Mesh Integration and Disintegration

Welcome and thank you for download this package. Hope you find it useful. Here in this document I will tell you every step you need to know to use it. Also, if you don’t want to read, there is an [easy video tutorial of how to use it](https://www.youtube.com/watch?v=P7CMt00kHik). You will discover how simply it is. So, this package and the code is fully documented. But, if you have any questions or doubts about it, you can always write me at [pablodusi@gmail.com](mailto:pablodusi@gmail.com) . I will be happy to answer all of your questions.

**Video demonstration**

**[](https://www.youtube.com/watch?v=Rii1Il_a2bc)**

**Description**

“Mesh Integration and Disintegration” integrates and disintegrates your models in tiny submeshes, and let you create effects like explosions, disintegrations, loading effects, and many more. And it does in an elegant and harmonically way. It can be used for a variety of utilities. For instance, you can use to explode a character every time a bazooka hits him. Or, maybe, you can use it for loading a scene, where the meshes load slowly by integrating its subparts. Or maybe you can use it for teletransportation purposes. Just three examples, but there are hundreds. You can use your imagination and do whatever you want. So let’s started to know how to use it.

**Video Tutorial**

If you don’t want to read the instructions, you can always [see the video tutorial](https://www.youtube.com/watch?v=P7CMt00kHik).

**Installation instructions**

There is nothing special about it. You just have to download the package as the usually way through the Unity Assets Store Window, and import it into your project.

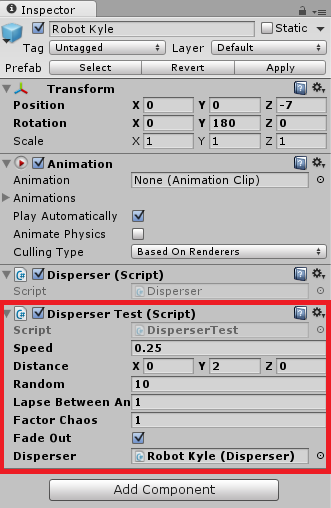
Please keep in mind that once you learned how to use this package, you can erase the “Robot Kyle” folder (3D Model for testing the package) if you want.

**Quick test (Easy Introduction)**

1. If you want, you can check it the very simple scene called “Test” (Under the Scenes folder).
2. You will see a robot named “Kyle” facing at the camera (It’s a 3D model from Unity Technologies).

****

1. Now you can test the scene. Please enter the play mode and test it.
2. You see? Now…there is a lot of things you can do with this behavior. Do you see the “Robot Kyle” GameObject at the hierarchy scene? Please select it.
3. You will see that it has a DisperserTest.cs class attached. This is the only class of the package that is not part of the solution. It’s just a test, so you can see the behavior.



1. But if you open the script, you will see that it uses the Disperser.cs class that is the class that serves as an interface for all the “Integration” and “Disintegration” behavior.
2. It’s very simple. When the line 69 executes, then the method disperser.StartActivate is called, and the integration occurs. And in the opposite way, when the line 82 executes, then the method disperser.StartDeactivate is called, and the disintegration occurs.

****

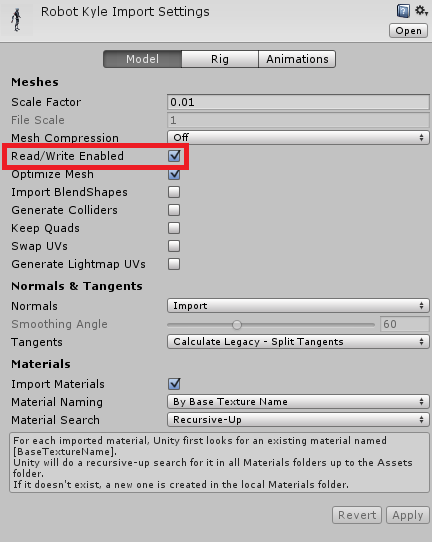
****

1. So finally, I encourage you to change some public variables of the DisperserTest.cs script, so you can see the different behaviors that you can make with this. For example, try to change the speed property to make it “slow motion”. (Something like 0.025f). Later, you will know more about it.
2. But for now, you just learned the basics of how to use this package. Congratulations! Once again, remember that the “DisperserTest.cs” class is just a test (I encourage you to make your own scripts) and if you want to call the “Integration” and “Disintegration” methods, you have to use the “Disperser.cs” class.

