UniTile

Thank you for purchasing UniTile, a tile-based 2D map editor for Unity.

Preparing your tileset

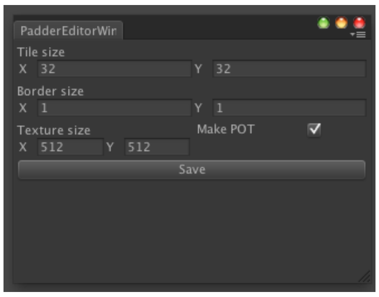
On most platforms, it's best to work with power-of-two (PoT) textures. This means both their width and height should be a power of two (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048).

One problem of creating 2D maps in a 3D environment is edge bleeding (where you can see part of the next or previous tile in the tileset at the edge of the current tile).
A related problem is empty gaps showing up between tiles - this will be discussed later as it is not related to preparing your tileset.

In order to solve the edge bleeding problem, I created a tool called the Texture Padder. With this, you can add a padded border to each tile in your tileset. Please note that it does not just create an empty border around each tile, because that would cause gaps between the tiles instead of edge bleeding. Instead, it will take pixel data from the edges and extrude that.

Important: After using the Texture Padder, your image will have different dimensions. Because of this, you should plan ahead when making the tileset. For instance, if your texture was already 512x512 pixels, and it had 16x16 tiles in it (meaning the tiles themselves are 32x32), adding a 1 border pixel to each tile will result in a 544x544 texture - and since you probably want to keep it PoT, the final image would be 1024x1024. This is not ideal at all.
So instead, you should only have only 15x15 tiles in your original image. This would result in a 510x510 image, and after making it PoT, it would be 512x512.
The math is simple : imageSize = (tileSize + tileBorder * 2) * numberOfTiles. Round that up to the nearest PoT value and you've got the size of the resulting image.

To open the Texture Padder, select the texture you want to modify in the Project panel, and select UniTile/TexturePadder from the top menu, or press alt-t.
You'll get a window that looks like this :


PadderEditorWir
Tile size
X 32 Y 32
Border size
X 1 Y 1
Texture size
X 512 Y 512
Make POT ✓
Save

Please note that it is also possible to open the Texture Padder from a UniTile layer by pressing the "Padder" button.

In this window, you have to set up the size of the individual tiles, and the size of the border you want to add. In most cases, a border of (1, 1) will work fine, but if you plan to scale your texture down later (for instance, if you would like a different version of the texture for lower resolution screens), a value of (2, 2) might be advisable.

You can select whether to keep the texture's dimensions PoT or not. In most cases, you will want to keep this checked.

The texture size is automatically calculated whenever you change one of the other values. You can modify it yourself if needed, but you can not make it smaller than the space that will be needed for the tiles. Making it bigger might be useful for adding extra graphics to the tileset that are not related to the tileset, if you're trying to keep the amount of draw calls in your scene down to a minimum, but usually the automatic size should be fine.

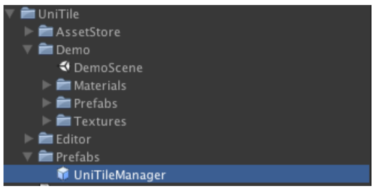
After you press Save, you will be presented with a Save dialog. It is recommended that you save your texture with a new name.

After the texture has been created, you need to set it's import settings, and create a material for the texture. UniTile comes with a shader called "2D Art for iPhone" that can be used for the material.

Please note that you will have to tell UniTile about the tile border (the Tile Border Size property). This will be discussed in the next section.

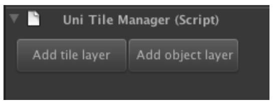
The Basics

The first step in creating a map, is adding a UniTileManager prefab to your scene. To do this, go to UniTile/Prefabs/UniTileManager in the Project panel, and drag it to the Hierarchy panel or the Scene view.

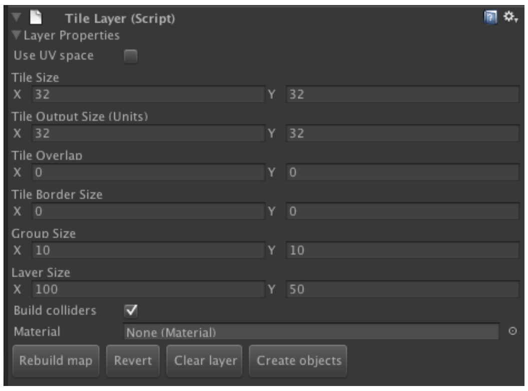


```
UniTile
  AssetStore
  Demo
    DemoScene
    Materials
    Prefabs
    Textures
  Editor
  Prefabs
    UniTileManager
```

Once it is in place, press the "Add tile layer" button on the UniTileManager's inspector panel.



