Do it with Unity Assets

Mezanix Fx Free

A tutorial of using the Unity Mezanix Fx Free Package

Table of contents

Introduction

NB. In addition to the effects, Mezanix Fx Free contains a set of scripts (explained in the section: Mezanix Fx Free folders structure and scripts explanation). These scripts perform bullets and movement behaviors that are helpful in many of your projects.

Mezanix Fx Free is a **diverse** pack of **particle system effects**, actually it contains 15 effects, accompanied by their **sound effects**. By this pack I offer a Fx solution for all game types (action, rpg, fantasy, casual and more).

Actually Mezanix Fx Free contains 15 effects. Its aim to show you the philosophy behind Mezanix Fx paid version.

The paid version contains actually 28 effects, that number will increase constantly for every update. It cost now 15 US $. This price will be increased by increasing the effects number. By buying it now you will get all new effects of newer versions for free, because downloading updates will be free of charge.

How to put 'Particle System' prefabs in the scene

You can use any 'Particle System' in 3 ways: directly, using the 'ParticleSystemSpawner.cs' script and using the 'MultiParticleSystemSpawner.cs' script.

Way 1, directly:

Drag and drop the 'Particle System' prefab in your scene.

'Particle system' prefabs are in the folder:
Assets\Mezanix\MezanixFX\1_ParticleSystems

Way 2, using the ParticleSystemSpawner.cs script:

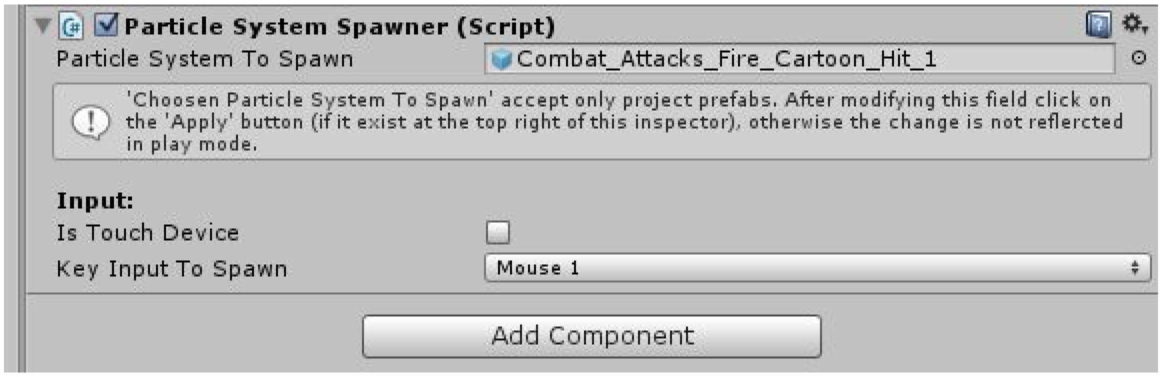
You can perform this in 4 steps:

Step 1: In the scene, create a gameobject, it's recommended that this gameobject contains a collider component.

Step 2: To the gameobject created in the step 1, add the script 'ParticleSystemSpawner.cs'. The script ParticleSystemSpawner.cs is in the folder: **Assets\Mezanix\MezanixFX\2_Scripts\2_Spawner**

Step 3: In the component 'ParticleSystemSpawner' there is a field called '**Particle System To Spawn**' (Image.1). To this field add a 'Particle system' prefab. 'Particle system' prefabs are in the folder: **Assets\Mezanix\MezanixFX\1_ParticleSystems**

Image.1



Particle System Spawner (Script)

Particle System To Spawn: Combat_Attacks_Fire_Cartoon_Hit_1

'Choosen Particle System To Spawn' accept only project prefabs. After modifying this field click on the 'Apply' button (if it exist at the top right of this inspector), otherwise the change is not reflercted in play mode.

