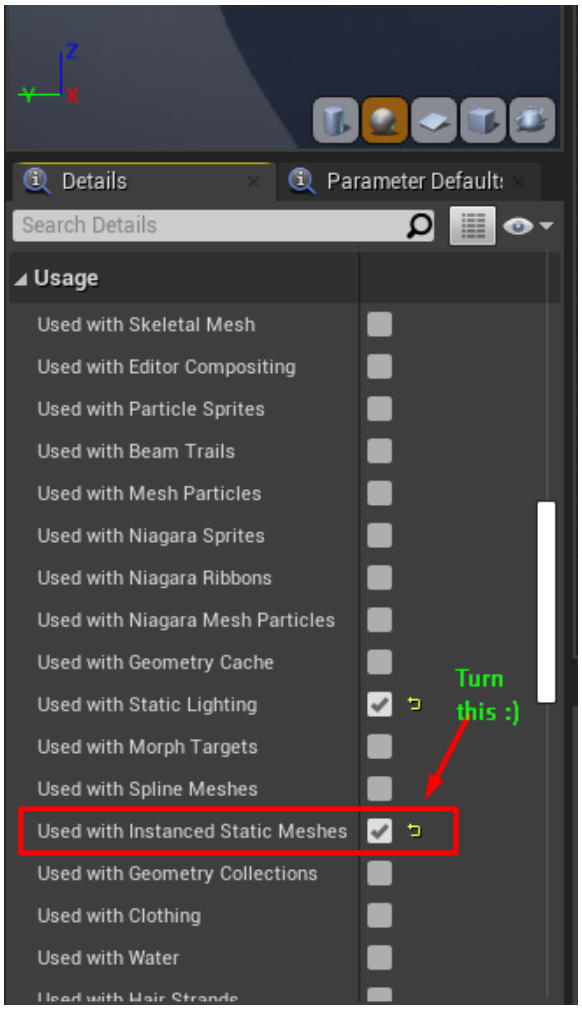
Contents:

!Warn!
In order for your materials to work correctly, you need to enable the "Used with Instanced Static Meshes" parameter in the material settings.



[september 6, 2018]
v.1.0:
> Release.

[september 25, 2018]
v.2.0:
> Saves System added.

[October 6, 2018]
v.3.0:
> Parameters and references saves added.
> Fix bugs in bp library and terrain widget. Deleted some print nodes. Improvements of blueprints comments.

[October 11, 2018]
v.4.0:
> User-friendly interface. The main menu is removed, and all buttons are conveniently distributed on the screen. Editing levels is now easier, faster, and more obvious. The general interface space remains the same and does not interfere with the review.
> Ability to delete saves and clear current level.
> Changeable Panorama backgrounds and floor materials (water as default) (savable).
> Fog adjustment (savable).
> Autosaves with adjustable timer. Saving before the start of the mode of ingame and loading after it is turned off (to reset the position of activity on the map after the game).

If you want to offer something or report about a bug, I will be glad if you contact me at the following contacts:
Discord group

1. Adding to the created project.

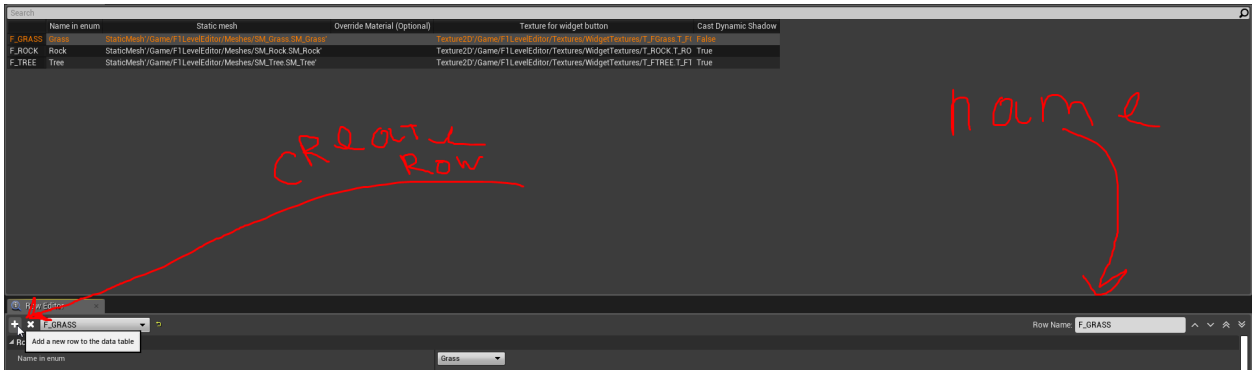
There are a few things you need to do to add an editor to your project.
1) You need to take control of the character "CH_EditorCamera", located in the "Blueprints" folder (You can do it with “Poss” function in blueprint or edit project settings).
2) Call the level that comes bundled with the asset. You can edit this level, but be careful with deleting objects.