This will create a new GameObject in the scene called "Layer 1". Go ahead and select it. You'll get a panel that looks like this :



Tile Layer (Script)

Layer Properties

Use UV space	☐	
Tile Size	X 32	Y 32
Tile Output Size (Units)	X 32	Y 32
Tile Overlap	X 0	Y 0
Tile Border Size	X 0	Y 0
Group Size	X 10	Y 10
Layer Size	X 100	Y 50
Build colliders	✓	
Material	None (Material)	

Let's go through the different options.

- **Use UV space**
  By default, UniTile uses texture space for it's coordinates. This means that a value of 32 corresponds to 32 pixels in the texture. When you check this toggle, UniTile will use UV coordinates (0 to 1) rather than pixel coordinates. This is useful if you plan to resize your texture later.

- **Tile Size**
  The size of the individual tiles in your tileset - either in pixels or UV coordinates.

- **Tile Output Size**
  The dimensions of a tile in Unity units. If you set this to the same value as the Tile Size, one pixel will be mapped to one unit (unless you scale the layer).

- **Tile Overlap**
  As previously mentioned, sometimes it is possible to see small (1px) gaps between tiles. If you see this happening, set the Tile Overlap to a very low value. Something like 0.1 will usually work well, but some experimentation might be in order.

- **Tile Border Size**
  You can add a border to the tiles in your tileset to fix edge bleeding (see "Preparing your tileset"). If you have one pixel on each side of your tiles, set the Tile Border Size to (1, 1).

- **Group size**
  The amount of tiles that will be grouped together in tile groups. This will be discussed later on. The default values should be fine for most cases.

- **Layer size**
  The size of the current layer (amount of tiles). Usually, all the layers in your map will have the same size, but if you want to you can have different sizes.

- **Build collision**
  Use this to turn off creation of the creation of Box colliders. This will be discussed later on.

- **Material**
  The material used for the layer. Without a material, you can't draw any tiles.

- **Rebuild map**
  This will apply any changes that you made in the above properties to the layer, and recalculate the entire map. So if you change the Layer's size, it will only resize the layer once you press this button. Please note that the changing material is instant.

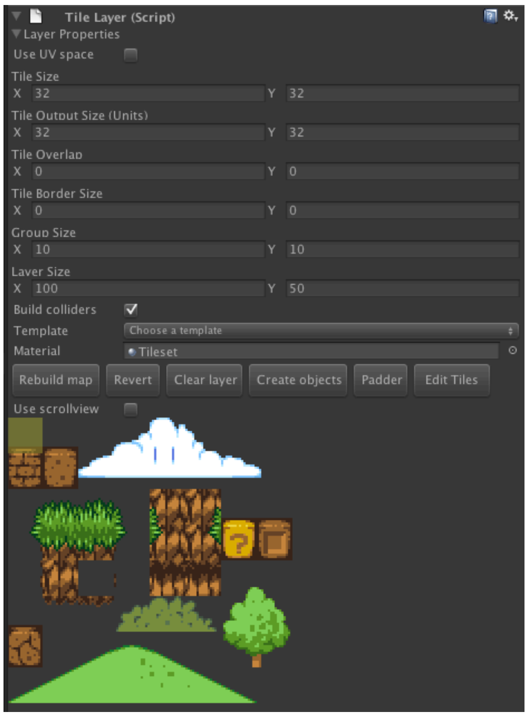
- **Revert**
  If you changed any of the properties, you can press Revert to go back to the current state. Deselecting the GameObject has the

same effect.

- **Clear layer**
  This will clear the entire layer. You can still use Undo to get your map back.

- **Create objects**
  This will create colliders and instances of prefabs. This will be discussed later on.

Once you selected a material, you get a few more options. The panel now looks like this :



Tile Layer (Script)
Layer Properties
Use UV space ☐
Tile Size
X 32 Y 32
Tile Output Size (Units)
X 32 Y 32
Tile Overlap
X 0 Y 0
Tile Border Size
X 0 Y 0
Group Size
X 10 Y 10
Layer Size
X 100 Y 50
Build colliders ✓
Template Choose a template
Material Tileset
Use scrollview ☐

Let's review the new options.

- **Template**
  Templates are a group of tiles that you either pick from the tileset or your map, in order to reuse later. From this dropdown, you can choose one of your templates, or add your current selection as a new template. More on this later.

