ZVEDZA
SOFTWARE

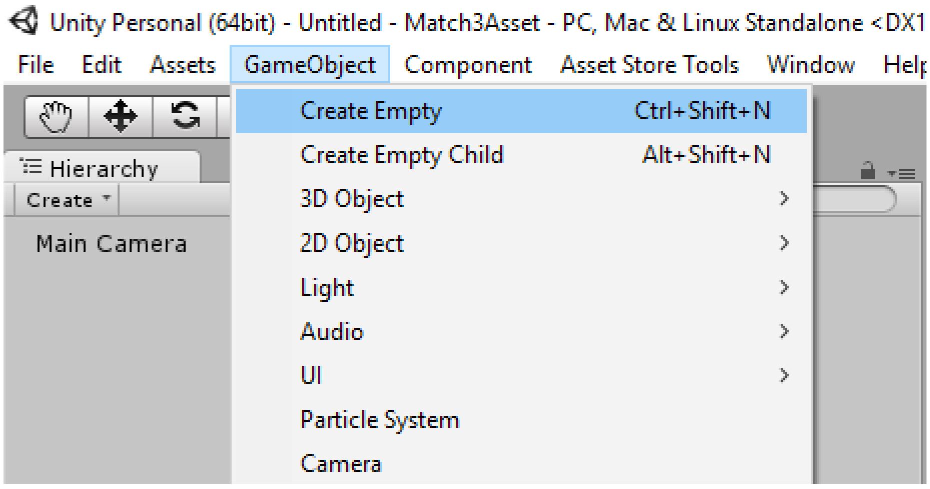
Match 3 – Starter Kit

Quick Start guide

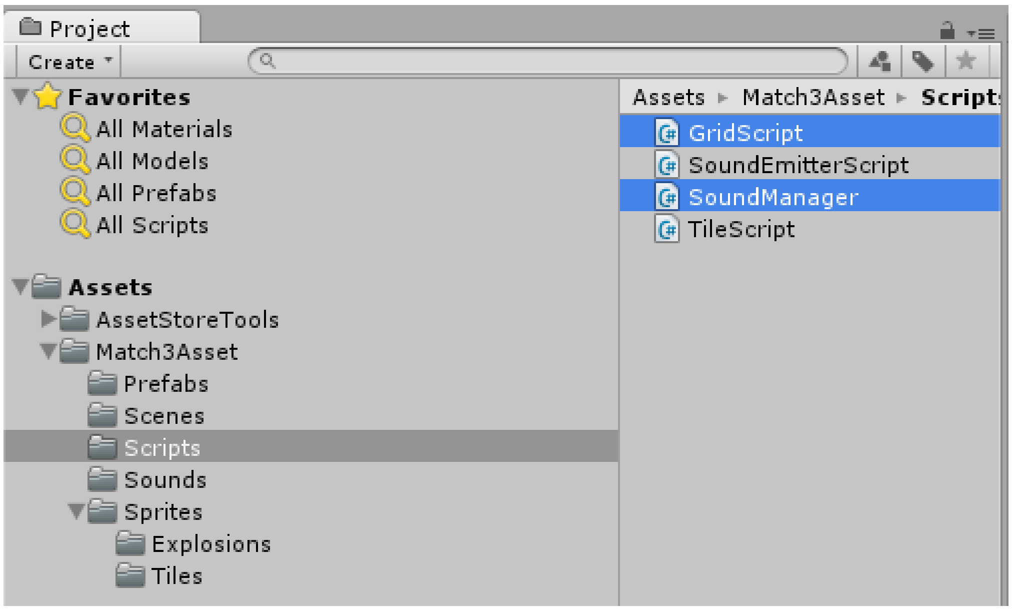
First steps

First thing we need is a blank scene, the project has the scene “PlayScene” by default, this scene is “ready to play”. You can create a new scene by navigating to **File>New Scene**, every unity project starts with a blank scene only containing a Main Camera GameObject. Remember to save the scene.

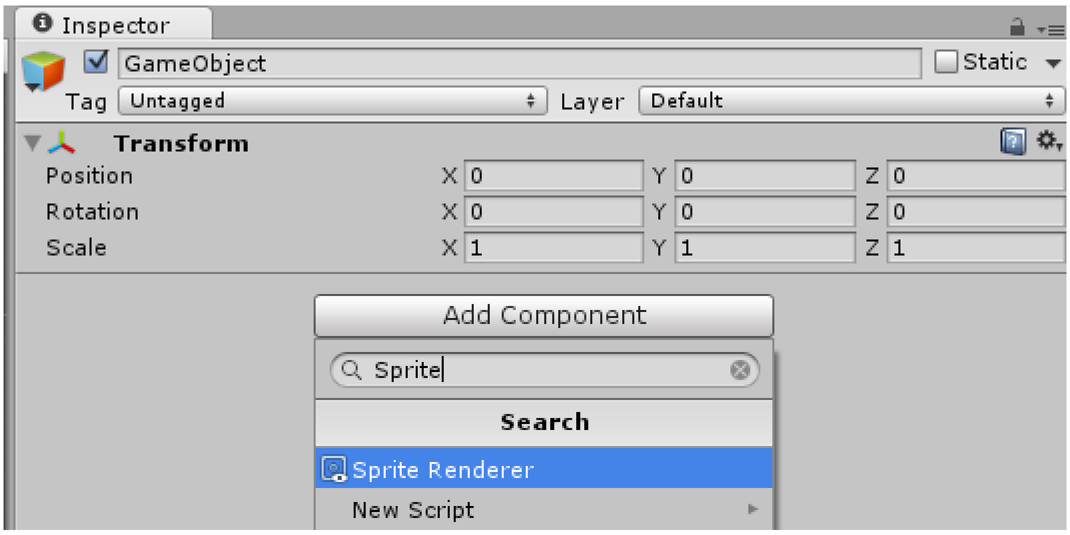
Open a scene and create a new GameObject (**GameObject>Create Empty**), we will be naming it “Grid”.



