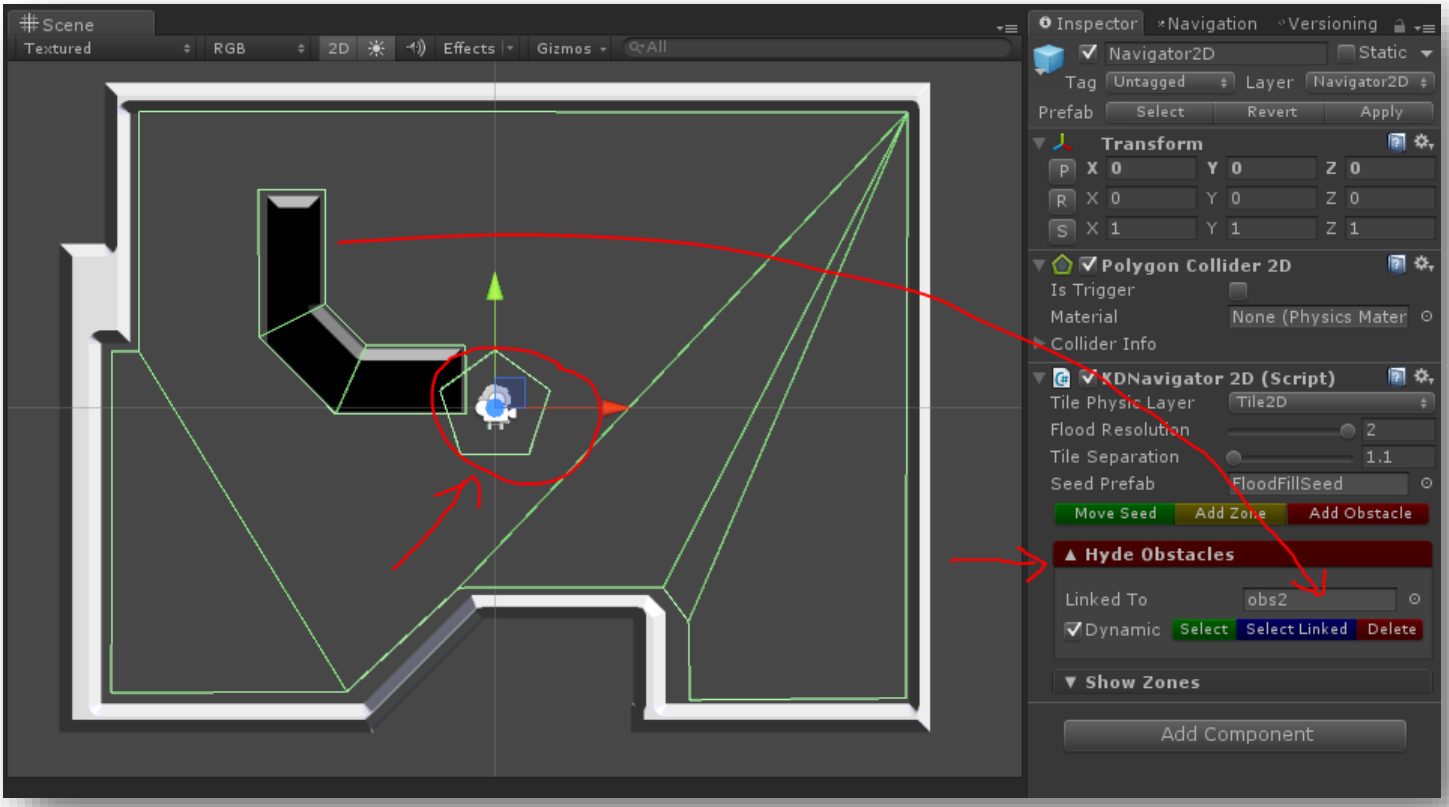
Transform component of **‘Navigator2D’,** polygon collider 2D component (Used as a navigator polygon) and **‘KDNavigator2D’** inspector properties. As we see on the picture we must to push **‘Add-> Obstacle’,** with this action we create a new obstacle inside of our main navigation polygon.