- **Padder**
  Press this button to open the Texture Padder for the currently selected material's main texture.

- **Edit tiles**
  You can use this to set tile properties for all currently selected tiles. More on tile properties later.

- **Use scrollview**
  This checkbox will show your tileset in a scrollview. This is useful for working with a large tileset, or when you don't have a lot of monitor space. When unchecked, the tileset will be scaled to the size of the panel.

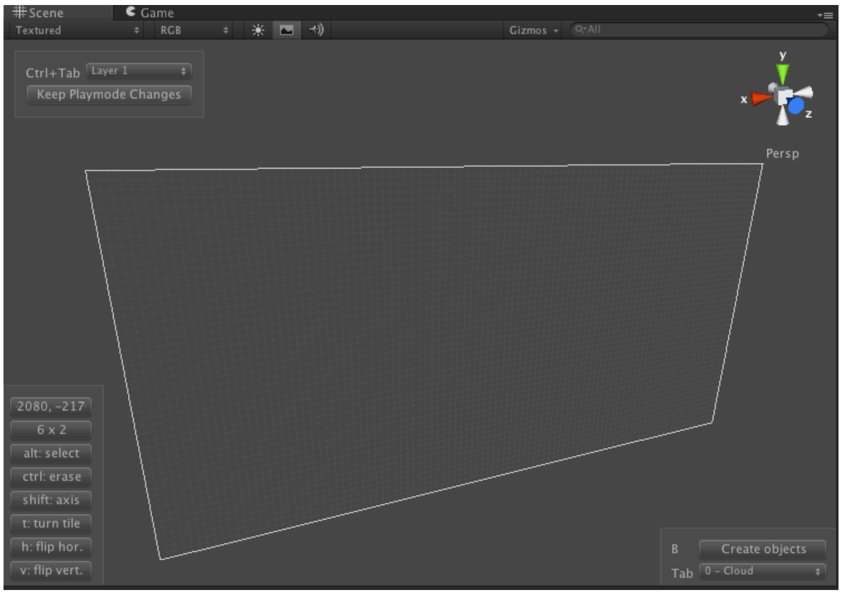
Drawing basics

To draw a map, you need to select a tile from your tileset. Just click any tile, and it will be selected. You can also click and drag to select multiple tiles, or right click to select a blank tile.

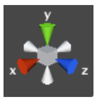
Next, you should select the Scene view. To do this, just right click anywhere in the scene.

UniTile will display a grid in the scene view. If you don't see this, make sure you have a layer selected. If you still don't see it, press F while in the Scene view, and Unity will focus on the currently selected GameObject (in this case, the empty layer).

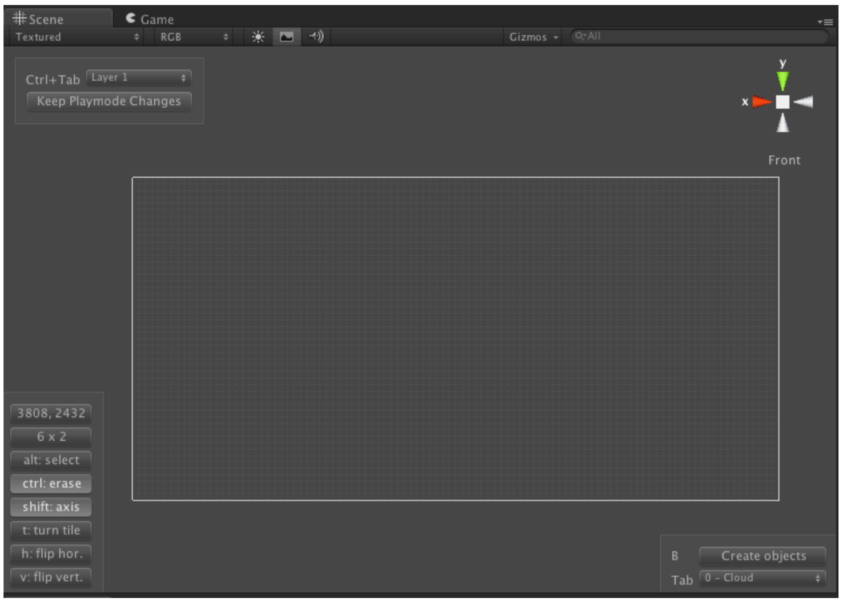
It should look like this :



While it is possible to draw while in a perspective view, it is usually more practical to go into orthographic mode, while viewing the layer from the front. To do this, simply press the opposite side of the blue arrow in the gizmo at the top right of the scene view. You might have to turn around a bit to see it.