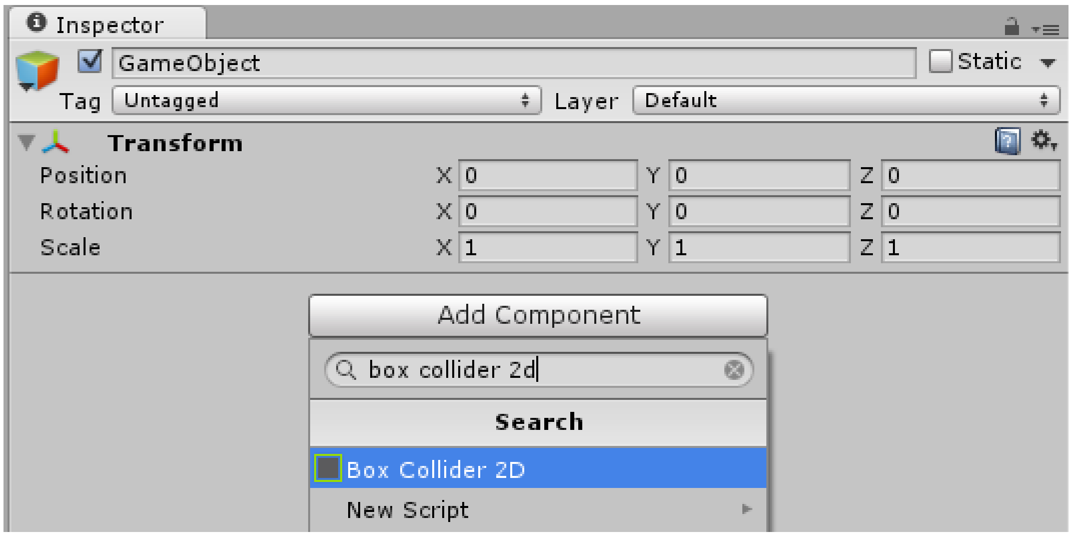
The **Grid** gameobject needs 2 scripts to work: **GridScript** and **SoundManager** both are in the folder **Assets>Match3Assets>Scripts**.



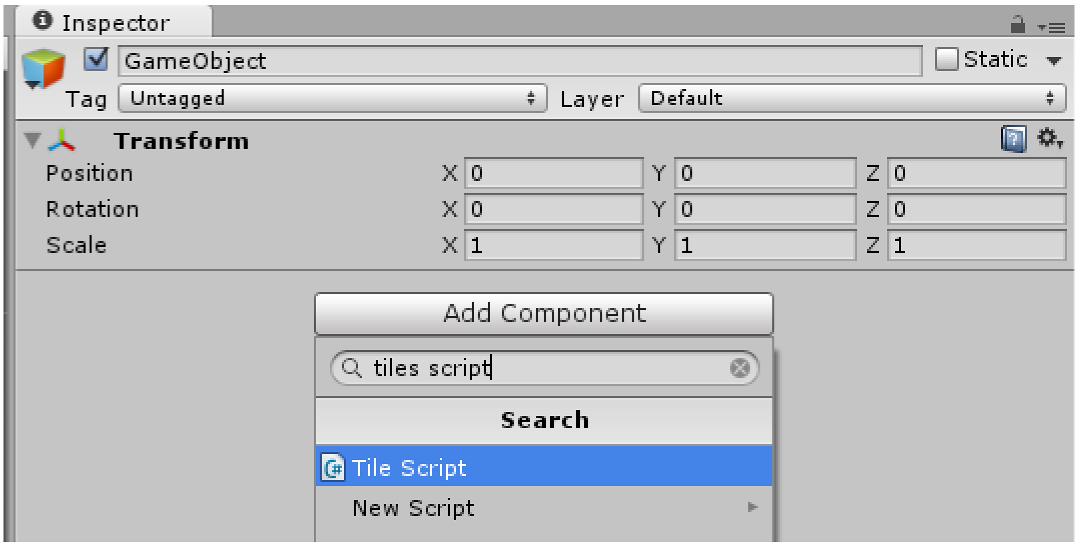
We also need **Tiles**, at least 2 tiles to be able to play and to configure the Grid Gameobject; the tiles are prefabs made from a Gameobject containing A **Sprite Renderer**, A **Box Collider 2D** and a **TileScript** (found in **Assets>Match3Assets>Scripts**). Remember to reset the values of the **transform** component. You can name this GameObject like you want.