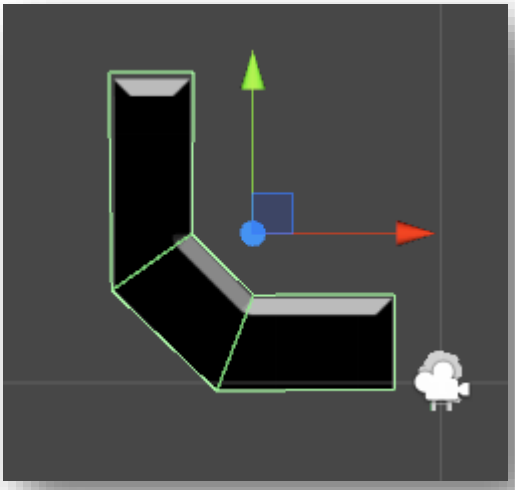
It will instantiate a new game object considered as an obstacle, and automatically it will be attached as child object inside of “Navigator2D” root object.



A “diamond” will be appear on the screen. So we can edit all points of this new obstacle zone that must to envelop the obstacle (for example a 2D Sprite like a rock or tree) that our “Navigation 2D agents” must to avoid.



We can now “link” the polygon collider to the obstacle (in this case the sprite), if we do that we can move the sprite over the map and our polygon collider obstacle will follow the object over the space. Without parent/child relationship. If we marc this obstacle as static, if we do that obstacle will not react with the path engine if we move this sprite over our navigation polygon on playing mode. If this obstacle