This will result in a view like this :



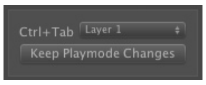
You can zoom in and out with the scroll wheel of your mouse. And, providing that you have the Scene view selected (if not, right click in the scene), you can move around using the arrow keys of your keyboard.

Now all you have to do is start drawing by clicking the left mouse button and dragging within the grid.

Drawing options

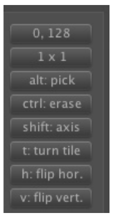
There are three panels in the Scene view that help you while working on maps.

Panel 1



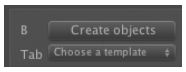
- This panel allows you to select other layers (if you have any). As you can see, you can also press Ctrl+Tab (while the Scene view is selected) to cycle between layers.
- The "Keep Playmode Changes" option allows you to make changes to your map while you are testing your game.

Panel 2



- Your current location in the map, in Unity units
- The size of your current selection, in number of tiles
- Alt modifier : Hold alt and click in the map to pick a tile. You can also click and drag to select multiple tiles
- Ctrl modifier : Hold down ctrl while drawing to erase tiles
    - Note : If you press both Alt and Ctrl while drawing, you will cut those tiles
- Shift : Hold down shift while drawing to limit UniTile to either horizontal or vertical drawing (based on your first movement).
- Turn: Press T to rotate your current selection 90 degrees clockwise
- Flip horizontal : Press H to flip your current selection horizontally
- Flip vertical : Press H to flip your current selection vertically

Panel 3



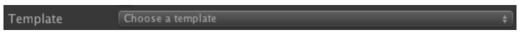
- Clicking "Create objects" or pressing "B" has the same effect as the "Create objects" button in the inspector panel, as discussed above.
- The Templates drop down allows you to quickly select templates (see further on). Pressing Tab cycles between templates.

Tileset Objects

UniTile automatically creates .Tileset objects in your Project, with the same name as the material used for the tileset. Usually you shouldn't have to worry about these objects, but it's useful to be aware that they exist and what they're used for.
They contain information about the tiles in the tileset (see the "Tile Properties" section), but they also store your saved Templates (see "Templates").

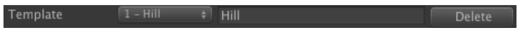
Templates

You can save the currently selected tiles (whether you selected them straight from the tileset or picked them from the map) as a template.



To do this, you need to have a layer selected, and open the "Choose a template" popup box in the inspector. Select the "Add current selection as a template" option.
Once you've added one or more templates, you can select them from this list, or from the list in Scene Panel 3 (see above).

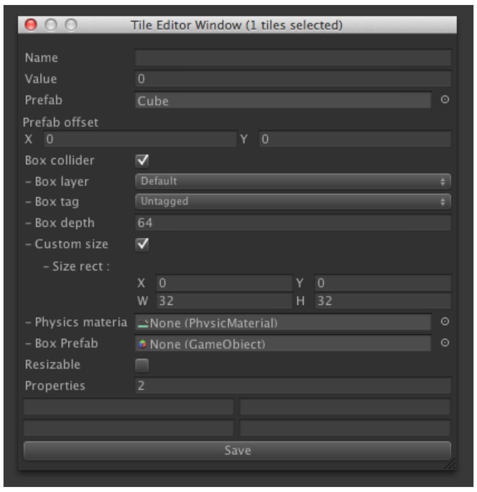
Once you have selected a template from the List in the inspector, you can rename or remove the template.



Tile Properties

Each tile in a tileset can have custom properties assigned to it. To open the Tile Properties window, double click on a tile in the tileset, or press the "Edit Tiles" button.
When using the "Edit Tiles" button, you can set properties on multiple tiles at once.

The Tile Properties window looks like this :


Tile Editor Window (1 tiles selected)

Name
Value 0
Prefab Cube
Prefab offset
X 0 Y 0
Box collider ✓
- Box layer Default
- Box tag Untagged
- Box depth 64
- Custom size ✓
- Size rect :
X 0 Y 0
W 32 H 32
- Physics materia None (PhysicMaterial)
- Box Prefab None (GameObject)
Resizable
Properties 2

Save

Let's go through all the properties.

- **Name**
  Here you can enter a name for the tile. This can be accessed through code (see the API section).

- **Value**
  Here you can enter a numeric value. This can be accessed through code (see the API section).