Input:

Is Touch Device: ☐

Key Input To Spawn: Mouse 1

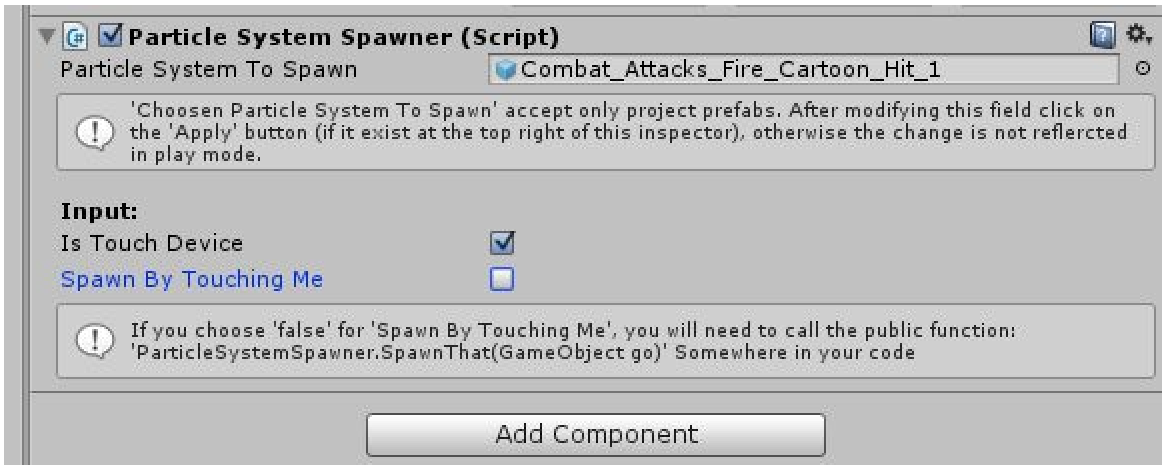
NB. When you add a 'Particle system' prefab to the field 'Particle System To Spawn', the name of the 'ParticleSystemSpawner' gameobject change to: ‘Pss_particleSystemPrefabName', it's helpful because it's indicate directly which 'Particle System' is contained in the 'ParticleSystemSpawner'.

Step 4: In the component 'ParticleSystemSpawner' there is a check field called '**Is Touch Device'** (Image.1). You can choose between 2 cases:

Case 1: your target isn't a touch device, so you let this field unchecked, in this case, you have to choose which keyboard or mouse input will spawn the 'Particle System' prefab, you can do that by changing the value of the field **'Key Input To Spawn**', by default this field is set to 'Mouse 1' that's mean 'Right Mouse Click'.

Case 2: your target is a touch device, so you check on the field **'Is Touch Device'** (Image.2), in this case you have to choose between 2 options:

Image.2



Particle System Spawner (Script)

Particle System To Spawn: Combat_Attacks_Fire_Cartoon_Hit_1

'Choosen Particle System To Spawn' accept only project prefabs. After modifying this field click on the 'Apply' button (if it exist at the top right of this inspector), otherwise the change is not reflercted in play mode.

Input:

Is Touch Device: ✓

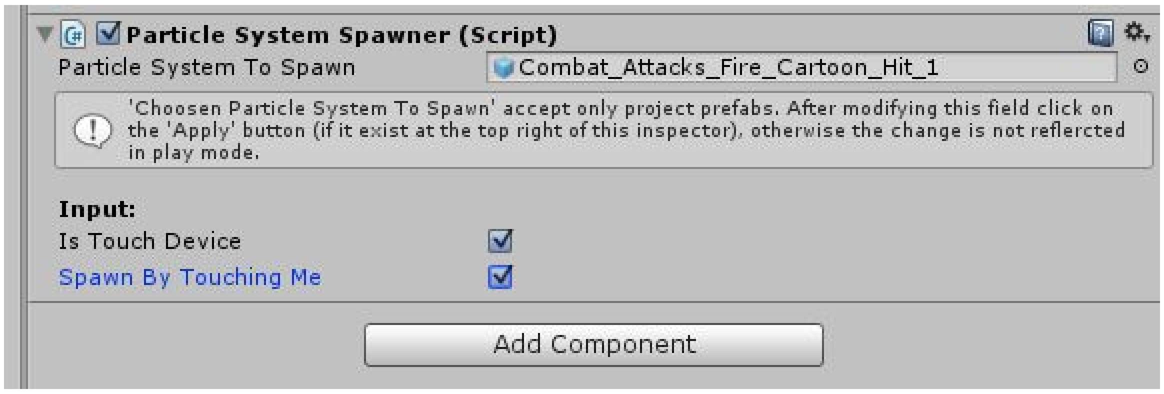
Spawn By Touching Me: ☐

If you choose 'false' for 'Spawn By Touching Me', you will need to call the public function: 'ParticleSystemSpawner.SpawnThat(GameObject go)' Somewhere in your code

Option 1: Spawn the 'Particle System' prefab by touching the 'ParticleSystemSpawner' game object, so you have to check on the field '**Spawn By Touching Me**' (Image.3).

NB. In this option, the 'ParticleSystemSpawner' game object must have a collider component.

Image.3



Particle System Spawner (Script)

Particle System To Spawn: Combat_Attacks_Fire_Cartoon_Hit_1

'Choosen Particle System To Spawn' accept only project prefabs. After modifying this field click on the 'Apply' button (if it exist at the top right of this inspector), otherwise the change is not reflercted in play mode.

Input:

Is Touch Device: ✓

Spawn By Touching Me: ✓

Option 2: Let the field '**Spawn By Touching Me**' unchecked (Image.2), in this option you have to call the the public function: 'ParticleSystemSpawner.SpawnThat(GameObject go)' Somewhere in your code.

NB. 'ParticleSystemSpawner.SpawnThat(GameObject go)' is not static, to call it you have to create in your code, a variable of type 'ParticleSystemSpawner' and "put in" a gameobject containing a 'ParticleSystemSpawner' component.

Way 3, using the MultiParticleSystemSpawner.cs script:

You can perform this in 6 steps:

Step 1: In the scene, create a gameobject. An empty game object can do the job.

Step 2: To the game object created in the step 1, add the script 'MultiParticleSystemSpawner.cs'. The script MultiParticleSystemSpawner.cs is in the folder: **Assets\Mezanix\MezanixFX\2_Scripts\2_Spawner**

Step 3: To the game object created in the step 1, add several childs.

Step 4: To each child add the component 'ParticleSystemSpawner' like explained in the 'Way 2'.

Step 5: Now play the scene, you will notice that only the first child (ParticleSystemSpawner) is active in the scene.

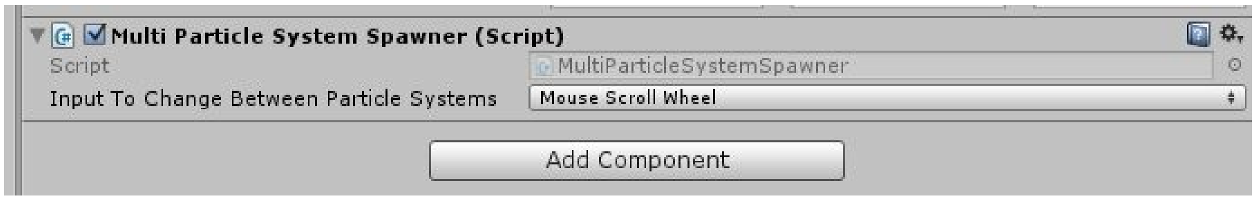
Step 6: In the 'MultiParticleSystemSpawner' there is a field called **Input To Change Between ParticleSystems** (Image.4), this field define the input to use if we want to change the active child (ParticleSystemSpawner). This field can have one of 3 options:

Option 1: mouse Scroll Wheel, using the mouse wheel to change.

Option 2: up Down Arrows, using up and down arrows to change.