is not static we can move over our navigation polygon at it will react with navigation agents on playing mode. After that we must to draw correctly all points of our polygon for overlap full area of our obstacle. Set a small free space between the sprite and polygon collider for best results avoiding corners.

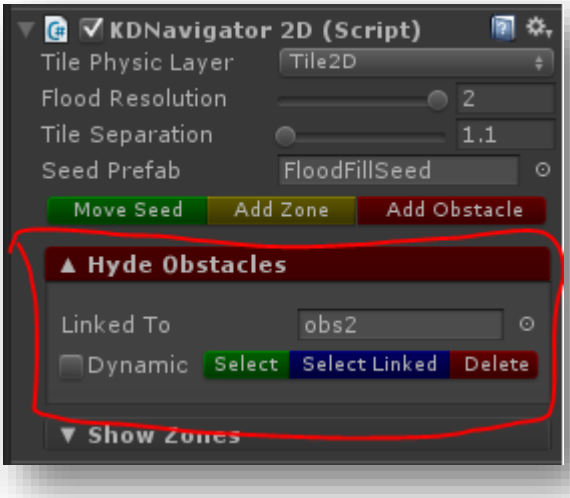


The buttons of the inspector means:

“**Select**” button will select the obstacle collider object (but not the linked object if this linked object exists), use this button for fast modifications of the polygon area, or set other values of the obstacle.

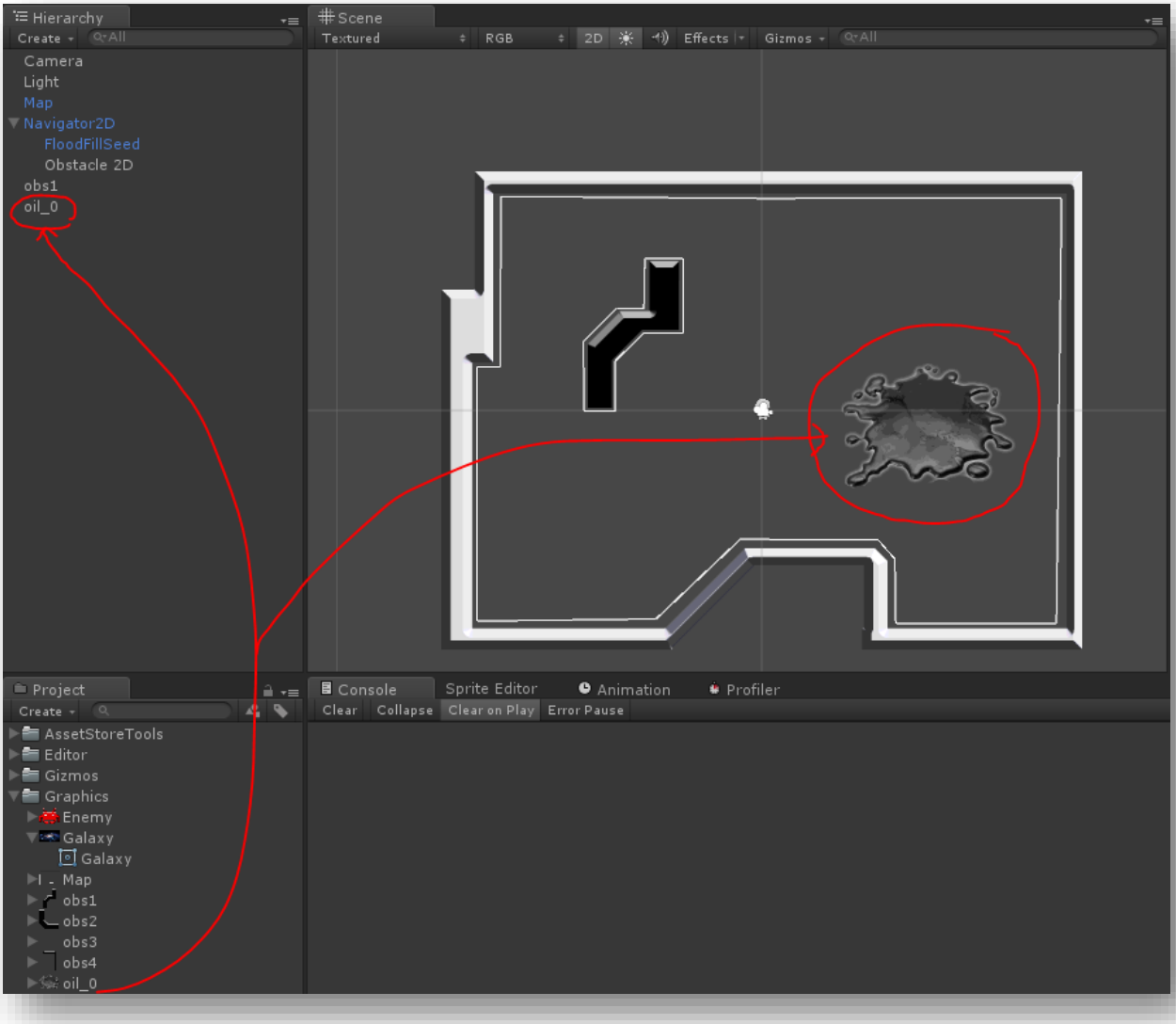
“**Select Linked**” will select the linked game object, with this option we can move the linked sprite over the map and our obstacle component will follow the sprite 2D position in the space

