Guide to an in-depth understanding of the Tile Builder Package.

Please read this part

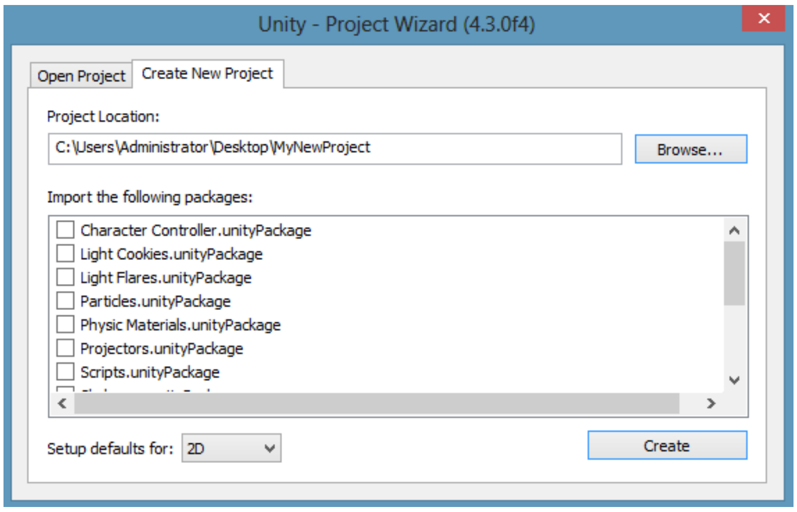
First, I would like to thank you for purchasing the Tile Builder Package. In this guide I will be documenting a clear vision of how to use this level editor tool. If something remains unclear or not well enough explained, please do contact me at wouterversloot@gmail.com as I will be glad enough to explain it to you.

I will be explaining everything step-by-step with images so that every user, pro or beginner, can keep up in a solid way. The most important notes will be repeated at the bottom of this guide. I hope you enjoy using this package as I enjoyed creating it.

Please don’t forget to leave a review on my asset store page!

So we downloaded the Package, what's next?

Create a new project, move it where ever you want and put its Setup defaults to: 2D. You don't have to import anything just yet! Hit the create button and wait for Unity to start.



Unity - Project Wizard (4.3.0f4)

Open Project | Create New Project

Project Location:

C:\Users\Administrator\Desktop\MyNewProject

Browse...

Import the following packages:

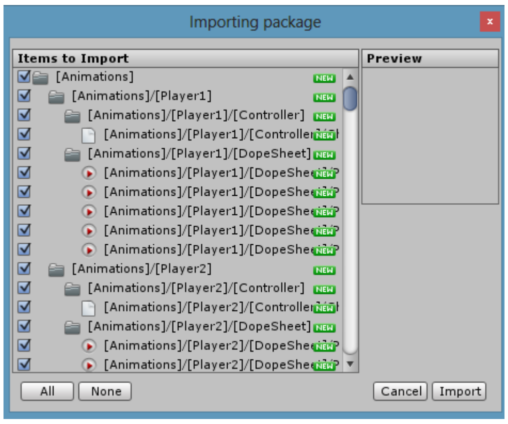
- [ ] Character Controller.unityPackage
- [ ] Light Cookies.unityPackage
- [ ] Light Flares.unityPackage
- [ ] Particles.unityPackage
- [ ] Physic Materials.unityPackage
- [ ] Projectors.unityPackage
- [ ] Scripts.unityPackage

Setup defaults for: 2D

Create

If you already have a previous version of the Tile Builder Package, please remove all files (except your Tilesheets and PNG's) because I made a major overhaul in version 1.2. See the Change log file for more information.

Now once we are in Unity, go to the Asset Store window and Import the package. Make sure to hit the all button to make sure you import everything.



Importing package

Items to Import | Preview

- [x] [Animations] NEW
- [x] [Animations]/[Player1] NEW
- [x] [Animations]/[Player1]/[Controller] NEW
- [x] [Animations]/[Player1]/[Controller NEW
- [x] [Animations]/[Player1]/[DopeSheet] NEW
- [x] [Animations]/[Player1]/[DopeShe NEW
- [x] [Animations]/[Player1]/[DopeShe NEW
- [x] [Animations]/[Player1]/[DopeShe NEW
- [x] [Animations]/[Player1]/[DopeShe NEW
- [x] [Animations]/[Player1]/[DopeShe NEW
- [x] [Animations]/[Player2] NEW
- [x] [Animations]/[Player2]/[Controller] NEW
- [x] [Animations]/[Player2]/[Controller NEW
- [x] [Animations]/[Player2]/[DopeSheet] NEW
- [x] [Animations]/[Player2]/[DopeShe NEW
- [x] [Animations]/[Player2]/[DopeShe NEW

All | None | Cancel | Import

Upon importing, you should see that your Project window (Ctrl + 5) has changed. It has added some default, and important, new folders to locate its assets. At the time of writing, this will probably look a little different than what it will look like in the final product, because I haven't released this asset just yet.