- **Prefab**
  You can drag a prefab to this field. Instances of this prefab will be instantiated when you press the "Create objects" button.

  - **Prefab Offset**
    This is only visible if you selected a prefab. Here you can assign an offset relative to the bottom-left corner of the tile in your map.

- **Box collider**
  If you check this box, a box collider will be created when you press the "Create Objects" button. More on this later.

  - **Box layer**
    The Unity layer that will be assigned to the box colliders created for this tile

  - **Box tag**
    The Unity tag that will be assigned to the box colliders created for this tile

  - **Box depth**
    The depth of the created box colliders along the z axis

  - **Custom size**
    When checked, you will be able to enter a custom dimensions for this box collider

    - **Size rect**
      Custom dimensions for the created box colliders, relative the bottom-left corner of the tile.

  - **Physics material**
    The physics material that will be assigned to box colliders for this tile

  - **Box prefab**
    If you leave this blank, a new GameObject will be created to add the box collider to, but you can also assign a prefab that will be instantiated for this.

- **Resizable**
  When checked, this tile will be marked as resizable. See the section on Optimizations for more info.

- **Properties**
  You can assign any number of custom properties to a tile. Just enter the number of properties you want, and then you can enter the key/value pairs below. This can be accessed through code (see API section).

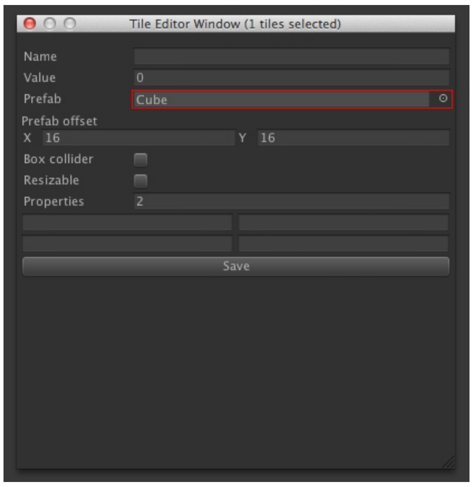
Important: Don't forget to press "Save" when you're done editing the properties.

Colliders and prefabs

As mentioned above, you can configure tiles so that when you press the "Create objects" button, it will spawn prefabs or box colliders.

Prefabs

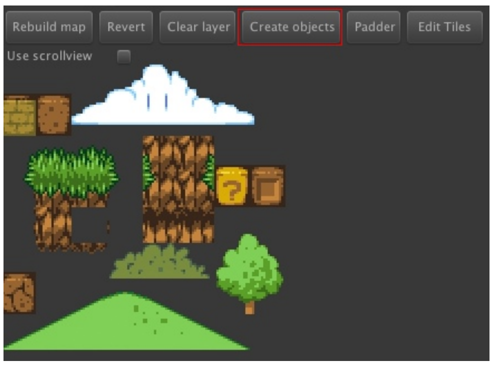
Let's assign a prefab to a tile. There's a simple cube prefab included in the package.



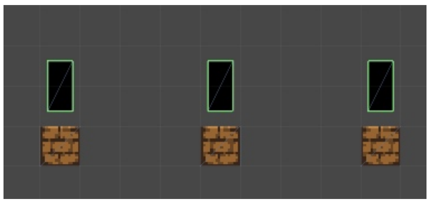
Name
Value 0
Prefab Cube
Prefab offset
X 16 Y 16
Box collider
Resizable
Properties 2

The prefab offset will automatically be set to the center of the tile (for a tile output size of 32x32, that's 16,16). Let's move the prefab up a little, to 16, 64, and press Save.

Make sure you have drawn this tile in your map. Then, press the "Create objects" button :



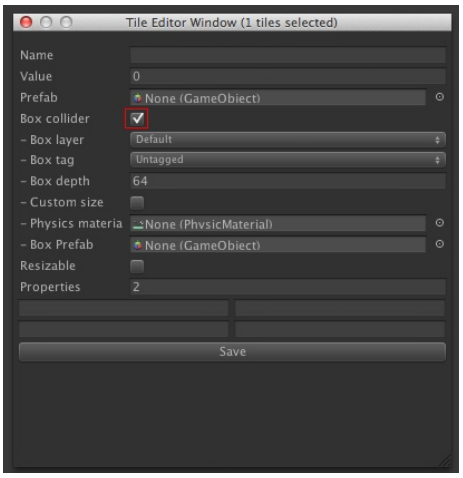
After you press this button, UniTile will create all the prefabs you selected. The result will look something like this :



Please note that if you press the button again, all previous prefabs (and box colliders) will be removed before the new ones will be created.