“**Delete**” will delete the obstacle (the Polygon collider with KD2DObstacle component but not the linked object)



Adding Zones

The concept of zone in Navigator 2D engine is very simple, a zone represents a part of space wen can walk but is not the best option to go across. For example a part of the map full of oil, or a part of the forest that has quicksand for example. A way from start to destination can be longest but we can avoid this “dangerous” zones if we have the option of do that. But if there is possibility to avoid the zone, we can use to build the path between our position and destination point. Zones has a penalty values, it means that we can configure zones with more or less impact to the way that we build paths. A zone can has a penalty of 10% or 300%, it depends on what kind of zone we want it will be.

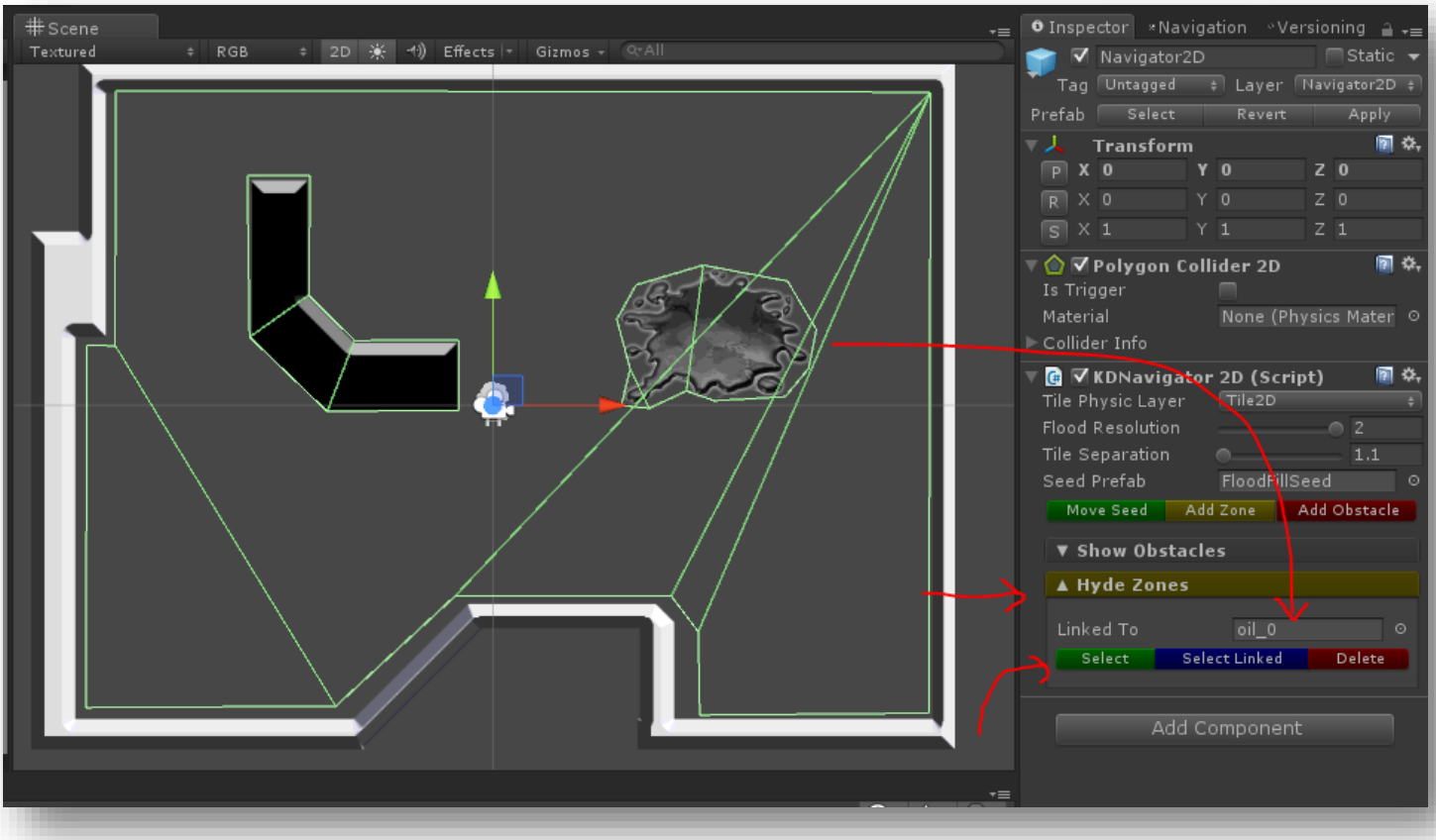


Navigator 2D

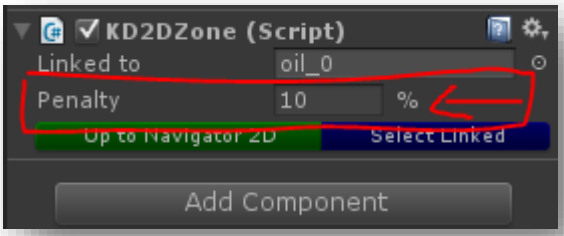
Got to Navigator 2D object in your scene hierarchy and push on the inspector “Add Zone” button. Like creating obstacle this button will add zones to a general area list (including obstacles and zones). As obstacles, zones can be linked to a sprite object (or not, is not mandatory, but it´s recommended). Zones are considered always dynamic.

For example if this oil sport is moving inside the zone, and it´s marked as dynamic, it will be affect the path engine, increasing the penalty of the points inside.

Equal than obstacles a pentagon appears on the editor screen and we can link the sprite that we are going to overlap with our new polygon collider.