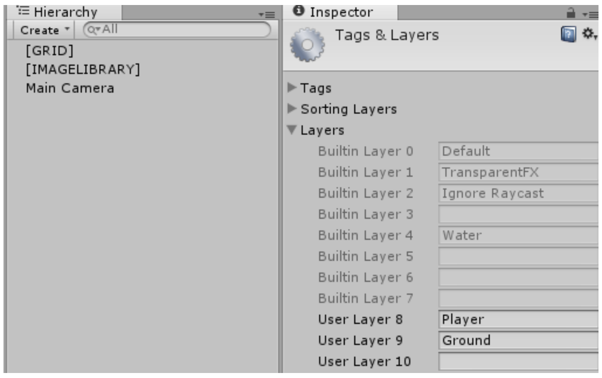
Setting up the default play area

If you wish to jump right into action, there is a MainScene ready to be explored. If you want to take the full tour, this is where we move on.

Save your new scene and add a basic script into our scene.
Onto your **Main Camera**, drag the script: “LerpToPlayer.cs”

Let’s also add some interactivity to our scene. A scene with no playable character isn’t really fun to play with right? Right!

First off, locate and choose one of the possible characters you wish to play with in your *“[Prefabs]/[Players]”* folder. Drag it onto the screen, but don’t press play yet.
Then, add two simple Layers. Name one ***“Player”*** and the other ***“Ground”***.



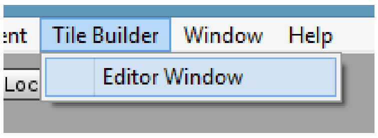
Builtin Layer 0	Default
Builtin Layer 1	TransparentFX
Builtin Layer 2	Ignore Raycast
Builtin Layer 3	
Builtin Layer 4	Water
Builtin Layer 5	
Builtin Layer 6	
Builtin Layer 7	
User Layer 8	Player
User Layer 9	Ground
User Layer 10	

Now, on your Player prefab, set the layer to *“Player”*.
Next, locate the object the player will be walking on. This object is found in the *“Resources/[Prefabs]/”* folder. Select the **TilePrefab** and set its layer to ***“Ground”***.