Also, when you press the "Play" button already in the game itself, the character "CH_Player" is taken under control. How to change it on your own is described below in paragraph #5.

This is enough to bind the editor to your game.

2. Add or change (except activity).

To do this, you need to do just two basic things.
1. - Go to the "Data" folder and open the enum file that you need. For example, "Enum_Foliage". Create a new item there and give it a name. So your object in the data table will be called.
2. - Open the desired data table. Their names begin with "DT_". As an example, this could be "DT_Foliage". Create a new line or modify an existing one. You can give the line a name, but this is not necessary.



2.1 - Substitute the required data in the line. In different types, these data may different, but they are all quite understandable and simple. For example, foliage has a "Cast Dynamic Shadow" parameter that other types do not.

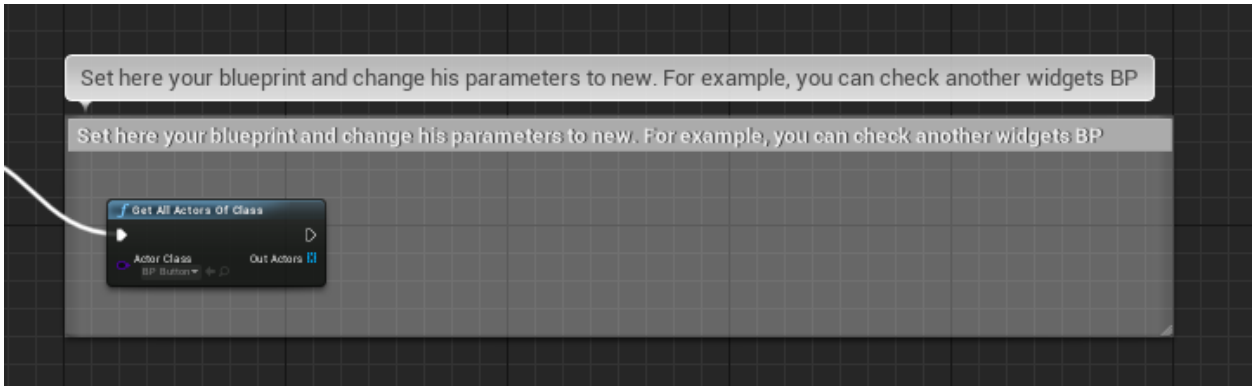
That's all, your object is added to the game. Congratulations!
It's easier than it sounds, isn't it?

3. Add or change activity.

If you do not need dynamically tuned parameters, then adding activity is no different from adding other types of objects.

If you need parameters, then create a new object in the same way as described on the last page. After that follow the further guide.

3. - The data table requires a widget. Copy the widget "UI_ClearForCopy" and paste it into the table.
4. - Add the necessary things to the widget, such as various sliders, text fields and all that is useful to you.
5. - In the area marked by the comment, apply all changes. This event will be called after closing the widget. If you encounter any problems with this, you can see how it is implemented in two other widgets located in the same folder.



That's all you can check in the game. If you need the functionality of “adding clicks” to actors, see how this is implemented in neighboring widgets. You can generally just copy them.

4. Optimization Tips.

Here I will introduce some optimization tips. They may seem obvious, but my business is to remind :)

1. Adjust the range of dynamic lighting. This can be done in the source settings. The in-game editor obviously does not support static lighting, so you should be careful with dynamic lighting. Try to choose the optimal range / speed. You can try to remove skylight from the level if you do not need it.

2. Look at your materials. As an example, in the asset there is material "M_ModLandscape". Try to apply it to the object and create a few dozen of these :) With a large number of objects - material optimization is of great importance.

3. The number of polygons is also of great importance. When working with foliage, you can increase the size or decrease the density (you can set the minimum and maximum available to the player in the foliage widget). When working with other objects, you just need to be careful.

4. Foliage uses HISM, which means full support for LODs. Use it! I also noticed an interesting thing that the drawing range does not work correctly, but you can kill the distant LOD at 0% of the vertices.