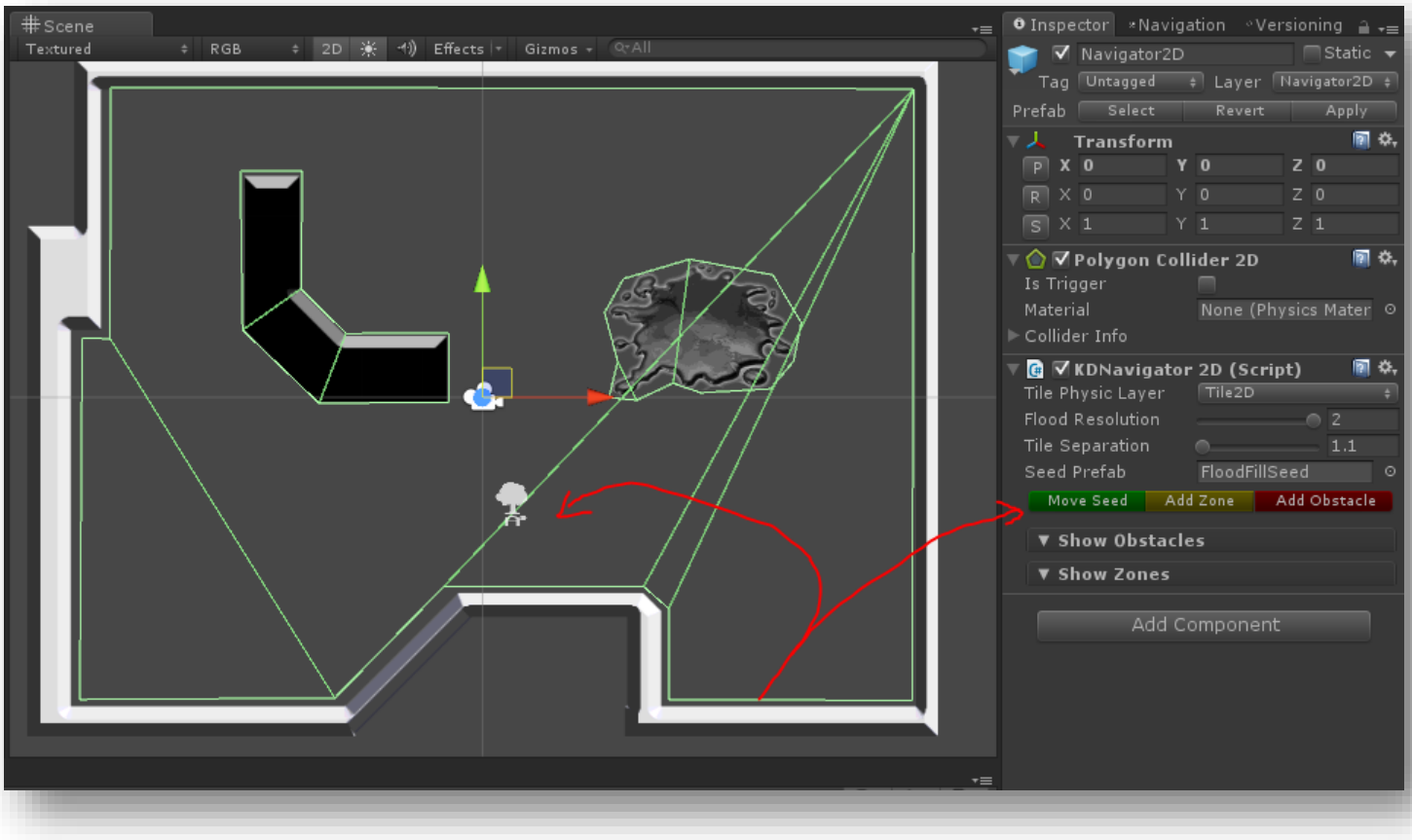
If we push on “Select” button, we can edit the penalty values of this zone. This value will increase or decrease the penalty. 0 of minimum, infinite maximum. All zones are dynamic by default.



Navigator 2D

Flood Fill & Flood Fill Seed

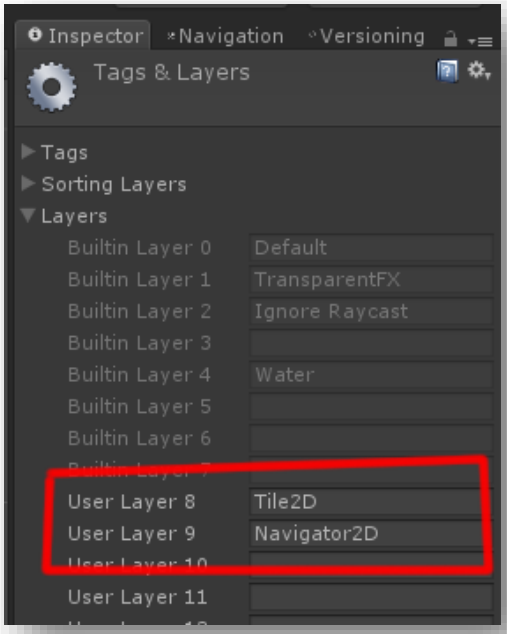
You can see inside of the polygon navigation area the small white tree gizmo with the A* write at the bottom of the icon. This gizmo represents the position of the flood fill seed. In other words, for move over the navigation polygon we are going to flood its area of small tiles, like a chessboard. Position of the A* seed (the white tree), is the point that we are going to start to fill the area of tiles. Move the seed over this are for find the best results on drawing the chessboard for move inside of the navigation polygon2D.