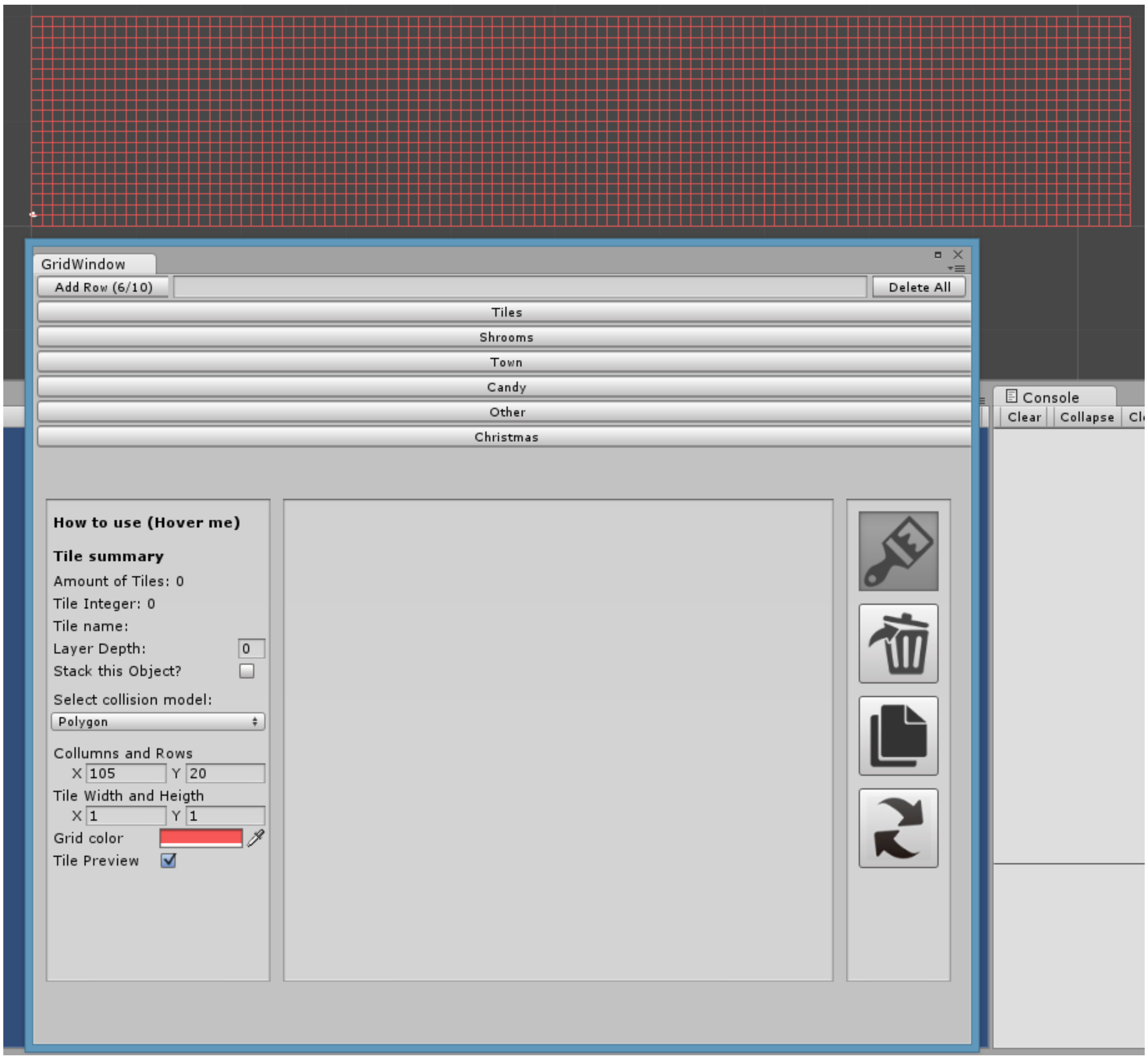
We made sure that, when we are done building our level, our player can walk over it whenever we hit play.

Open Editor Window, Grid and _LevelParent

To open the Editor Window, simply navigate to the top of your unity application and select the Tile Builder tab inside the tab menu.



A new window should appear. Inside the Editor Window there are a few settings you can change. More information on these settings later. Also, a grid graphical interface and a _LevelParent gameObject are spawned inside the scene view. The grid interface will represent the area you can build your tiles into, and the _LevelParent gameObject acts as a parent object that holds every spawned tile so that your Hierarchy will not get cluttered with lots of tiled objects.



How to use (Hover me)

Tile summary

Amount of Tiles: 0

Tile Integer: 0

Tile name:

Layer Depth: 0

Stack this Object? ☐

Select collision model:

Polygon

Collumns and Rows

X 105 Y 20

Tile Width and Heigth

X 1 Y 1

Grid color 🟥

Tile Preview ✓

Images!

Let’s get on with the fun and interesting part. In order to spawn images into the grid slots, you will have to insert your own atlas sprite sheet. (Don’t worry, I have also included a sprite sheet with over 500 different sprites for you to experiment with).

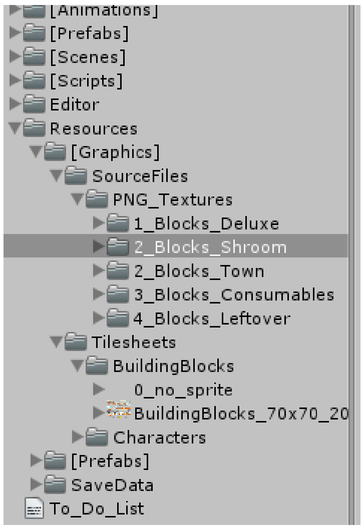
To create an atlas, I used, and recommend, *TexturePackerGUI* available here: http://www.codeandweb.com/texturepacker. Simply insert your own PNG tile textures into the programme, play with the settings and publish a fresh tile sheet texture. The tile sheet and settings I used can be found here: http://i.imgur.com/gJMYB18.png. Also select the option “*Clean transparent pixels*” at the bottom that just fell out of screen.
*Set the common divisor to the width and height of your tile, mine are 70 x 70.