GameObject
Tag Untagged Layer Default
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite



GameObject
Tag Untagged Layer Default
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
box collider 2d

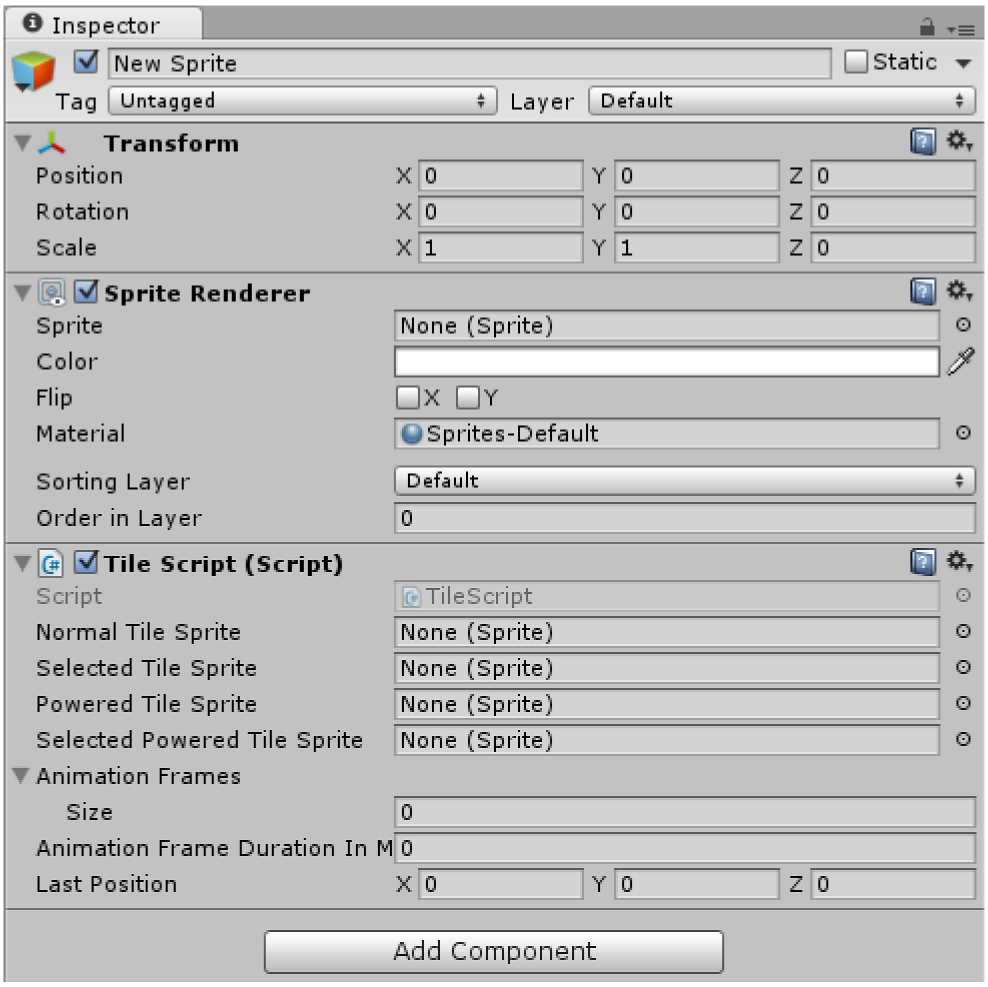


GameObject
Tag Untagged Layer Default
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
tiles script

For the tiles we need **4 sprites** for the tiles and a secuence of sprites to make an exploding animation when a match is done (**the animation is not mandatory, the 4 sprites are**).

We provide some sprites for the tiles, they were made by **Kenney Game Studio**. Check their web site at: http://kenney.nl, they have more assets for you to use. The animations were made by **lumimae**, downloaded from: http://opengameart.org/content/candy-pack-1

Once our scripts are set, we should have something like this:



Inspector

New Sprite — Static

Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	0

Sprite Renderer

Sprite	None (Sprite)
Color	
Flip	X Y
Material	Sprites-Default
Sorting Layer	Default
Order in Layer	0