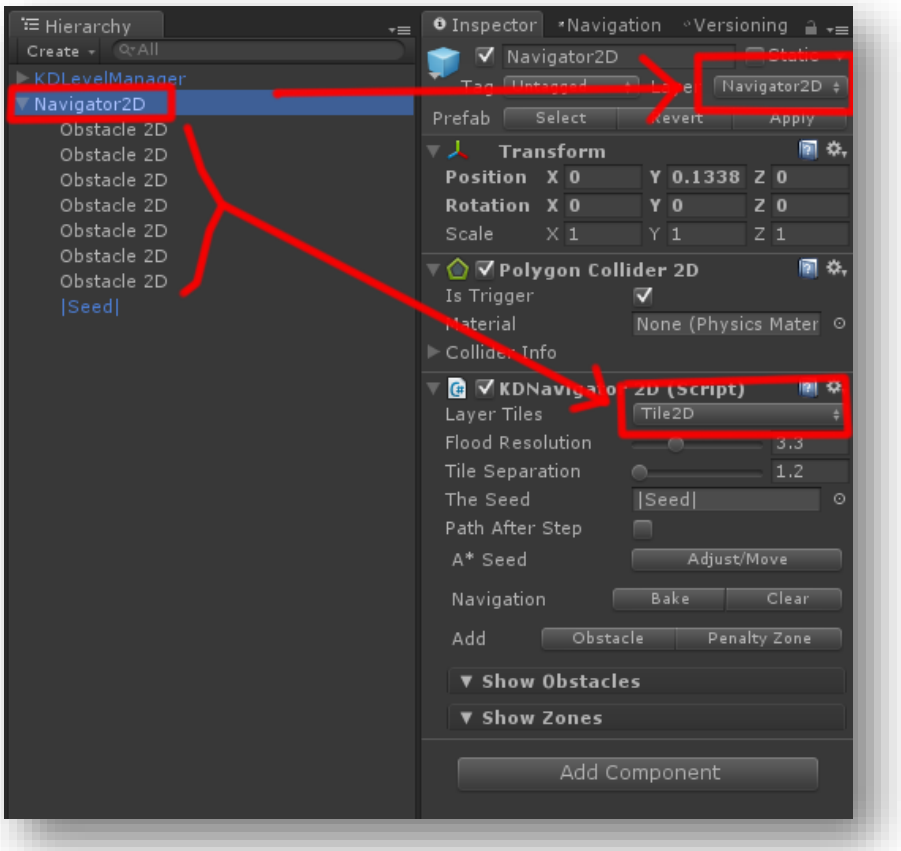
After configure the obstacles and zones (if we have inside of our navigation polygon) you will see if you enter on play mode the chessboard unfolded over the 2D surface.

Navigator 2D

Is important create two new physic layer, one for flood fill, obstacles and zones (Recommended name Tile2D), and another for the navigation area (Recommended name Navigator2D), for good working of the plugin navigation polygon (root navigation object) and tiles must be on different layers.



Remember create two new layers and put the “Navigator2D” game object on one layer and then select KDNavigator2D inspector and set a different physic layer for Tiles.



Remember that Tile2D physic layer is for obstacles, zones and "**|Seed|**" game object, and Navigator2D physic layer is for Navigator2D root object.

Baking the navigation area

You must to pay attention to **"BAKE"** the navigation 2D area. Before you can use the tiles to find path, you must to push "bake" button to see the results of doing flood-fill with the resolution and the separation settled before. A small number of tiles represents best performance, but not accurate paths. A big number of tiles, represents bad performance but best accurate path. Decide what you need. And "play" with flood-fill configuration parameters for best results for find path as fast as the engine can. **"Clear"** button will break all Tiles clear navigation area.