Once you have created your atlas, drag the tile sheet into Unity. The tile sheet has to have a specific path for the tile builder to be found. Either put it in: *“Resources/[Graphics]/Tilesheets”*

or change the path (in the ImageLibrary.cs file).

*“Object[] spriteImageFromDataPath = Resources.LoadAll**(@"[Graphics]/Tilesheets**", typeof(Sprite));”.*

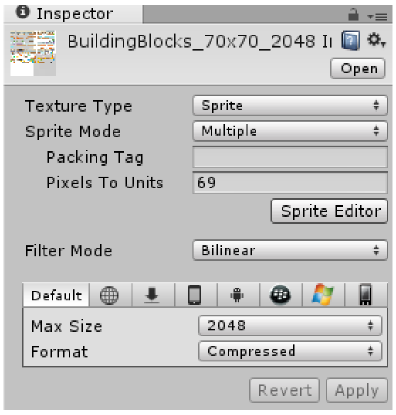
Your project folder should now look something like this:


```
▶ [Animations]
▶ [Prefabs]
▶ [Scenes]
▶ [Scripts]
▶ Editor
▼ Resources
  ▼ [Graphics]
    ▼ SourceFiles
      ▼ PNG_Textures
        ▶ 1_Blocks_Deluxe
        ▶ 2_Blocks_Shroom
        ▶ 2_Blocks_Town
        ▶ 3_Blocks_Consumables
        ▶ 4_Blocks_Leftover
    ▼ Tilesheets
      ▼ BuildingBlocks
        ▶ 0_no_sprite
        ▶ BuildingBlocks_70x70_20
      ▶ Characters
  ▶ [Prefabs]
  ▶ SaveData
  To_Do_List
```

Inside the BuildingBlocks folder, I have added my tile sheet “BuildingBlocks_70x70_2048”. Please make sure that you put any of your own tile sheet atlases inside the “Tilesheets” folder for the Editor Window to find it.

Next up: Importing

Click on your tile sheet, in my case “BuildingBlocks_70x70_2048” and change its settings to the following:


Inspector

BuildingBlocks_70x70_2048 I

Texture Type	Sprite
Sprite Mode	Multiple
Packing Tag	
Pixels To Units	69
Filter Mode	Bilinear
Max Size	2048
Format	Compressed

*Max size should be the size of your imported squared tile sheet.
*Pixels to Units should be the width/height of your average tile minus 1. This will reduce seams while placing tiles next to each other. My tiles are 70x70 so a Pixel to Unit ratio of 69 should do!

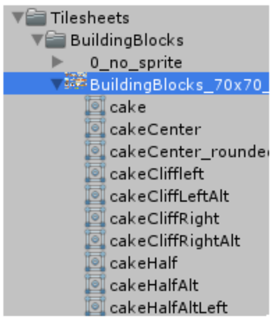
Next up, open the Sprite Editor and slice all your texture images. You can either do this automatically or by slicing a grid.

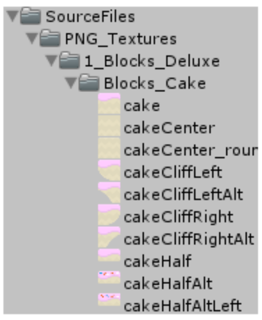
I chose to do this automatically because in the TexturePackerGUI toolkit, I have already separated each tile with a 4x4 offset. If you wish to do it by grid, then you will have to place every tile against each other. Both options have their own pro’s and con’s.

If you wish to do it automatically you will have to manually check each and every tile if its size is still 70x70 (the Sprite Editor cuts off alpha in textures).
If you wish to do this by grid-style, you are risking the chance of some textures to have an extra 1 pixel border on each side of the sprite. This 1 pixel border is taken from the texture space next to it which is rather sloppy.

Done slicing

When you are done slicing, you have two options to think about. Either rename every sprite so that the Editor Window can categorize the different sprites. For instance a tag with the name “Cake” will only find the sprites that have the word Cake in them and leave all other sprites not shown. Or keep their default names; “YourTileSheetName01”, “YourTileSheetName02” …. “YourTileSheetName235”. Once you keep their default names, and search for the tag “YourTileSheetName”, it will show every sprite accompanied from that texture atlas. If you wish to quickly prototype your art assets, and you keep getting newer and / or updated versions of your current art, then I would highly recommend to use the second option and not wasting time on naming each individual sprite.