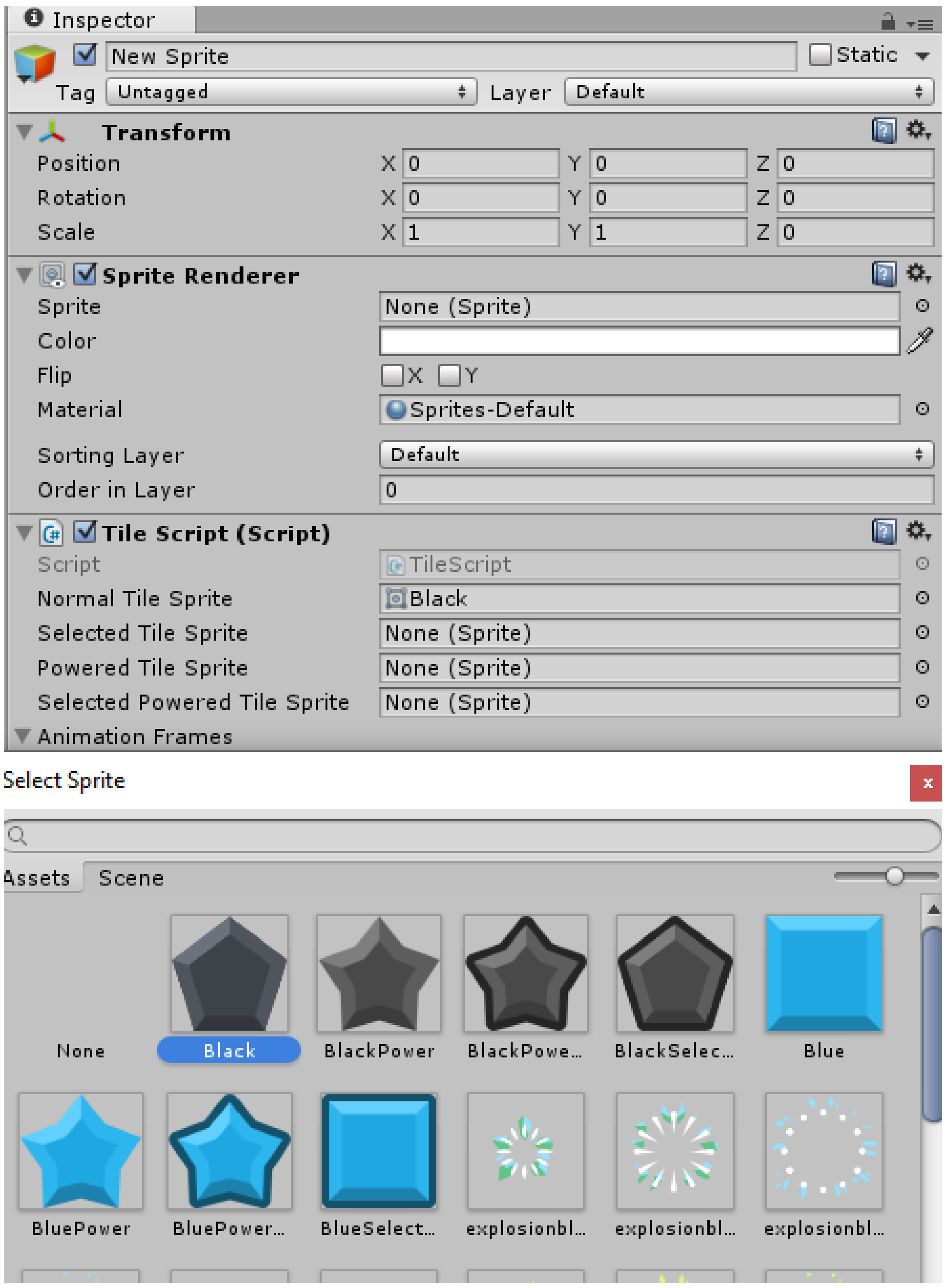
Tile Script (Script)

Script	TileScript
Normal Tile Sprite	None (Sprite)
Selected Tile Sprite	None (Sprite)
Powered Tile Sprite	None (Sprite)
Selected Powered Tile Sprite	None (Sprite)
Animation Frames	
Size	0
Animation Frame Duration In M	0
Last Position	X 0 Y 0 Z 0

The field Sprite of Sprite is not important, you can leave it blank.

The field on the **Tile Script** Component are more important, **you shouldn't leave them blank**. Pick a Sprite for the 4 fields:

- Normal Tile Sprite: is the default sprite.
- Selected Tile Sprite: is the sprite when the tile is selected
- Powered Tile Sprite: is the sprite of the tile when a power is available
- Selected Powered Tile Sprite: is the sprite when the tile has a power and is selected


Inspector

New Sprite — Static

Tag: Untagged — Layer: Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 0

Sprite Renderer

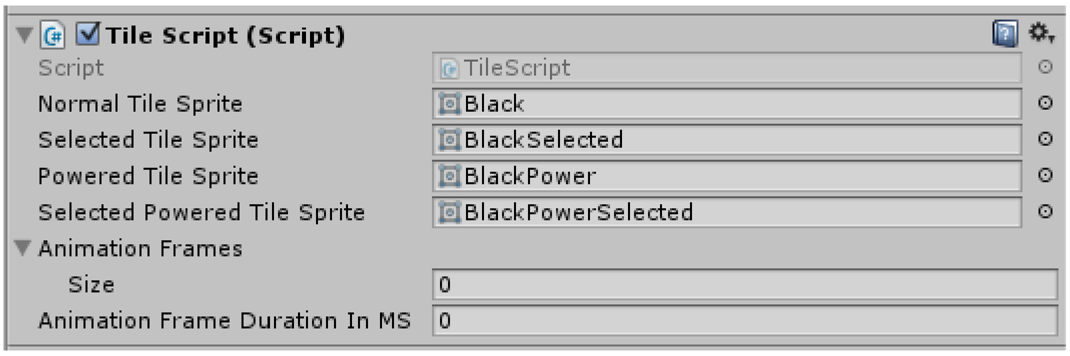
Sprite	None (Sprite)
Color	
Flip	X Y
Material	Sprites-Default
Sorting Layer	Default
Order in Layer	0

Tile Script (Script)

Script	TileScript
Normal Tile Sprite	Black
Selected Tile Sprite	None (Sprite)
Powered Tile Sprite	None (Sprite)
Selected Powered Tile Sprite	None (Sprite)