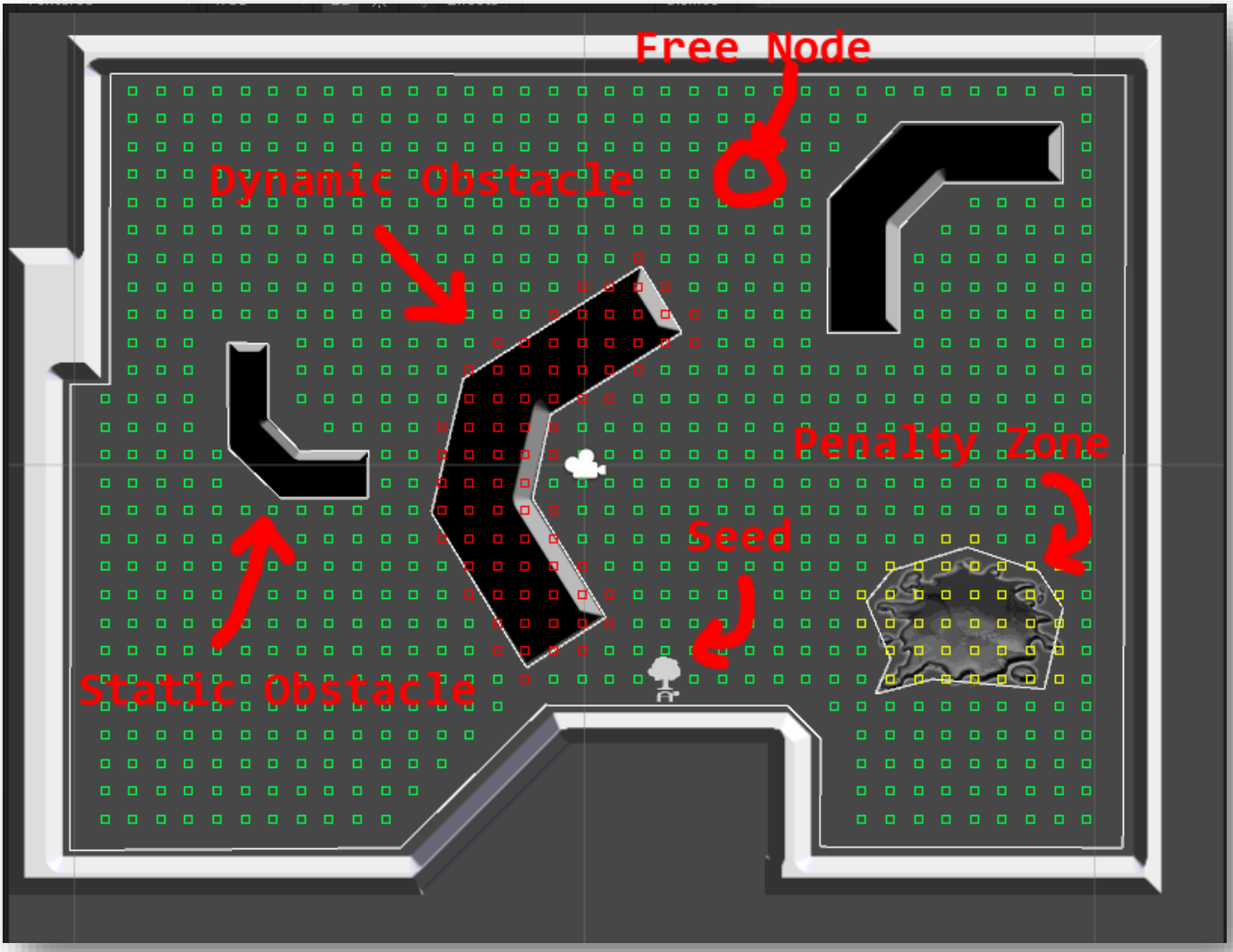
As we see on the next picture, the seed has flooded all the area with small tiles.

This tiles has four states. Green states, normal tile and completely walkable without penalty. Yellow state, represents a tile with penalty (all tiles inside of penalty area).

Red state, this tile is not walkable, but you can walk over if the object moves of this place. Red tiles are all tiles inside of an obstacle configured as “Dynamic obstacle”.

Dynamic obstacle will affect the routes of the agents if they enter inside of a rendered path between start and destination point.

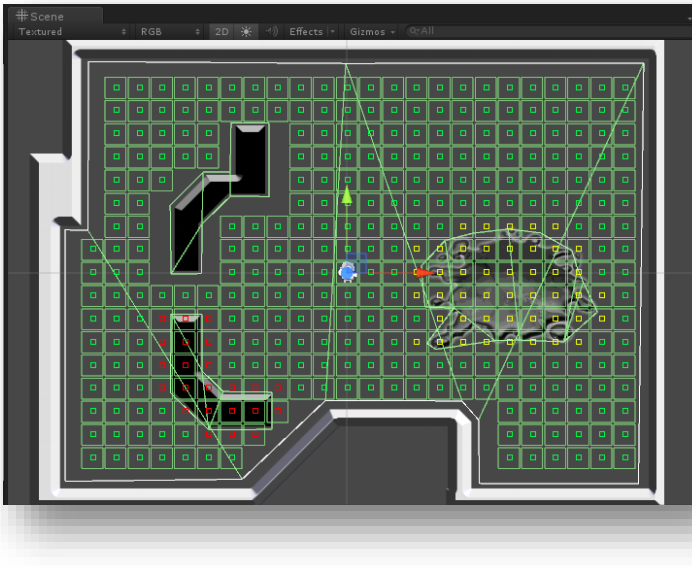
The fourth state is dead tile. If the obstacle is configured as not dynamic, the path engine will consider that is never going to move later, so is ignored on the chessboard for performance reasons.