**How to use it (Step by step)**

(AUTOMATIC PROCEDURE – FOR MOST CASES)

1. I will assume that you have an empty scene at your project. So let’s start learning how to do it from the beginning.
2. First, find the 3D Model that you want to add at your scene. For instance, this can be an .fbx file. **Please be sure that the “Read / Write Enabled” property is enabled.** **This is very important.**

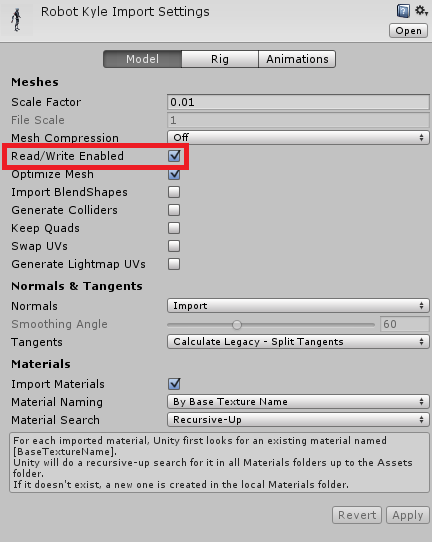
****

1. Once you have that, let’s add the 3D Model to the Hierarchy scene by dragging and dropping it. Rename the GameObject as you want and move it whatever position you want (Just make sure the camera can see it).
2. Remember to add a Directional Light if you want.
3. So, your GameObject has one parent and many childrens, right? In some of those childrens, you have the MeshRenderers (Or SkinnedMeshRenderers) attached.
4. The only thing you have to do in this procedure, is to find the parent GameObject at the top of the Hierarchy, and attach a DisperserBuilder class. This will automatically do all the work for you. It will attach all components in the parent, and the childrens.
5. If your model has two or more renderers, it will work too. For every renderer, the DisperserBuilder will attach some scripts.
6. Finally, you have to drag an drop the “Disperser Manager” prefab that’s in the prefab’s folder to the Hierarchy.
7. If you erase some scripts that are necessary or you forget to do something, the DisperserBuilder will fix everything for you. But in some cases there will be warnings telling you that something is wrong. So pay attention to them, please.
8. Now, you can test the scene to see what happens. If your model is high poly, or your machine is very slow, probably the performance will be ugly. But for those cases, please increase the TrianglesPerSubmesh value on the DisperserAnimatorManager script, and test it again.
9. If you want to remove all the scripts that the DisperserBuilder has attached you can add a DisperserDestructor script to the parent’s GameObject. Warning: This will destroy all the Disperser scripts for that GameObjects and his childrens.
10. Below you can learn how to configure the package and what are the relevant properties.

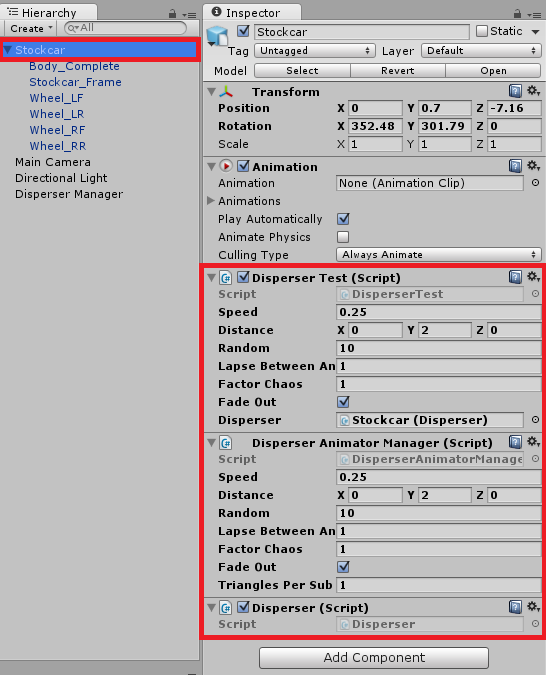
**How to use it (Step by Step)**

(MANUAL PROCEDURE – ONLY FOR SPECIAL CASES)

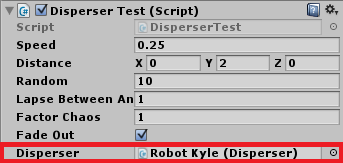
1. I will assume that you have an empty scene at your project. So let’s start learning how to do it from the beginning.
2. First, find the 3D Model that you want to add at your scene. For instance, this can be an .fbx file. **Please be sure that the “Read / Write Enabled” property is enabled.** **This is very important.**