Box Colliders

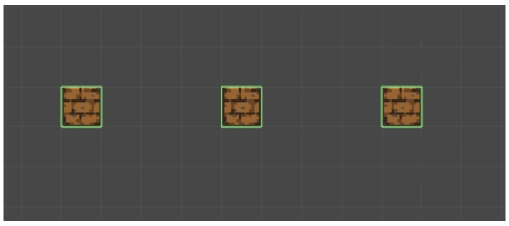
Another useful feature is the automatic creation of box colliders. To do this, the first thing you need to do is select the "Box collider" checkbox for your tile.


Tile Editor Window (1 tiles selected)

Name	
Value	0
Prefab	None (GameObject)
Box collider	✓
- Box layer	Default
- Box tag	Untagged
- Box depth	64
- Custom size	☐
- Physics materia	None (PhysicMaterial)
- Box Prefab	None (GameObject)
Resizable	☐
Properties	2

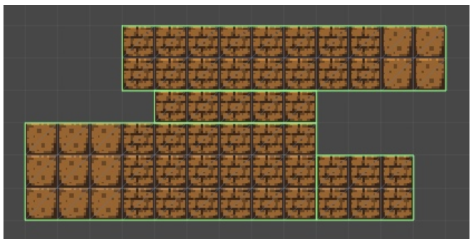
You can leave the other parameters (layer, tag, depth, size, physics material and box prefab) as they are. For a list of what they do, please see above.

Press Save, and the press the Create Objects button. The result will look like this :



You can clearly see the generated boxcolliders in the Scene view. By default, they will be the size of one tile in the grid, but this can be changed with the Custom size property.

UniTile has an algorithm that automatically groups together adjacent boxcolliders. Here's an example :



You'll notice that it grouped the colliders even though they belong to a different type of tile. It does this as long as the layer, tag, depth, physics material and prefab are the same, providing you haven't set a custom size for those tiles.

Optimizations

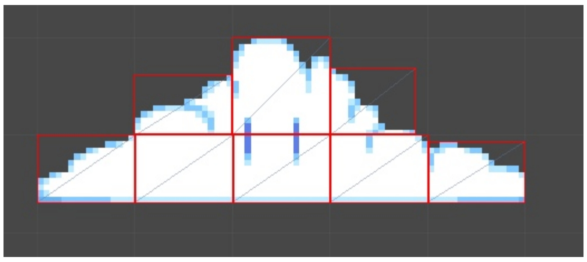
Tile Groups

UniTile works by dynamically creating new GameObjects, (tile groups), which contain a mesh that displays a number of tiles. This is done for performance reasons - drawing a large map into one mesh could get slow, so it is divided into groups.
These tile groups are added (as children of the layer) and removed automatically as needed so you don't have to worry about them too much. You can set the size of the groups per layer in the inspector panel.

Removal of transparent pixels

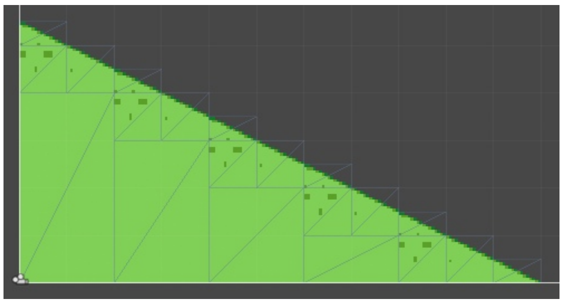
UniTile automatically crops tiles so transparent pixels are removed from the edges, in order to reduce the amount of overdraw. For this to work, your texture has to be ARGB32, RGBA32, RGB24, AutomaticTruecolor or Alpha8.

This image illustrates how tiles get cropped :



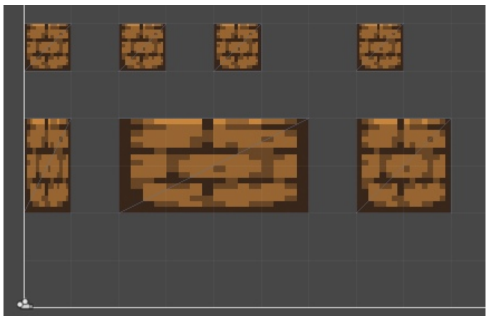
Scaling tiles

If you checked the "Resizable" box in the Tile properties window for a tile, UniTile will stretch this tile when you put multiple instances of it next to each other. This way, the amount of polygons can be greatly reduced. Here's an example :



UniTile has an algorithm that finds boxes of adjacent resizable tiles. This is done within each tile group, so it won't stretch tiles across multiple groups.