Animation Frames

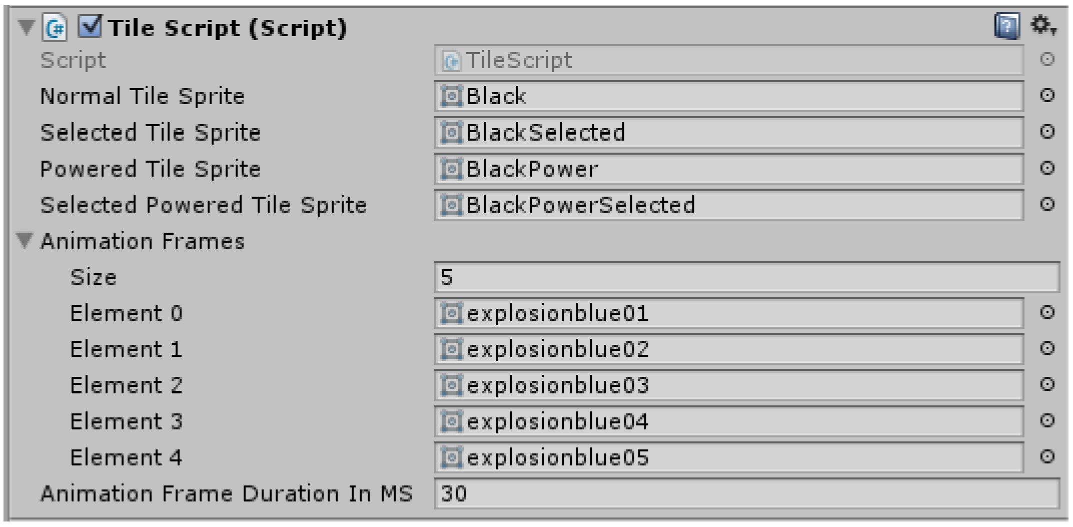
This is how should look the filled fields:



Tile Script (Script)	
Script	TileScript
Normal Tile Sprite	Black
Selected Tile Sprite	BlackSelected
Powered Tile Sprite	BlackPower
Selected Powered Tile Sprite	BlackPowerSelected
Animation Frames	
Size	0
Animation Frame Duration In MS	0

Now we have to configure the animation, the **Size** field is the number of frames of the animation the **Animation Frame duration in ms** is self explanatory, the duration of every frame of the animation in milliseconds, 30 is a good number.

This is how it should looks with everything filled:



Tile Script (Script)	
Script	TileScript
Normal Tile Sprite	Black
Selected Tile Sprite	BlackSelected
Powered Tile Sprite	BlackPower
Selected Powered Tile Sprite	BlackPowerSelected
Animation Frames	
Size	5
Element 0	explosionblue01
Element 1	explosionblue02
Element 2	explosionblue03
Element 3	explosionblue04
Element 4	explosionblue05
Animation Frame Duration In MS	30

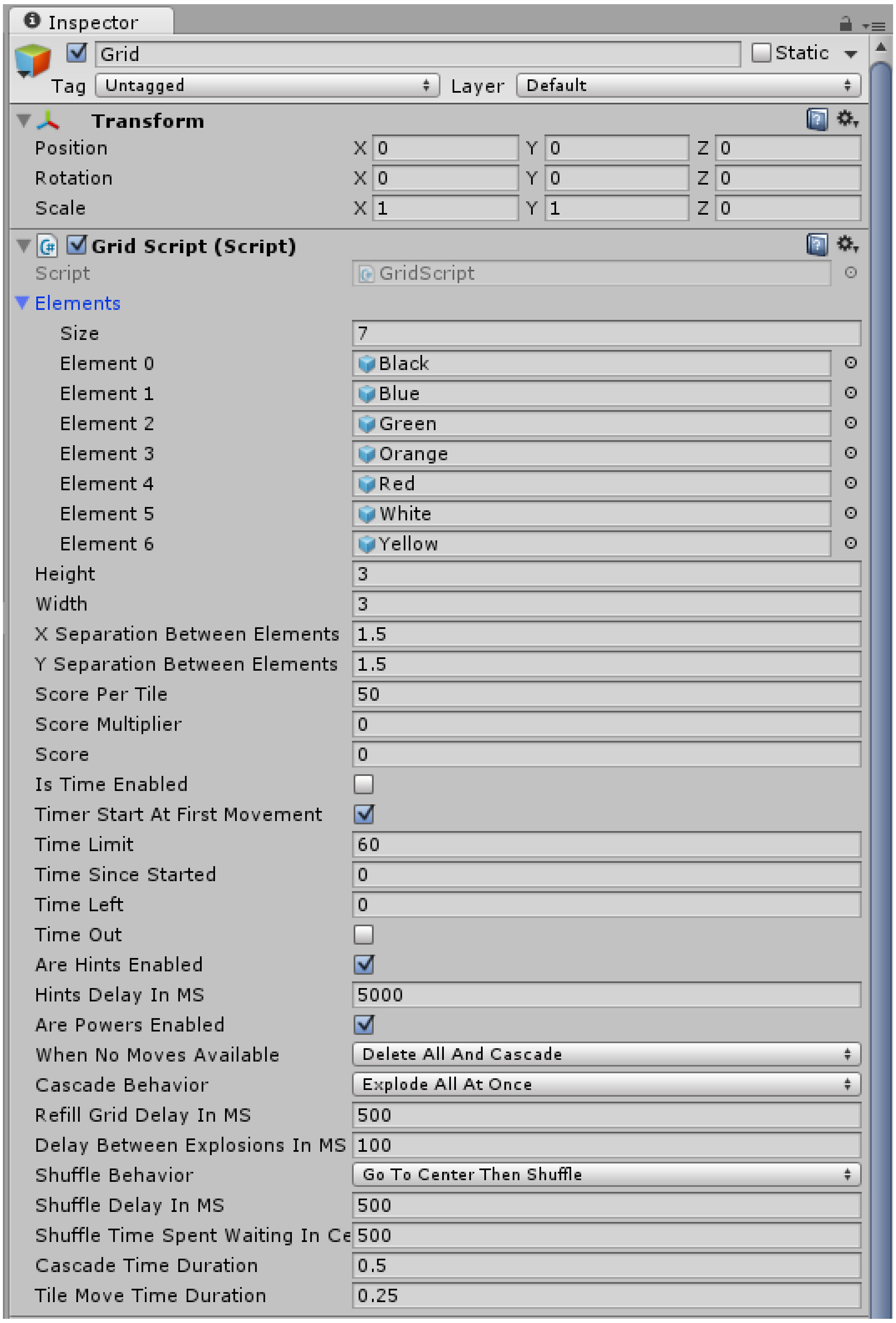
the values for the Box Collider 2D are default in every field but in the Size field, They should be a little bigger than the four basic sprites, it's 1.3 for "X" and "Y" using the provided sprites.