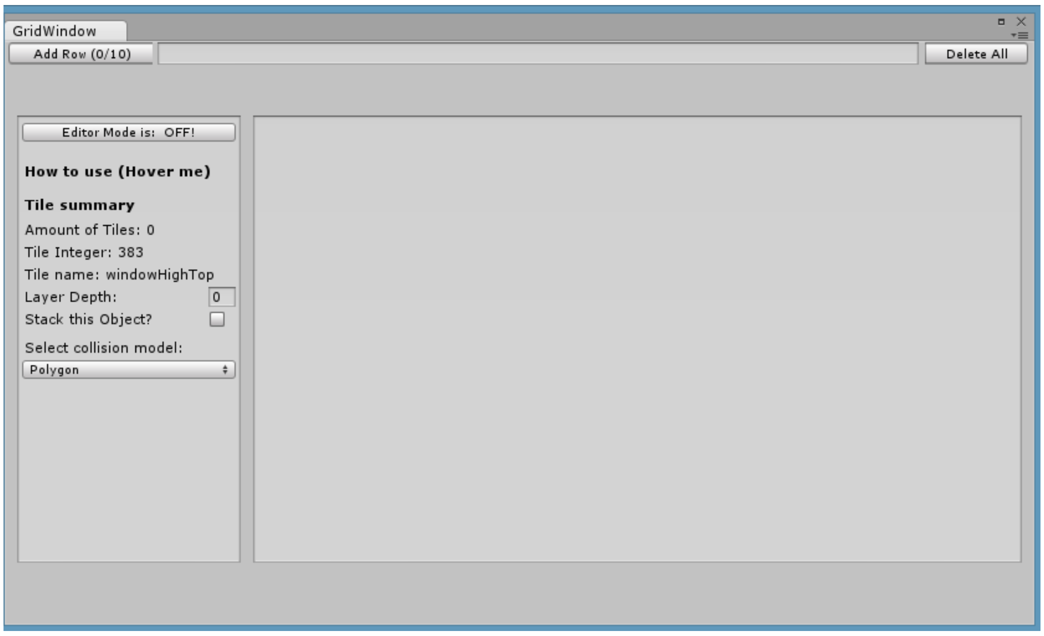




If you wish to rename each individual sprite, please make sure you do not have duplicated names for your tiles.

Editor Window, finally!

A new window should appear: (Changed a little in the newer version).



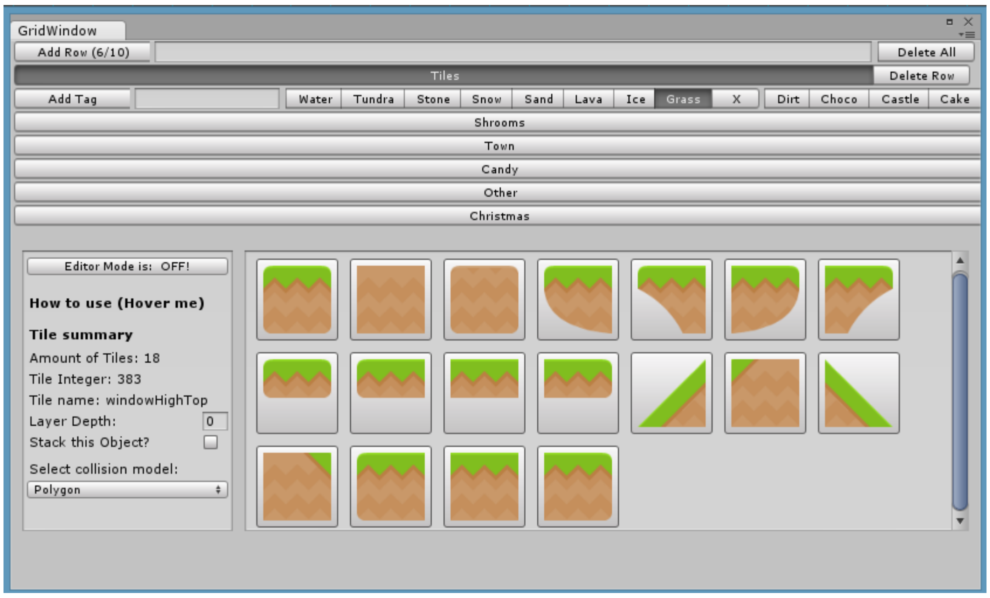
How to use (Hover me)

Tile summary

Amount of Tiles: 0
Tile Integer: 383
Tile name: windowHighTop
Layer Depth: 0
Stack this Object?
Select collision model:
Polygon

By default, its rather empty. But you are able to customize it to your own liking! By inserting a name and clicking "Add Row" you will insert a new category. Once pressed on that tab, a new dropdown follows. This category can hold all sorts of keywords you wish to add to find the tiles from your tile sheet. The way this system works is as the following: Whenever you insert a tag like "*Grass*", "*Candy*", "*Choco*", "*House*" or "*Cake*", it will find every sprite available with the matched name in it. It acts like a search bar that categorizes your tiles.
Note that Rows and Tags cannot start with a number, since numbers are used for internal code flow.