****

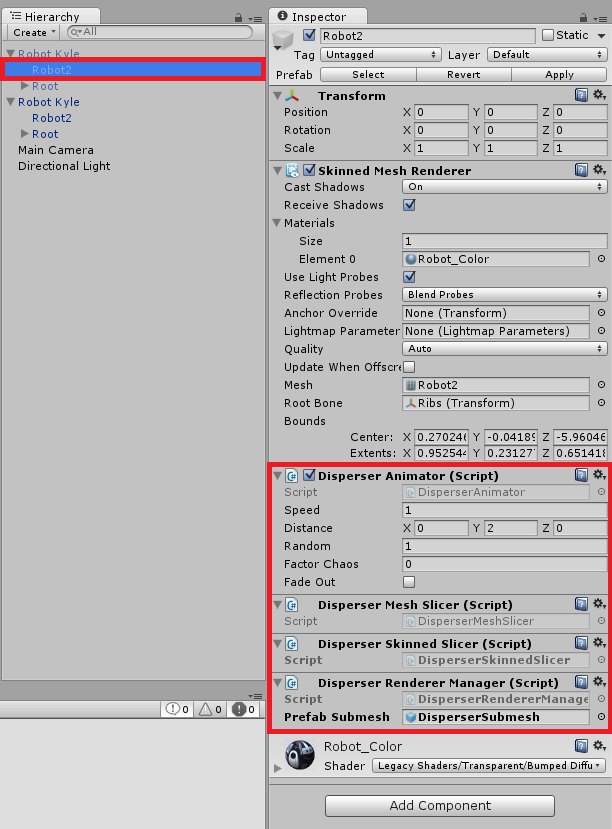
1. Once you have that, let’s add the 3D Model to the Hierarchy scene by dragging and dropping it. Rename the GameObject as you want and move it whatever position you want (Just make sure the camera can see it).
2. Remember to add a Directional Light if you want.
3. At the parent GameObject of your character (or whatever object you are representing) please add the “Disperser.cs” class, the “DisperserAnimatorManager” class, and the “DisperserTest.cs” class. Remember, “DisperserTest.cs” is just a test.

****

1. Make sure you have configured all the properties of the DisperserTest.cs class (See below for detailed information).
2. Remember to assign the Disperser Property of the DisperserTest class to a GameObject that has a Disperser (This Game Object that you selected has one!). This tell the script what Disperser you want to “integrate” / “disintegrate”.

****

1. Once you have that, let’s select the child that has the MeshRenderer or SkinnedMeshRenderer component. Select it, please.
2. Then, please attach a DisperserAnimator.cs class and a DispenserRendererManager class. If your Renderer is a MeshRenderer, then attach the DisperserMeshSlicer class. And if your renderer is a SkinnedMeshRenderer, then attach a DisperserSkinnedSlicer class.

****

1. Please check that the DisperserRendererManager class has the property PrefabSubmesh with a correct prefab (I added one for you, but it can be whatever prefab with a MeshFilter, MeshRenderer and a DisperserAnimatorChild component).
2. Please remember to make this thing (Step 8 to 10) for every renderer you have. Sometimes, one 3D Model is represented in two or more Renderers. (Find the childrens to check it, please.
3. Finally, drag and drop the “Disperser Manager” prefab into the Hierarchy.
4. Now enter the play mode and test it. That’s it! You learned it! Congratulations! If you want to learn more about it just keep reading.

|  |
| --- |
| You will see that the materials that I use are the legacy ones. This is for version compatibility, but you can use whatever material you want.  But…if you are using the Legacy Shaders materials, and you want to the submeshes alpha to be fade out as a disappearing effect, then you have to use **transparent materials**. Otherwise, if you are using the Standard Shader materials, then you have to use the Transparent rendering mode materials. Remember, this is just to make sure your submeshes makes the alpha fade out. |

**UV Mapping**

If you don’t want the submeshes to have the same material as the real mesh has, you can always disable “UV Mapping” from the DisperserProperties class localted at the DisperserManager GameObject.

**Events**

If you want to catch when an “integration” or “disintegration” has been completed, please use the “OnDisperserDeactivatedMesh” and “OnDisperserActivatedMesh” methods of the Disperser class.