This obviously only looks good with tiles that contain one single color. If you use it on a tile with more detail, it will look odd, like this :

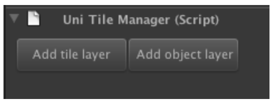


This stretching effect might be useful in some cases, but usually it's best to use it only on tiles of a single color.

Object Layers

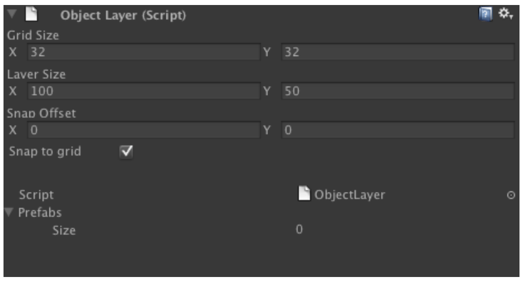
Object Layers make it easy to place and move GameObjects in your maps. This is mostly useful for working with sprites.

Select your UniTile Manager, and press "Add object layer".



This will create an Object Layer named "Object Layer 1". Go ahead and select it.

You'll get an inspector panel that looks like this :



Object Layer (Script)
Grid Size
X 32 Y 32
Layer Size
X 100 Y 50
Snap Offset
X 0 Y 0
Snap to grid ✓

Script ObjectLayer
Prefabs
Size 0

- **Grid Size**
  The size of the grid in which objects will be placed (if you have "Snap to grid" selected). Usually, this should match the "Tile output size" of your tile layers.

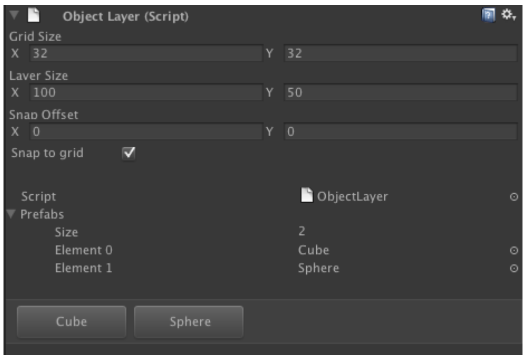
- **Layer size**
  The size of the layer, in number of tiles. Usually, this should match the "Layer size" of your tile layers.

- **Snap offset**
  The offset at which objects are placed in the grid when "Snap to grid" is selected. Usually this should be 0, 0 or half the grid size (in this case 16, 16).

- **Snap to grid**
  Keep this checked if you want objects to be snapped to the grid.

- **Prefabs**
  An array of prefabs that you can assign to this layer for easy instantiation.

Go ahead and set the Size of the Prefabs array to 2, and assign two prefabs to the array. The result will look like this :



Object Layer (Script)
Grid Size
X 32 Y 32
Layer Size
X 100 Y 50
Snap Offset
X 0 Y 0
Snap to grid ✓

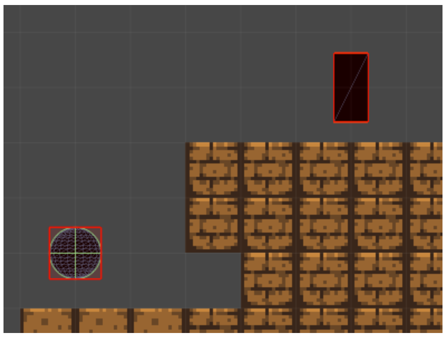
Script ObjectLayer
Prefabs
Size 2
Element 0 Cube
Element 1 Sphere

You may notice two new buttons, with the same names as the prefabs.

If you press "Cube", you'll notice that an instance of the prefab will be created at the center of the scene, as a child of the Object Layer.

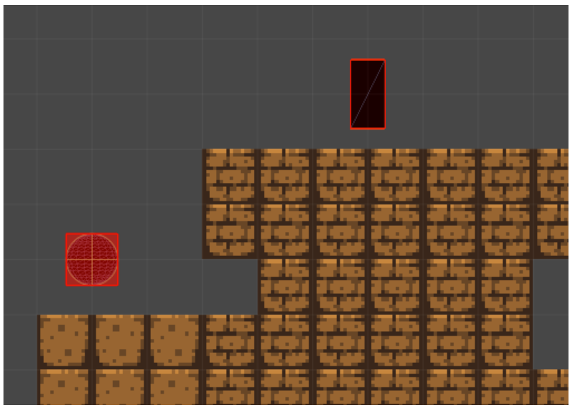
Now you can just drag the new cube around in the scene view.

