

# 2D Polygon Level Design + Dynamic Lights

---

*Author: OtioseEntertainment (Leon Möhring)*

Simple workflow for 2D level design in the Unity Editor. By changing the PolygonCollider the points of the Mesh can be edited easily. The parallax effect is generated automatically and can be either on the inside or the outside of the Mesh. Its pivot point is also adjustable. Additionally multiple custom sprites can be randomly placed on the edges of the Mesh. Another core feature is the 2D dynamic light system. Dynamic and static lights as well as dynamic and static walls are supported and the performance is optimised. Multiple lights and different colours can be easily achieved.

I hope you find this package useful and everything works fine in your projects.

## Package Features

- Mesh from PolygonCollider
- Parallax background layers from Mesh
- Detail placement on Mesh edges
- Dynamic 2D light Mesh for static and dynamic objects

## Getting started

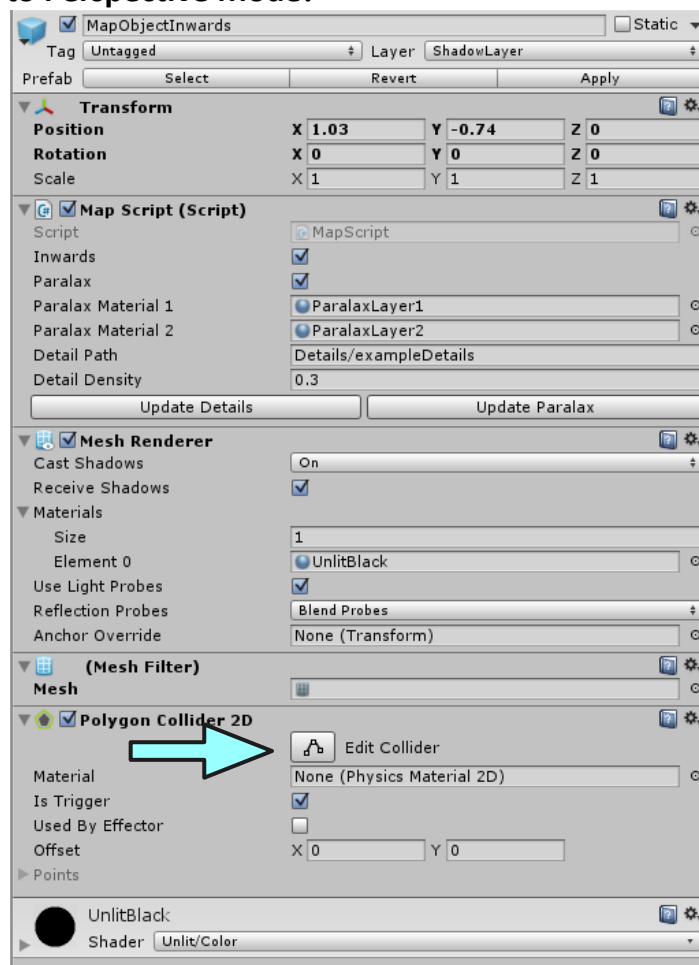
After importing the package you can take a look at the core features in the ExampleScene. A great example of how a game using this system might look is WildGrowth. The design process of one level can be seen here: [https://www.youtube.com/watch?v=cik6KC\\_S3aM](https://www.youtube.com/watch?v=cik6KC_S3aM). The package contains some prefabs to enable you to easily use all features. Furthermore all Resources used by the scripts and prefabs are located inside the Resources folder and you are able to manipulate the Materials and Sprites for customisation of the system.