Option 3: Gui, create a ui button, at the 'On Click' field add an action, in this action you have to add the game object containing the component 'MultiParticleSystemSpawner', after that, choose the function MultiParticleSystemSpawner.UpperParticleSystem, like so when the button is clicked the upper child (ParticleSystemSpawner) is activated in the child's list. If you want a button to activate the lower child (ParticleSystemSpawner) in the child's list, you will have to create a new button, add to it an action (at On Click field), this action should point to the function MultiParticleSystemSpawner.LowerParticleSystem. Gui option is helpful for touch devices.

Image.4



Multi Particle System Spawner (Script)

Script: MultiParticleSystemSpawner

Input To Change Between Particle Systems: Mouse Scroll Wheel

Mezanix Fx Free folders structure and scripts explanation

Mezanix Fx contains 7 folders

0_Documentation

1_ParticleSystems

2_Scripts

3_Sounds

4_Materials

5_Textures

Editor

Folders names are explicit and indicate the nature of each folder, so I will explain only necessary things.

The folder **1_ParticleSystems** hold all Particle system prefabs of the package, it contains 5 sub-folders:

1_Combat

2_Family

3_Nature

4_Environment

Like the slogan of Mezanix Fx Free said "All game types unity effects", this package tend to offer the most complete and rich Particle System pack. In upcoming versions I will add more categories and extend the existing categories.

The **2_Scripts** folder hold all scripts that define 3 things: the functionalities of effects, their spawners and their interactions with the environment (like collisions and so on).

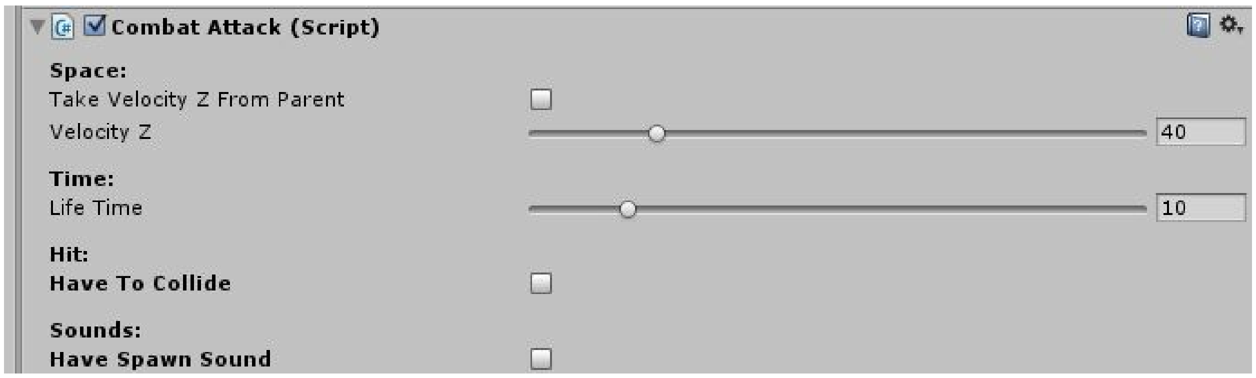
Now I will explain the most important scripts contained in the 2_Scripts folder.

The CombatAttack.cs script

This script is located at:
Assets\Mezanix\MezanixFX\2_Scripts\1_ParticleSystems\1_Combat\1_Attacks

This script is a bullet script, when attached to a game object, it move this gameobject forward (local +Z direction). CombatAttack.cs has the following categories of fields: **Space**, **Time**, **Hit** and **Sounds** (Image.5).

