

Sci-Fi Effects

Version 1.1

Copyright © 2015 FORGE3D

<http://www.forge3d.com>
support@forge3d.com

Contents

1. Introduction
2. Brief overview
3. Getting started
4. Example scenes
5. Scaling particle effects
6. Script reference

Introduction

Thank you for purchasing Sci-Fi Effects!

This guide describes the features of the Sci-Fi Effects integration in Unity3D. A basic understanding of the Unity3D engine, as well as C# programming language is assumed. Having basic knowledge of the shader and visual effects design in Unity3D may be advantageous.

For more information please visit www.forge3d.com

If you have any questions, suggestions, comments or feature request please do not hesitate to contact us at support@forge3d.com

Brief overview

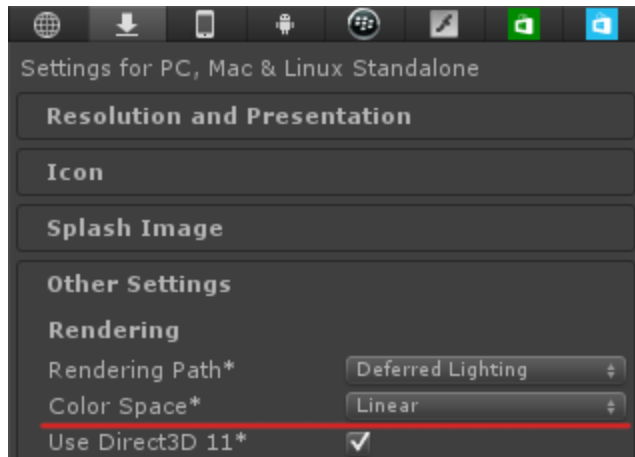
Sci-Fi Effects is a collection of more than 90 ready to use prefabs including complex particle systems, billboards, sounds and helper scripts. Example scenes provided within the package utilize some of the prefabs and demonstrate usage of these effects while all scripts are self explanatory and well documented.

For ease of use this package includes particle scaling script to help you match prefabs scale to your scene scale. See corresponding section of this manual describing particle scaling procedure in better details.

Please note this package is best used with linear color space.

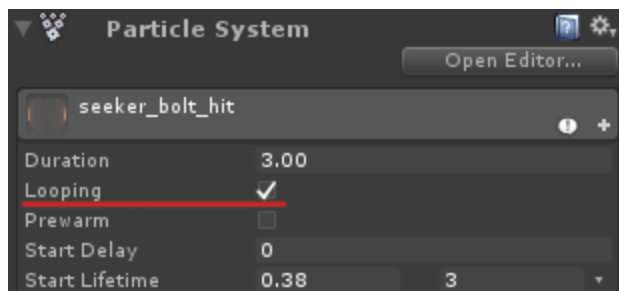
Getting started

First of all make sure you set your project settings to linear color space. If you haven't done so go to **Edit -> Project Settings -> Player** and select Linear using drop down list:



** If you decide to stay in gamma space it may require tweaking color values depending on your personal preferences. Most of the materials are good for using with both color spaces, however a separate copy of a material is provided for cases with significant visual difference.*

Most of the particle effects provided are set to be looped over time for demo purposes. Before using the actual prefabs make sure to uncheck **Looping** flag for the current parent and all of the child particle systems if appropriate:



Example scenes

There are several example scenes included under **Assets/FORGE3D/Sci-Fi Effects/Examples** path:

01_turret_demo - It is important to note, that this scene uses specific copies of prefabs located inside the Example folder.

02_effects_preview - This scene has most of the prefabs came with the initial version of the asset, which are ready to be used in your project. Make sure to disable 'Looping' flag on a particle system when appropriate.

03_burnout_example (gamma / linear) - Check the F3DBurnoutExample script located on the turret for a basic usage example.

04_warp_tunnel - This example uses several cameras technique to render the tunnel and nebula clouds on different layers. The nebula boards are combined with CombineChildren script in runtime which saves a great amount of draw calls. The tunnel can be faded by modifying the `_TintColor` alpha value.

05_holographic_example - The holographic shader settings are self explanatory. The interlaced effect applied by using screen space coordinates.

06_nebula_example - The nebula cloud is faded by using normal to view angle difference, as well as camera distance to board surface. The nebula boards are combined with CombineChildren script in runtime which saves a great amount of draw calls.

07_debris_example - The debris field is faded according to camera distance to its surface to achieve smooth transitions.

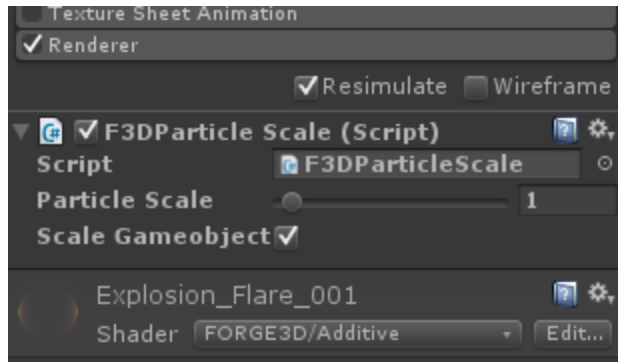
08_warp_jump_example - The warp jump effect uses combination of particles systems and special mesh. Each mesh is scaled at the moment of opening, while its texture is twisted with help of special shader. You can use `_ShipPosition_` game object to match the ship's position with the warp spark moving through the tunnel. The effect can be scaled with help of F3DParticleSystem script.