# MapObjects

The walls of your level can be one of these three prefabs:

- **MapObjectInwards.prefab** ← static, parallax, inwards wall
- **MapObjectOutwards.prefab** ← static, parallax, outwards wall
- **DynamicObject.prefab** ← dynamic, non parallax wall

All of these objects use a PolygonCollider and can be edited by editing the Collider Component. **Important Note: Transform Scale and Transform Rotation should remain the default values!** Rotation (0, 0, 0), Scale (1, 1, 1) The position x, y can be modified, but z should remain 0. It is currently not possible to use a parent object to avoid this problem. The Mesh and Collider will still work, but features like parallax and details will break. All MapObjects and other shadow casting objects must be inside of the ShadowLayer to work with the included light system. **Important Note: For the parallax effect to work the camera must be set to Perspective Mode!**



## Edit object shape:


To edit the shape of the Mesh, click on “Edit Collider” in PolygonCollider2D. (Quick tip: never put a sprite on this GameObject or on any of its components, because it will possibly change the shape of the PolygonCollider and cannot be undone)


## Edit materials:

The material of the MapObject is currently UnlitBlack and can be assigned in the MeshRenderer. To change the materials of the two parallax layers change ParallaxLayer1 and ParallaxLayer2 in the MapScript Component.

## MapScript Component:

**bool** *inwards* ← Will the parallax effect be on the inside or the outside?

Object with inside 

Object with outside 

**bool** *parallax* ← Will the object have a parallax effect or not?

**Material** *parallaxMaterial1, parallaxMaterial2* ← The materials for the two parallax layers

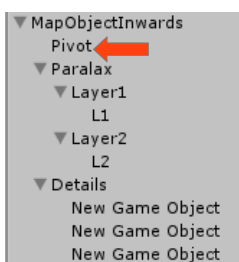
**string** *detailPath* ← The location of the details (Sprite with Sprite Mode: *Multiple*) inside the Resources folder *Resources/Details/exmpleDetails.png*  
Only use the underlined path part showed in the example.

**float** *detailDensity* ← The density of details to place on the edges and corners of the polygonal MapObject. Ranges from 0 to 1 with a maximum of 1 detail on every corner and 1 detail on every edge. (So the detail density also depends on the density of points in the polygonCollider)

Update Details ← Press to randomly place details depending on the two parameters *detailPath* and *detailDensity*. This will apply the changes.

UpdateParallax ← Press to generate the two parallax layers and to apply changes made to the paramaters *inwards, parallax, parallaxMaterial1, parallaxMaterial2*. **Important Note: If there is no ChildObject named Pivot the object will be created. Use the Pivot to change the orientation of the parallax background. Usually it should be located in the center of the MapObject Mesh.**

## MapObject structure:

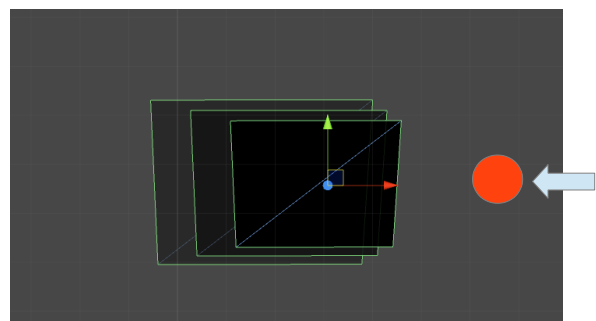
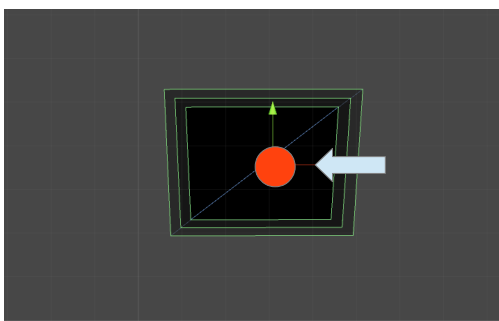


Change the position of the pivot object to change the orientation of the parallax layers.

Feel free to delete the GameObject Parallax and Details to get rid of the parallax effect and the details. Just press the buttons to generate them again.