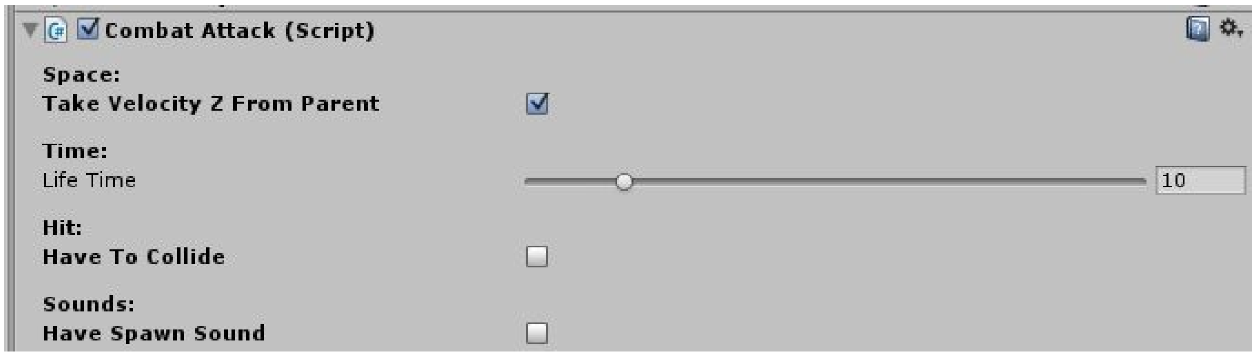
Image.5



```
Combat Attack (Script)
Space:
Take Velocity Z From Parent    [ ]
Velocity Z                     40
Time:
Life Time                      10
Hit:
Have To Collide                [ ]
Sounds:
Have Spawn Sound               [ ]
```

In the Space category we have: **Take Velocity Z From Parent** boolean field and the float field **Velocity Z**. If you check on the **Take Velocity Z From Parent** (Image.6), the gameobject will have the same forward (local) velocity of it’s parent, so it doesn't move if it hasn't a parent, if it has a parent, it will have the forward (local) velocity of its first parent. If you check off the **Take Velocity Z From Parent** (Image.5), the gameobject will move forward (local) by the **Velocity Z** value either it has or not a parent, so when it has a parent it's forward (local) velocity will be added to the forward (local) velocity of its parent (thanks Newton ;))

Image.6



Combat Attack (Script)

Space:
Take Velocity Z From Parent ✓

Time:
Life Time 10

Hit:
Have To Collide ☐

Sounds:
Have Spawn Sound ☐

In the Time category we have the **Life Time** float field (Image.5). For seek of performance, the **Life Time** of the gameobject destroy it when elapsed.

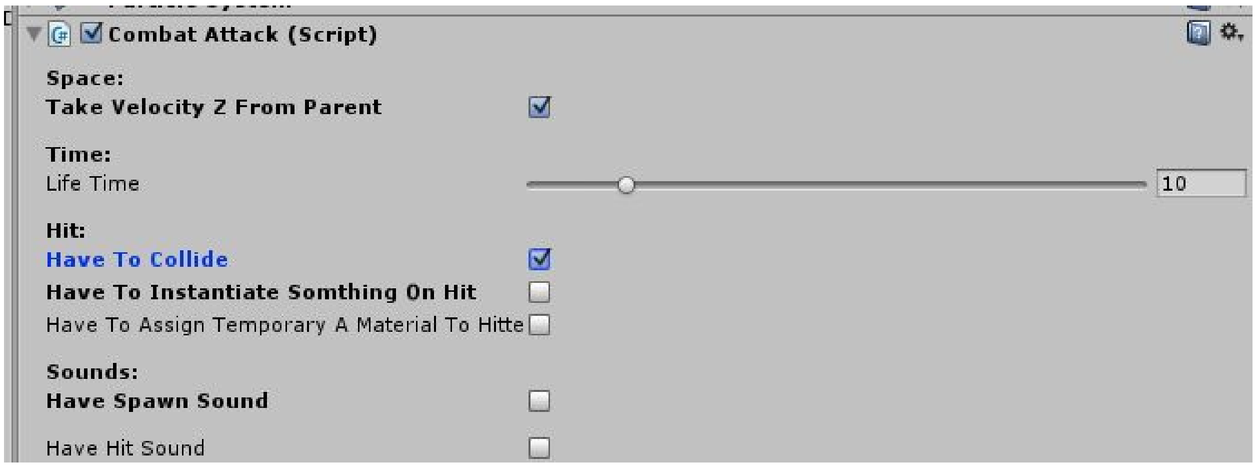
In the Hit category: In this category we have 3 fields: **Have To Collide**, **Have To Instantiate Something On Hit** and **To Instantiate On Hit**.