After creating both a cube and a sphere, the scene will look like this :



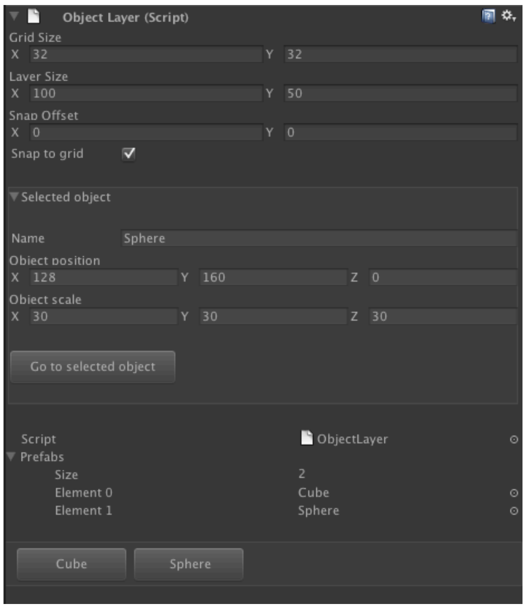
Note that UniTile draws a red bounding box around the objects. The bounding box is calculated by looking at the renderer or collider in the object. If it doesn't find those, it will draw a default bounding box.

When you click on one of them, it will be highlighted, like this :



If you want to get rid of an object, you can control-click it.

The highlighted object will also be shown in the inspector panel, under the "Selected object" section :



Object Layer (Script)
Grid Size
X 32 Y 32
Layer Size
X 100 Y 50
Snap Offset
X 0 Y 0
Snap to grid ✓

Selected object

Name Sphere
Object position
X 128 Y 160 Z 0
Object scale
X 30 Y 30 Z 30

Script ObjectLayer
Prefabs
Size 2
Element 0 Cube
Element 1 Sphere

Here you can easily change the object's name, or fine-tune it's position and scale. Pressing the "Go to selected object" button will select the object in the Hierarchy panel.

API

The TileLayer class has a number of methods for accessing tile information and properties.
It is recommended to access these methods at the start of your level, and store any values you need later.
Most of these methods require the x and y coordinates of a tile. These coordiantes are not in world or camera space, but the position of the tiles in the layer.
An example: The tile at the bottom left of your map is accessed with (0, 0), the tile next to it with (1, 0) and the tile above it with (0, 1).

Include Tile Data Define

At the top of the TileLayer class, there's this line :

```
#define INCLUDE_TILE_DATA
```

If you don't plan to use any of the API methods, you should remove or comment that line. This should save some memory.

Methods

- int GetTileId(int x, int y)
Gets the number of the tile at x, y. Each tile in the tileset is numbered, in reading order (so the top left tile is 0, the one next to that is 1, and so on).

- UniTileTile GetTile(int x, int y)
Returns an object with the tile properties for the tile at x, y. You can also access the individual properties with the methods below, but if you need to access a lot of different properties, getting this object will be more performant.

- TileInstance GetTileData(int x, int y)
Returns an object with information on the tile instance. This information is the tile's number, it's rotation (0 = 0 degrees, 1 = 90 degrees, 2 = 180 degrees, 3 = 270 degrees), and whether it's been flipped horizontally and/or vertically.

- GameObject GetPrefab(int x, int y)
Returns a reference to the prefab assigned to the tile at x, y

- string GetName(int x, int y)
Returns the name property assigned to the tile at x, y

- float GetValue(int x, int y)
Returns the value property assigned to the tile at x, y

- string GetProperty(int x, int y, string key)
Returns a property from the key/value pairs of the tile at x,y. This is slower than getting the name or value properties, so only use this if you need more properties.

- bool HasBox(int x, int y)
- int GetBoxLayer(int x, int y)
- string GetBoxTag(int x, int y)
- float GetBoxDepth(int x, int y)

These methods return information used for the automatic generation of box colliders. Usually you will not need to access these yourself.

- void InstantiatePrefabs()
Creates instances of the assigned prefabs, and also creates box colliders for the tiles that have the "Box Collider" checkbox checked. You can do this in the editor by pressing the "Create Objects" button, but if you prefer you can do this by code by calling this method.

- void SetTile(int x, int y, int newTileId)
Use this method to change a tile. This will need to recreate the entire group that the tile belongs to, so if your layer uses a large group size, this is not very performant.