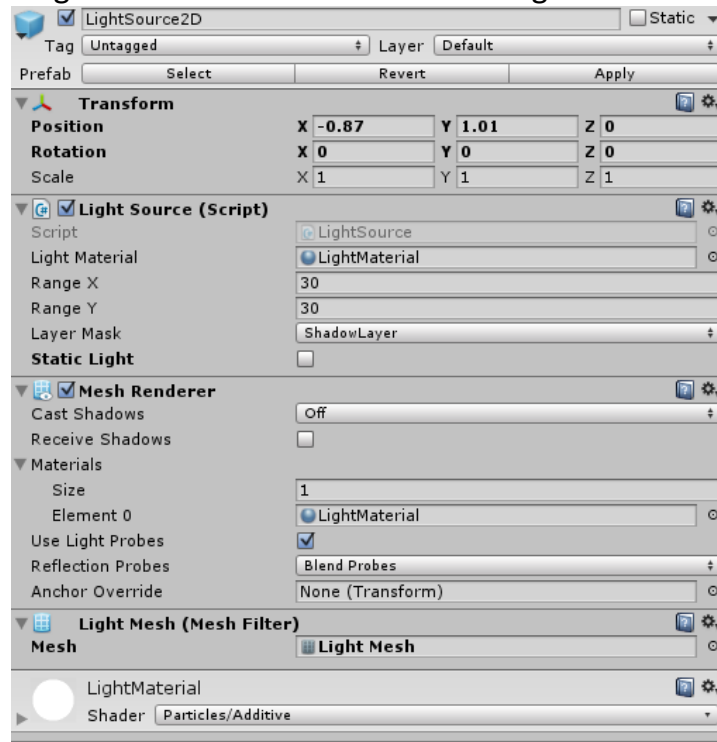
Left: Pivot point centered.

Right: Pivot point on the side of the object



# LightSources

To use dynamic 2D lighting in your game just instantiate the LightSource2D prefab. The LightSource will work with all PolygonCollider2D objects inside of the ShadowLayer (or the layer you specified inside of the LightSource Component). It is recommended that all LightSources and all shadow casting objects stay on position.z = 0. Otherwise the system will still work, but the light Mesh could be rendered wrong.



**Important Note: Dynamic (moving) shadow casters must be tagged “DynamicPos” and to improve performance all static walls should not be tagged like this!**

**Important Note: Transform Scale and Transform Rotation should remain the default values! Rotation (0, 0, 0), Scale (1, 1, 1)**

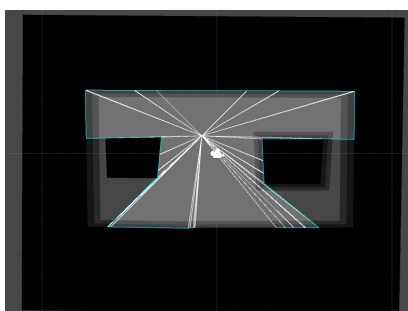
## Light Source Component:

**Material** lightMaterial ← The material used for the light Mesh. Change this to create different coloured lights and different light intensities.

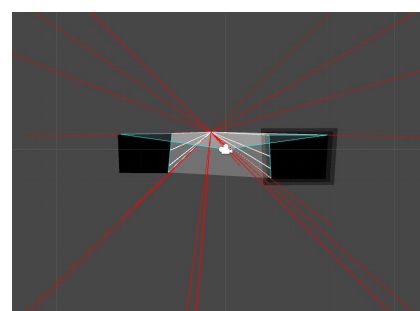
**float** rangeX, rangeY ← The maximum range of the rays the light casts.

**Important Note: The light source only works in closed off areas (mapObjects on all sides).**

Left: Closed area works fine.



Right: Open area results in errors.



**LayerMask** *layerMask* ← The LayerMask to search for PolygonCollider2D to use as shadow casters. By default use the “ShadowLayer”. You can set up different lights with different LayerMasks.

**bool** *staticLight* ← To optimise performance set staticLight for non moving lights to true. This way the Mesh will only be calculated on Load. Be aware that moving shadow casters will not work with this.

**Important Note:** Light Sources will refresh their list of static shadow casters on Start, so if you see lighting glitches just press play and it should be fixed. Some glitches are also caused by too far distances to MapObjects.