If **Have To Collide** is false (Image.5), the gameobject doesn't collide with anything (expect when you add yourself on it a collider). Our hit system is based on raycast.

NB. I know that CombatAttack.cs is a bullet script, and naturally a bullet collide, but I tend to do customizable things as possible, for example imagine you have to design a firework scene, you probably don't need collisions in this case.

Now, if **Have To Collide** is true (Image.7), the gameobject is destroyed when colliding, in this case we can choose or not to instantiate something on hit (like an explosion), If we want to instantiate something on hit, we will have to check true the field **Have To Instantiate Something On Hit** and fill the field **To Instantiate On Hit** with a gameobject prefab (Image.8).

Image.7


Combat Attack (Script)

Space:
Take Velocity Z From Parent ✓

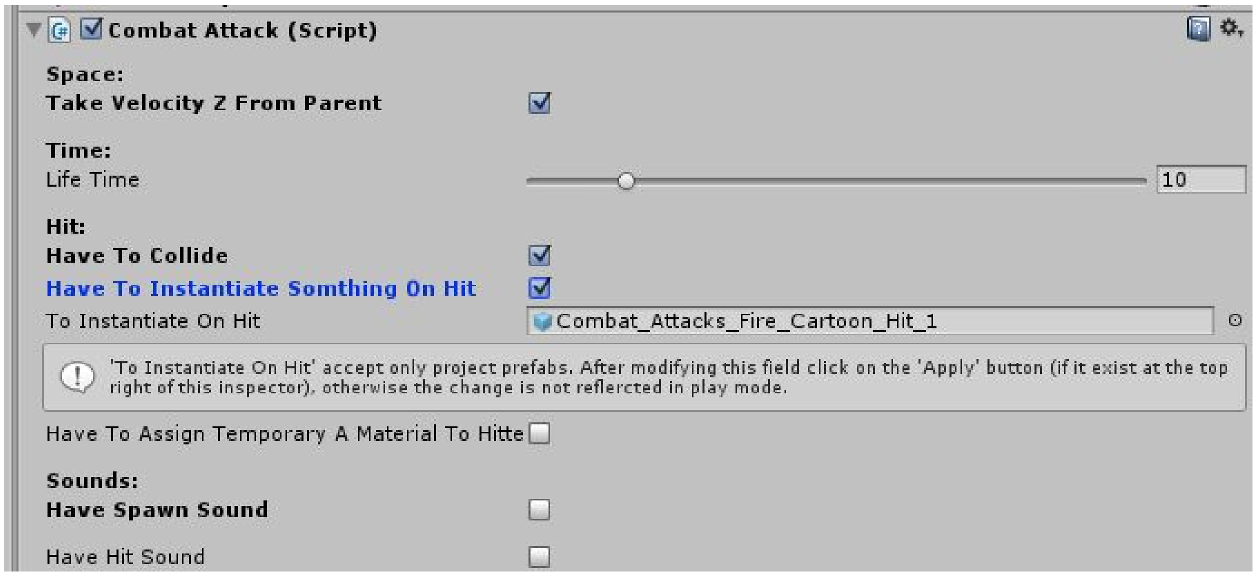
Time:
Life Time 10

Hit:
Have To Collide ✓
Have To Instantiate Somthing On Hit ☐
Have To Assign Temporary A Material To Hitte ☐

Sounds:
Have Spawn Sound ☐

Have Hit Sound ☐

Image.8


Combat Attack (Script)

Space:
Take Velocity Z From Parent ✓

Time:
Life Time 10

Hit:
Have To Collide ✓
Have To Instantiate Somthing On Hit ✓
To Instantiate On Hit Combat_Attacks_Fire_Cartoon_Hit_1

'To Instantiate On Hit' accept only project prefabs. After modifying this field click on the 'Apply' button (if it exist at the top right of this inspector), otherwise the change is not reflercted in play mode.