When all the steps were done, **you have to drag the GameObject to a folder in the project view to save it as a prefab**, rinse and repeat, having two of this GameObjects is mandatory for the **GridScript** to work.

Setting up the Grid

After doing all the things above, you should be able to configure the grid

This is a fully configured GridScript:



Inspector

Grid — Static ☐

Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	0

Grid Script (Script)

Script	GridScript
Elements	
Size	7
Element 0	Black
Element 1	Blue
Element 2	Green
Element 3	Orange
Element 4	Red
Element 5	White
Element 6	Yellow
Height	3
Width	3
X Separation Between Elements	1.5
Y Separation Between Elements	1.5
Score Per Tile	50
Score Multiplier	0
Score	0
Is Time Enabled	☐
Timer Start At First Movement	✓
Time Limit	60
Time Since Started	0
Time Left	0
Time Out	☐
Are Hints Enabled	✓
Hints Delay In MS	5000
Are Powers Enabled	✓
When No Moves Available	Delete All And Cascade
Cascade Behavior	Explode All At Once
Refill Grid Delay In MS	500
Delay Between Explosions In MS	100
Shuffle Behavior	Go To Center Then Shuffle
Shuffle Delay In MS	500
Shuffle Time Spent Waiting In Ce	500
Cascade Time Duration	0.5
Tile Move Time Duration	0.25

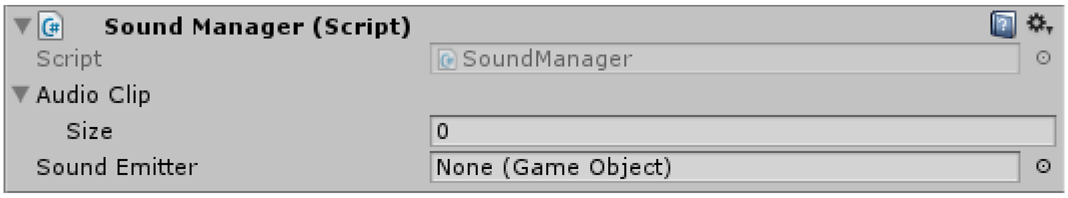
Almost Everything in the Grid Script's fields shown in the inspector is self explanatory.

Now we will explain the not so self explanatory and the important fields in the script:

- **Elements**: the Tiles prefabs we make previously, the minimum number is 2 (you can't match a single tile type), set the size of the list and then drag the prefabs to the fields.
- **Height** and **Width**: the number of vertical and horizontal tiles of the grid.
- **X/Y separation between tiles**: the horizontal and vertical separation between tiles, a bigger sprite needs a bigger number.
- **Score per tile**: the points awarded when a match is done, every tile in the match awards this (a default match of 3 tiles yield 150 points if the field is set at 50).
- **Score Multiplier**: when a chain reaction is triggered, this value increases by one and multiplies the Score per tile (You match 3 tiles and a chain reaction is triggered, if the score per tile is 50, you'll get 300 points for a chain reaction with three tiles involved), <u>you don't have to change this field</u>.
- **Is Time Enabled**: If this is checked, the game would have a time limit, unchecked means unlimited time.
- **Time Starts at first movement**: when checked, the time will start counting when the first move is made, **Is Time Enabled** is needed for this to work.
- **Time Out**: this gets checked when the time limit has been reached disabling the user to make more moves, <u>you don't have to change this field</u>.
- **Are Hints Enabled**: Enables hints when the user is taking too long to make a match/move.
- **Hints Delay In MS**: The Time in milliseconds the system will wait the user before showing a hint, <u>Are Hints Enabled has to be enabled for this to work.</u>
- **Refill Delay In Ms**: time it takes for the grid to replace the elements when no more moves are available.
- **Delay Between Explosions in MS**: time between explosions when the cascade behaviour is set to explode.
- **Shuffle delay in ms:** Time it takes to the system to start shuffling tiles.
- **Tile Move time duration**: Time it takes to a tile to change position.