I have taken the liberty to insert a couple of generic tile names into the editor window for you to play around with:



How to use (Hover me)

Tile summary

Amount of Tiles: 18
Tile Integer: 383
Tile name: windowHighTop
Layer Depth: 0
Stack this Object?
Select collision model:
Polygon

Buttons, what do they do?

Add Row:	Adds a new row that can hold new categories.
Delete All:	This button deletes every row and tag you inserted.
Add Tag:	This button adds a new tag, or category, into your row. These tags act as keywords. A tag named "*Grass*" will return every available sprite with the word "*Grass*" in it.
X button:	Next to the selected tag, this button will delete the selected tag.
Delete Row:	Deletes the whole row including the tags accompanied with it.

Summary

Amount of tiles:	The amount of tiles this tag holds.
Tile Integer:	The unique number used in the code flow per tile texture.
Tile Name:	Name of the PNG tile.
Layer Depth:	Z-Depth of the tile.
Stack Object:	Able to stack the selected tile onto another tile (Up to two tiles).
Collision Model:	Different collision types, choose from: *Polygon*, *Box*, *Circle*, *Edge* or *None*.
Collumns and Rows:	Amount of X Collumns and Y Rows.
Tile Width & Heigth:	The size of each grid tile.
Grid Color:	Color of the grid.
Tile Preview:	Toggle to show a rectangular preview at your mouse position within the grid.

Palette

Brush Tile:	Right click inside the grid or hold Space to spawn tiles.
Delete Tile:	Select one or multiple tiles and right click to delete them.
Copy Tile:	Select one tile and right click to copy it.
Reset Tile:	Select one or multiple tiles and right click to reset their scale and rotations.

More information can be found hovering upon the tile summary elements.

The good part

Now when you press on a tile button in the editor window, your current selected tile updates. Whenever you position your mouse over an empty grid space, press on the 'A' key to switch to your building brush and right click to build your selected tile. Press the 'D' or Delete key to switch to your deletion brush. Select one or multiple tiles you want to delete and right click to delete your tiles. Press the 'C' key to switch to your copy brush, select a brush and right click to copy it. Once copied, the brush automatically switches back to the Creation brush and you can directly create the copied tile. Press the 'R' key to switch to your Reset brush. Select one or multiple tiles and right click onto your tiles to reset their localScale and rotation back to their default settings. Press Ctrl + Z to undo your action. You are also able to select tiles in your scene and drag them to other grid spaces while snapping!

Notes:
- The default keys of 'A', 'D', 'C' and 'R' can be changed in script. Go to the GridWindow.cs script and modify the following lines of code to whatever you like:

```
private KeyCode buildingKey = KeyCode.A;
private KeyCode deletingKey = KeyCode.D;
private KeyCode copyKey = KeyCode.C;
private KeyCode resetKey = KeyCode.R;
```

- **Important note:** Edit the scripts with caution. Please do not edit scripts while in the process of building levels to prevent unwanted errors / behaviour! Whenever you have edited the scripts, you will have to re-open your editor window for changes to affect.

Let's Play!

Now that you have learned the basics, you are able to create all sorts of levels. This guide seems like a lot of information, but after a couple of tries you will get used to it quickly.

Go on and create your own level, press the play button, and let your character be free!
A quick little level: http://i.imgur.com/fe4onXV.png. Notice the low drawcalls, great for mobile!

Credits!

Thanks to Kenney Vleugels for his awesome sprites included in this package. More of his sprites can be found below at.
http://opengameart.org/content/platformer-art-complete-pack-often-updated

Imported summed up notes:

- Page 6: Pixels to Units should be the width/height of your PNG texture minus 1.
- Page 7: Make sure you do not have duplicated names for your tiles if you wish to rename their default names.
- Page 10: Important note: Edit the scripts with caution. Please do not edit scripts while in the process of building levels to prevent unwanted errors / behavior! Whenever you have edited the scripts, you will have to re-open your editor window for changes to affect.