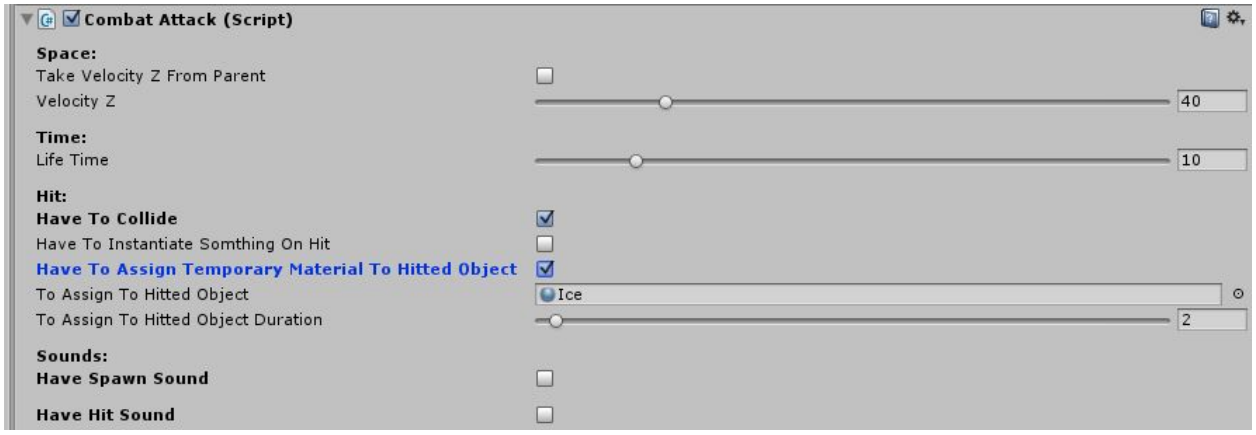
Have To Assign Temporary A Material To Hitte ☐

Sounds:
Have Spawn Sound ☐

Have Hit Sound ☐

Also if **Have To Collide** is true, we can choose or not to assign a temporary material to the hitted object (like a freeze effect for example), in this case, we have to set true the field **Have To Assign Temporary Material To Hitted Object,** then we have to define which material to add to the field **To Assign To Hitted Object**, and finally the duration of this material on the hitted object is defined in the field **To Assign To Hitted Object Duration** (Image.9).

Image.9



Combat Attack (Script)

Space:
Take Velocity Z From Parent ☐
Velocity Z 40

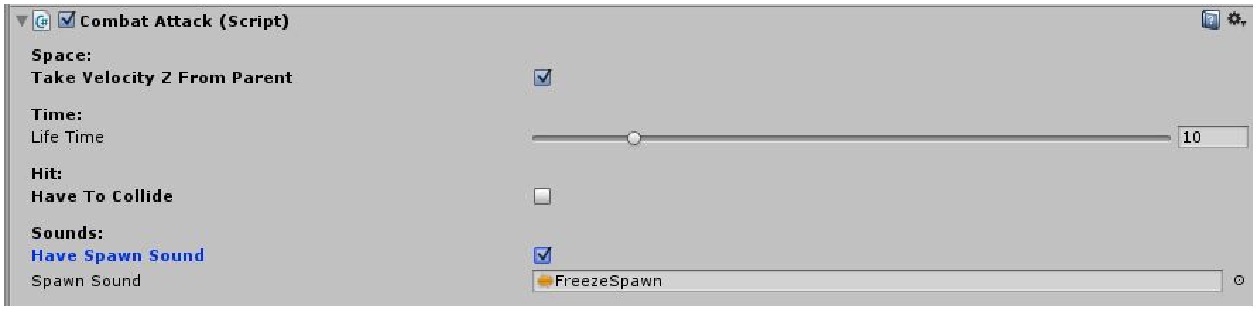
Time:
Life Time 10

Hit:
Have To Collide ☑
Have To Instantiate Somthing On Hit ☐
Have To Assign Temporary Material To Hitted Object ☑
To Assign To Hitted Object Ice
To Assign To Hitted Object Duration 2