Sound Manager (Some coding required):

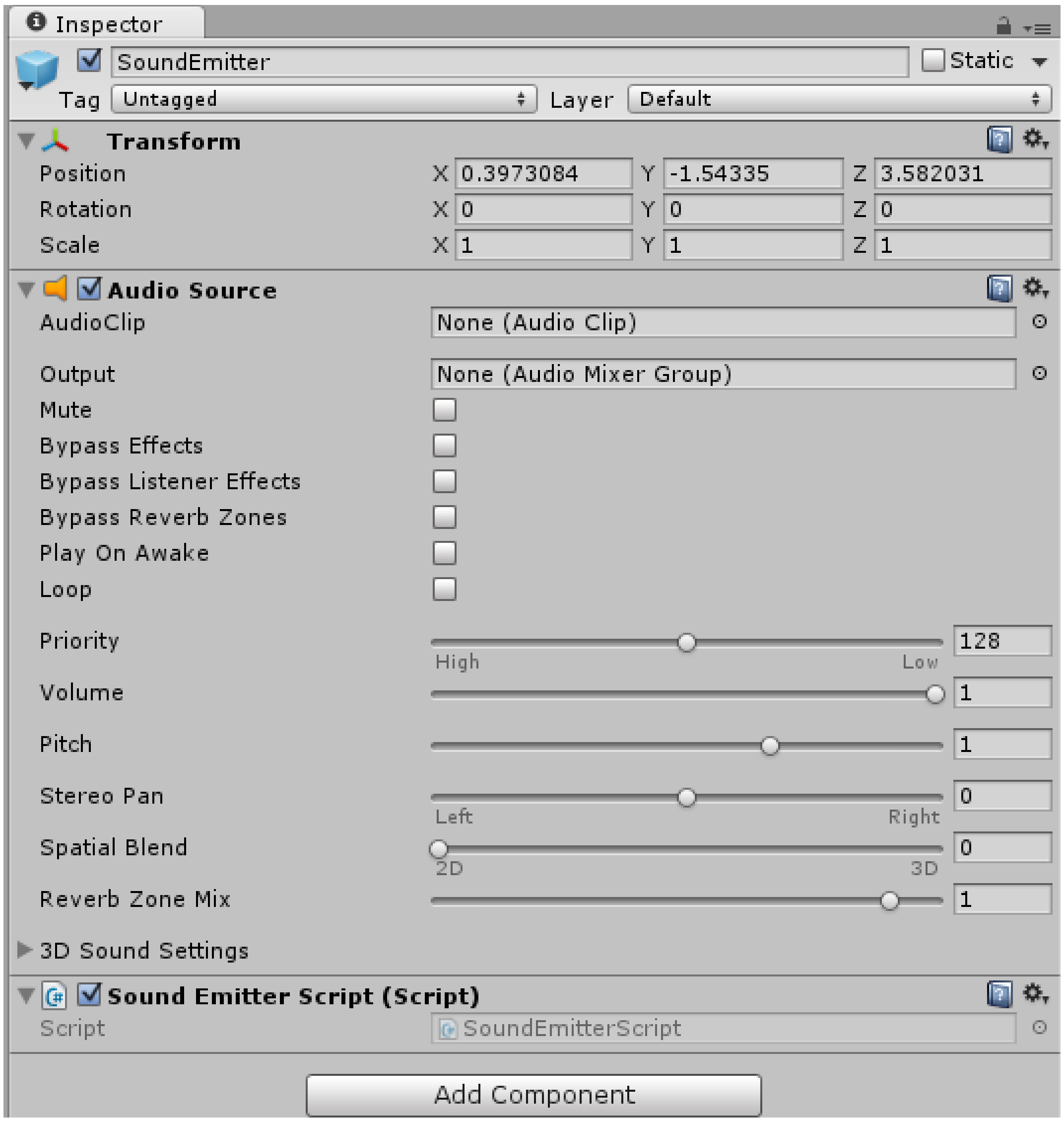
the sounds in the project are managed by a script called Sound manager, this is how it looks in the inspector:



Sound Manager (Script)

Script	SoundManager
Audio Clip	
Size	0
Sound Emitter	None (Game Object)

In the **size field** you should write the number of audio clips you want to manage (we will be using 2 for the guide purposes). The **Sound Emitter Field** Needs a prefab GameObject with a Sound Emitter Component and a **Sound Emitter Script** as seen in the following image, the script is in **Assets>match3Assets>Scripts** folder:



Inspector

SoundEmitter ☐ Static

Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0.3973084	-1.54335	3.582031
Rotation	0	0	0
Scale	1	1	1

Audio Source

AudioClip	None (Audio Clip)
Output	None (Audio Mixer Group)
Mute	☐
Bypass Effects	☐
Bypass Listener Effects	☐
Bypass Reverb Zones	☐
Play On Awake	☐
Loop	☐
Priority (High – Low)	128
Volume	1
Pitch	1
Stereo Pan (Left – Right)	0
Spatial Blend (2D – 3D)	0
Reverb Zone Mix	1

3D Sound Settings

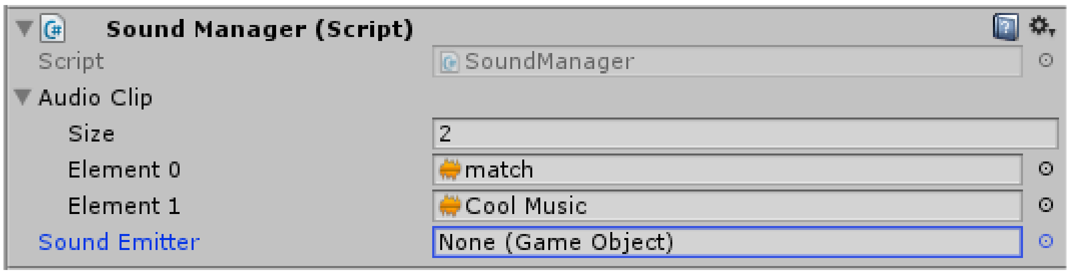
Sound Emitter Script (Script)

Script	SoundEmitterScript

Add Component

be sure to set the "Spatial blend" option to 0.