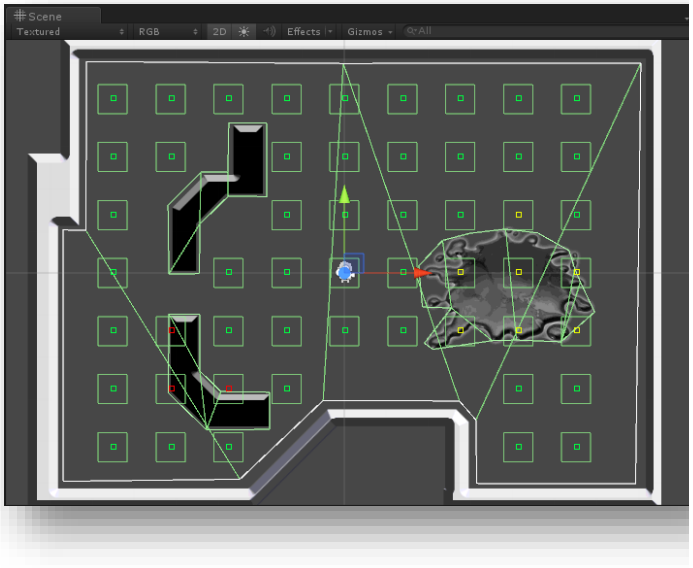


Flood Fill resolution and Tile Separation

We can change the flood resolution and increase and decrease the separation of tiles inside of the navigation polygon.

It will affect at the performance, highest resolutions and lowest separation factors has a lot of cost, and lowest resolutions and highest separation factors increase the performance but are not very accurate on path calculation. Do test by yourself and find the best solution for every map.





Navigator 2D

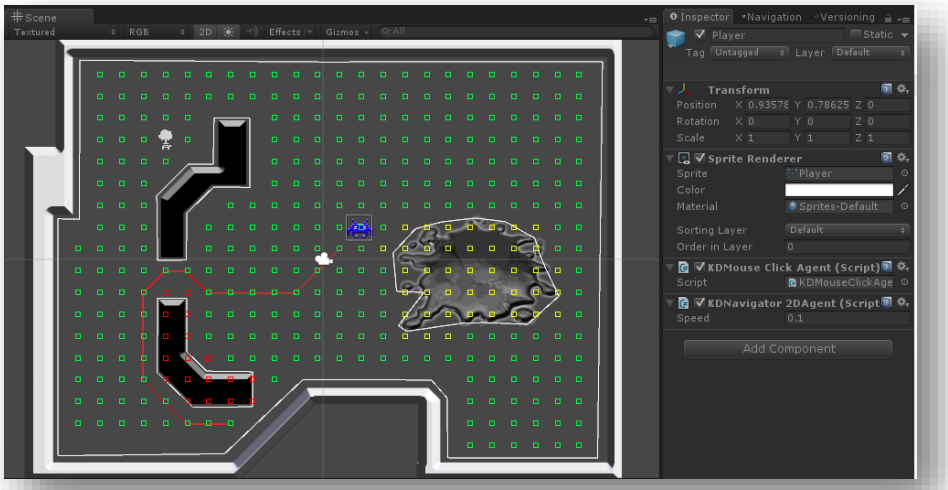
Navigation Agent

The navigation agent (‘**KDNavigator2DAgent**’ script) allows to walk automatically inside the navigation 2D polygon. With this way you can call the method ‘**Set Destination**’ and pass as a parameter the position on the 2D plane that you want to set as destination of this navigation agent.

```
#region "Properties"
/// <summary>
/// Navigation Agent to set destination of this object
/// </summary>
public KDNavigator2DAgent Agent { get { return _agent ?? (_agent = GetComponent<KDNavigator2DAgent>()); } }
#endregion //Properties

#region "Monobehaviour"
protected void Update()
{
    if (Input.GetMouseButtonDown(0) && Camera.main)
    {
        var mouseSpace = new Vector3(Input.mousePosition.x, Input.mousePosition.y, Camera.main.transform.position.z);
        Vector2 planePosition = Camera.main.ScreenToWorldPoint(mouseSpace);
        Agent.SetDestination(planePosition);
    }
}
#endregion //Monobehaviour
```

Remember that the agent will react dynamically at the invasion of the routes by dynamic obstacle, and it will render another route at the moment that the obstacle entered on the route. You can additionally select “Path after step” for more accuracy on real time path calculation, but it will affect to the performance of the game. Be careful with this option, and evaluate when you can activate or not.



Remember that “**KDNavigator**” engine is designed and optimized to work under 2D mode of Unity. And it will operate using “x” and “y” coordinates and physics 2D engine.

KD Productions Spain 2014 (Barcelona)

kd.interactive.barcelona@gmail.com

http://www.group-kd.com/companies#simple4