**Warnings**

Please pay attention to the warnings. If something goes wrong, i will notify you through the warnings. I will stop all the execution of this package scripts if an error occurs. The warnings messages will tell you what’s wrong and how to solve it.

**Methods**

The two mainly methods to do the “integration” / “disintegration” effect are StartActivate and StartDeactive from the Disperser.cs script.

**Configuration**

I will list the configuration properties that appears in the DisperserAnimator, DisperserAnimatorManager, and the DisperserTest script. This properties allows you to configure all the “Integration” and “Disintegration” effect.

* **Speed:** Velocity of the activation / deactivation. You can play with this property. For example, if you put lower values, you will discover a “slow motion effect”.
* **Distance:** This determines a constant distance from the real position of the mesh, that the submeshes begin to move (Or ends in case of Disintegration)
* **Random:** A random value for the start position of the submeshes in a Sphere. It is used with Random.insideUnitSphere.
* **FactorChaos:** If the value is lower, the submeshes will start in accordance with the "random" property above. If not, the submeshes will start in chaotically positions (Not inside a Sphere).
* **FadeOut:** If it’s enabled, then when the disintegration occurs, the tiny submeshes alpha will fade out slowly.
* **TrianglesPerSubmesh:** When the game starts, the 3D Model slices in thousands of tiny submeshes. If the model is a high poly one, there can be too much, and the performance will be affected. So, if you increase this property, you will see that the performance gets better. Because, while this property increases, the game creates less submeshes.
* **ThreeHundredSixtyDegreesEffect:** If it’s enabled, then the submeshes will rotate from 0 to 360 degrees repeatedly.

**Performance Notes**

|  |
| --- |
| If you note that for some reason the performance is low because of this package, you can always adjust the TrianglesPerSubMesh property on the DisperserRendererManager class. The optimal use is “1” (You will see thousands of tiny submeshes) but if you are experimenting troubles with the performance, you can increase this value and test it again.  You can also try by disabling the uvMapping from the DisperserManager GameObject. |

**Problems**

|  |
| --- |
| If you are experimenting a problem with the Disperser classes, then, I recommend you to destroy all the Disperser classes assigned to that Game Object and its children, by using the DisperserDestructor class on the parent Game Object and creating all of them again by using the DisperserBuilder class.  This is a nice way to get a clear and nice behavior from scratch. |

**Classes**

Now I will tell you all the purposes for every relevant class.

* **Disperser:** If you don’t want to touch any code, this class can be very useful. It serves as an interface between the system and everything else. It has two important methods to make the “Integration” and “Disintegration” very easily.
* **DisperserAnimator:** It makes the “Disintegration” and “Integration” animations for one specific renderer.
* **DisperserAnimatorManager:** It makes the “Disintegration” and “Integration” animations for every render in the childrens.
* **DisperserTest:** It’s just a test to probe the effect.
* **DisperserMeshSlicer / DisperserSkinnedSlicer:** It slices the mesh in thousands and tiny submeshes and creates one GameObject (With the necessary components) for every submesh created.
* **DisperserBuilder:** Just add this class to the parent’s GameObject and it will make all the job for you. It will attach all the classes needed.
* **DisperserDestructor:** Add this class to the parent’s GameObject and it will destroy all the components that DisperserBuilder had created previously.

There are more classes, but not enough relevant to show you what they are.

**Trying another 3D Model**

Please remember, “Mesh Integration / Disintegration” is designed to work in every single 3D Model, because it uses their renderers, that can be only two: MeshRenderer and SkinnedMeshRenderer. If you want to try another 3D Model before implement this package in your project, I encourage you to do it. I suggest to download a free 3D Model package from the Unity Assets Store. And if you want to test the “Multiple children with one renderer each” feature (In other words, this package works with 3D Models that have one or more Renderers), I suggest you to try with the [Stockcar Racecar](https://www.assetstore.unity3d.com/en/#!/content/1843) from Unity Assets Store. If you want to try with a character, I suggest you to implement the [Ghoul](https://www.assetstore.unity3d.com/en/#!/content/39110) package. This are simply 3D Models that you can import to your project and test the “Integration / Disintegration” effects again, with another objects. They are only a few suggestions. Try whatever you want!

**Comments**

Please note that all the code of the project is fully documented. If you have any questions or doubts about it, you can always write me at pablodusi@gmail.com . I will be happy to answer all of your questions.