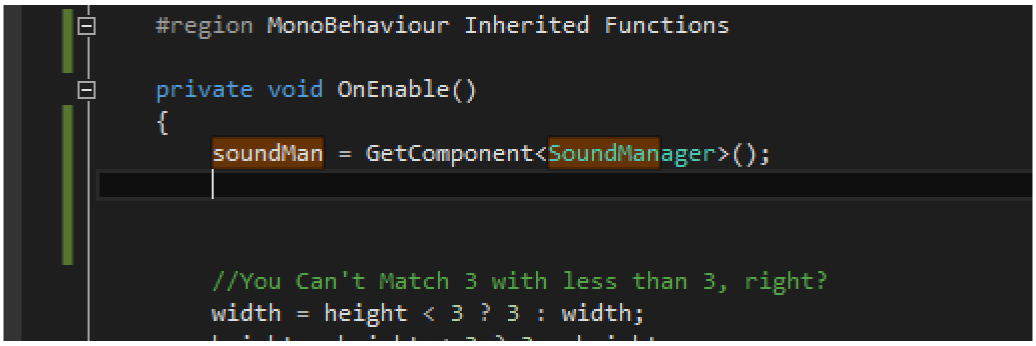
For the Sound manager script you have to fill the size field and then drag the audio files to the boxes.



Sound Manager (Script)	
Script	SoundManager
Audio Clip	
Size	2
Element 0	match
Element 1	Cool Music
Sound Emitter	None (Game Object)

Now you will need to edit the GridScript, we chose to play a tune at the start of the game:

First, locate the portion of the code where you want a sound to play, we chose the OnEnable() function just after the initialization of the sound manager **soundMan should be initialized for this to work**:



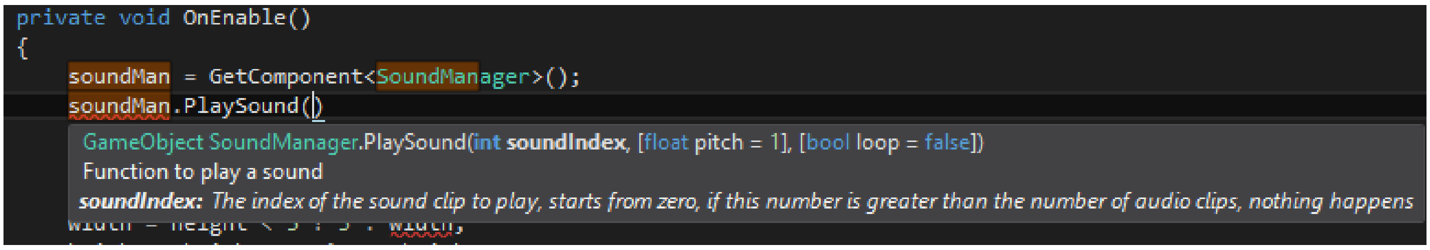
```
#region MonoBehaviour Inherited Functions

private void OnEnable()
{
    soundMan = GetComponent<SoundManager>();



    //You Can't Match 3 with less than 3, right?
    width = height < 3 ? 3 : width;
```

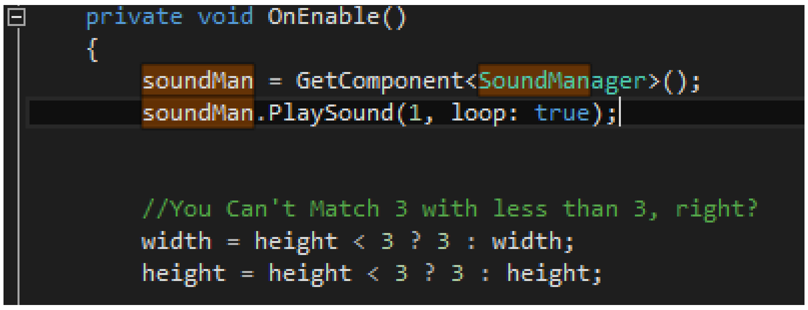
now we write the function soundman.PlaySound(), this function takes 3 arguments, the first one is the number of the element on the Audio Clip Array (see second picture above), the second is the pitch, it ranges between -3 and 3, the last one is the loop boolean, set it to true if you want the music to loop.



```
private void OnEnable()
{
    soundMan = GetComponent<SoundManager>();
    soundMan.PlaySound()
```

GameObject SoundManager.PlaySound(int **soundIndex**, [float pitch = 1], [bool loop = false])
Function to play a sound
soundIndex: *The index of the sound clip to play, starts from zero, if this number is greater than the number of audio clips, nothing happens*

We wanted to play some Cool Music at the start, so we wrote “1” (the “Match“ sound is “0”) in the first field, pitch and loop are optional fields, if you want to use them you have to write <<nameOfField: valueOfField>> as seen in the next image, we used the optional field loop:



```
private void OnEnable()
{
    soundMan = GetComponent<SoundManager>();
    soundMan.PlaySound(1, loop: true);


    //You Can't Match 3 with less than 3, right?
    width = height < 3 ? 3 : width;
    height = height < 3 ? 3 : height;
```

If you need help with this project you can reach us at zvedzasoftware@gmail.com