Sounds:
Have Spawn Sound ☐
Have Hit Sound ☐

The Sounds category: This category help to use a spawn sound and a hit sound. if the field **Have Spawn Sound** is true, we have to fill the field **Spawn Sound** with an audio clip, this clip is played once at the spawning of the Particle System (Image.10). Similarly, when the field **Have Hit Sound** is on, we have to fill the field **Hit sound** with an audio clip that will be played once when the particle system hit something (Image.11). Hit sound is visible only when the field **Have To Collide** is true.

Image.10



Combat Attack (Script)

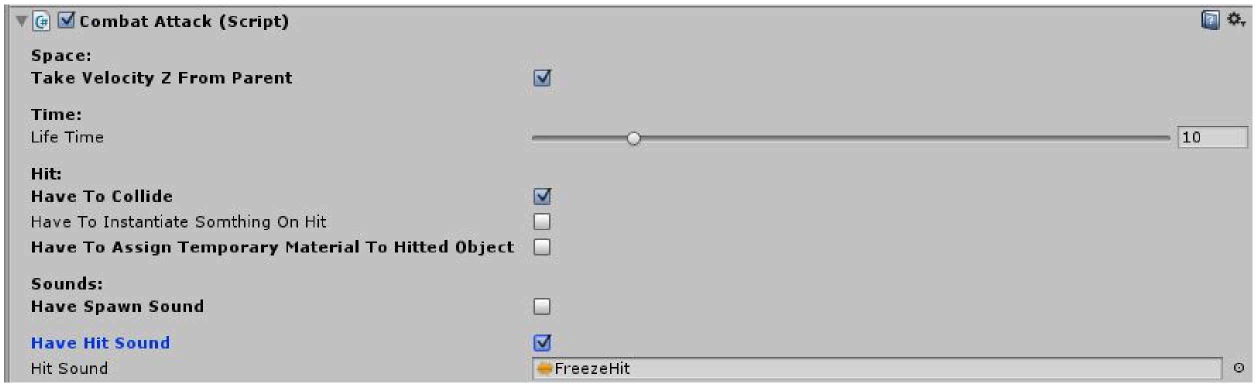
Space:
Take Velocity Z From Parent ☑

Time:
Life Time 10

Hit:
Have To Collide ☐

Sounds:
Have Spawn Sound ☑
Spawn Sound FreezeSpawn

Image.11


Combat Attack (Script)

Space:
Take Velocity Z From Parent ✓

Time:
Life Time 10

Hit:
Have To Collide ✓
Have To Instantiate Somthing On Hit ☐
Have To Assign Temporary Material To Hitted Object ☐

Sounds:
Have Spawn Sound ☐
Have Hit Sound ✓
Hit Sound FreezeHit

NB. The scripts CombatAttacksWithRotationY.cs and CombatAttacksWithRotationZ.cs inherit from CombatAttack.cs, so they are very similar to it, the only difference is that these scripts rotate (around local Z or Y axis). These scripts are located at:
Assets\Mezanix\MezanixFX\2_Scripts\1_ParticleSystems\1_Combat\1_Attacks

The RotatorY.cs and RotatorZ.cs scripts

These scripts are located at:
Assets\Mezanix\MezanixFX\2_Scripts\1_ParticleSystems\2_Family

They simply rotate an object around local Y and Z axis.

The UpDown.cs script

This script is located at:
Assets\Mezanix\MezanixFX\2_Scripts\1_ParticleSystems\2_Family

This script perform an oscillatory movement along the Y axis.

NB. Notice that CombatAttack.cs, CombatAttacksWithRotationY.cs, CombatAttacksWithRotationZ.cs, RotatorY.cs, RotatorZ.cs and UpDown.cs are bullets and movement scripts performing simple translations or rotations, but you can do nice movements when you combine them in the same gameobject.