** Please note that example scripts provided may require additional modifications before they can be used in your project or may not be suited for usage in different environments at all.*

Scaling particle effects

Particle scaling is made possible with use of **F3DParticleSystem** script located at **Assets/FORGE3D/Sci-Fi Effects/Code** path.

To scale a particle system attach this script to a parent object and use the slider to scale the parent and all included child objects at the same time:



Script reference

This section will give you brief details on scripts used by the turret example. It also important to know that most scripts rely on F3DTimer class and require an instance of such to be present in the scene. Please take your time to examine each script more carefully to fully understand what is happening behind the scene.

CombineChildren

Attach this script as a parent to some game object. The script will then combine the meshes at startup. This is useful as a performance optimization since it is faster to render one big mesh than many small meshes.

F3DAudioController

This script is an example of audio management and playback. What it does is playing an audio clip at specific position and modifies various audio settings such as random volume or pitch depending on the method called.

F3DBeam

This script is mainly used for updating beam weapons such as beam laser with uv animation for tiled textures, real time raycasting and interacting with rigid bodies by applying AddForceAtPosition. It is also scales the texture along its length depending on the beam length so it's never gets stretched.

F3DDespawn

This script is used to despawn most of the effects after predefined delay by calling corresponding method of the pool manager that is included in this package.

F3DFlameThrower

This script is used by flame thrower prefab to manage some utility tasks such as fading in/out the lights and despawning the effect.

F3DFXController

This script defines all the weapon types and the way they are spawned such as managing prefab references, rate of fire, invoking specific audio routines and finally the GUI drawing seen in turret example.

F3DLighting

This script is mainly used for updating lightning gun weapon such as updating amount of lightning points and animating the uvs. It is also scales the texture along its length depending on the beam length so it's never gets stretched.

F3DPool

This script is a pool manager which is used to pre instantiate all the provided prefabs before the scene starts playing. All weapon scripts use **OnSpawned** and **OnDespawned** methods which also makes them compatible with other pool managers found on the asset store.

F3DProjectile

This script is a projectile controller. It is using ray casting to detect colliders in advance and in case of an impact calls corresponding method to play sound effects and spawn impact prefabs.

F3DPulsewave

This script is used to control pulse wave scaling and fading over time.

F3DRandomize

This script is used to randomize transform's scale and rotation for currently spawned object. Mainly used with muzzle flashes and projectiles.

F3DShotgun

This script is used to manage shotgun particle system and react to particle collision events sent by spawning impact prefabs and playing audio clips at impact points.

F3DTime

F3DTime class is a singleton instance used to create and manage the timers. To start using this component simply attach it to any gameobject in scene use one of the following overloads to create a timer:

int F3DTime.time.AddTimer(float rate, System.Action callBack);
int F3DTime.time.AddTimer(float rate, int ticks, System.Action callBack);

AddTimer method has two overloads. The first one can be used to invoke a specified method at specified rate until stopped while the second one requires you to specify number of ticks before it stops. The return value of AddTimer is a unique int handle which should be used with RemoveTimer method to stop it's execution.

Let's look at the code example below where two timers are created. Note that we store the id for the first timer in myTimerId variable so we could dispose it later.

```
int myTimerId;

0 references
void Start() {
    myTimerId = F3DTime.time.AddTimer(0.1f, OnTimer);
    F3DTime.time.AddTimer(5f, 1, CalledOnce);
}

1 reference
void OnTimer() { Debug.Log("Tick"); }

1 reference
void CalledOnce() {
    Debug.Log("Removing timer " + myTimerId);
    F3DTime.time.RemoveTimer(myTimerId);
}
```

Once initialized the first timer will begin invoking OnTimer method each 0.1 seconds until stopped by second timer after 5 seconds elapses. Then both timers are disposed since we explicitly tell first timer to be removed and the second self removes since it's life scope is only a single tick.

F3DTurret

This script is used to control turret's base and barrel rotation in a specified range as well as checking user input and invoking weapon firing methods on F3DFXController.

F3DWarpJump

Controls the warp jump effect by sending the appropriate messages to child objects invoking mesh tunnel scaling. Moves the warp spark through the tunnel and updates `_ShipPosition_` gameobject. Make sure to disable `SendOnSpawned` before using with pool manager.

F3DWarpJumpTunnel

The script is responsible for scaling the warp jump tunnel mesh, fading colors, and rotation. Should be used with F3DWarpJump script.

F3DWarpTunnel

Randomly rotates the warp tunnel over time.