5. You can also increase the grid of objects, although this is an ambiguous solution. How to do this is described in the paragraph below.

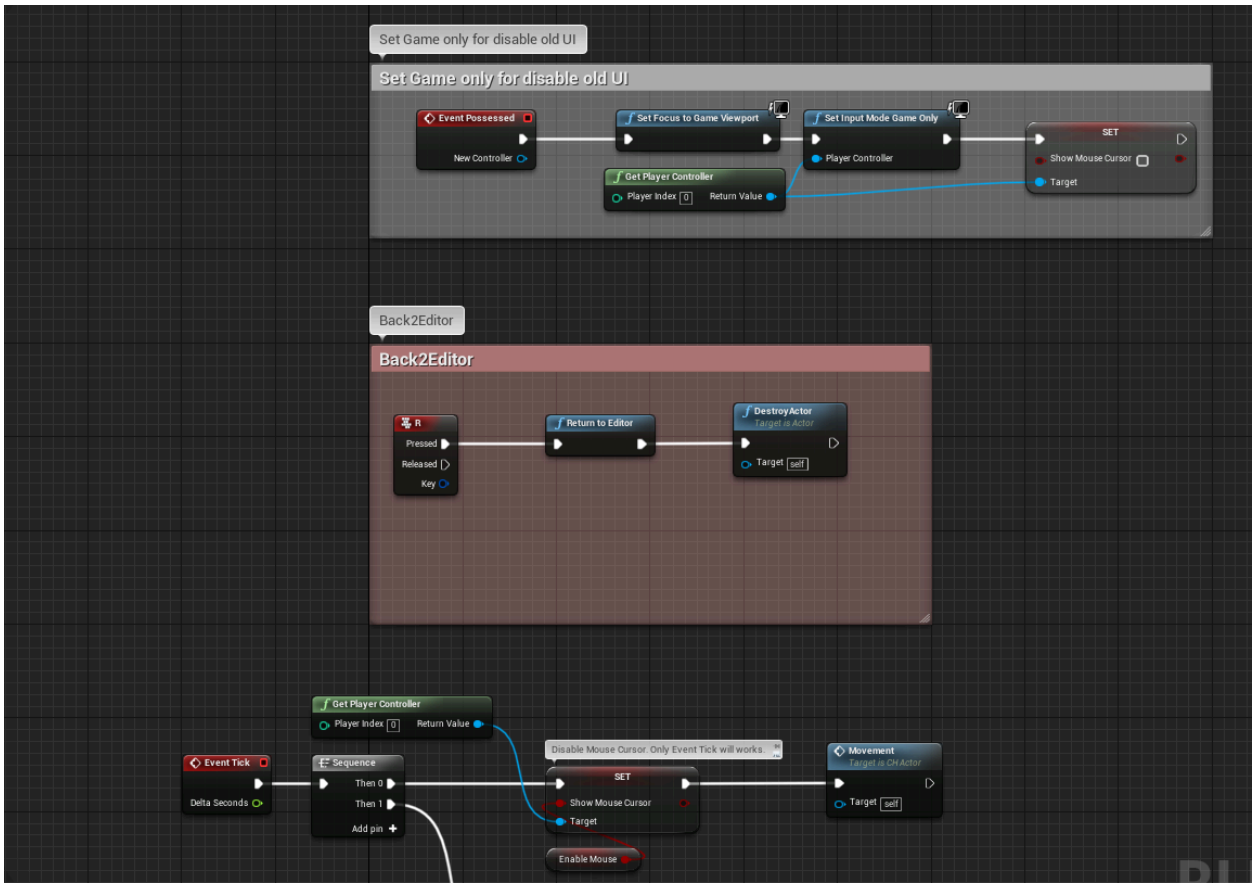
5. Resize grid for objects and another way to change playable pawn.

1. Open blueprint called "BP_Config".
2. In the parameters, change the value of "Grid Scale".
Do not forget that you will have to adjust the sizes of the objects so that their sizes correspond to the grid.

Also in this blueprint is the parameter "Player". Insert your character here. Editor with use this parameter for spawn ingame character.

Here you can also set the frequency of autosaves by changing the corresponding parameter. By default, it is 90 seconds. Saves do not load the computer much, but obviously you should not call them too often. I would recommend using values of at least 30 seconds, although with much lower values, the FPS will not fall too much.

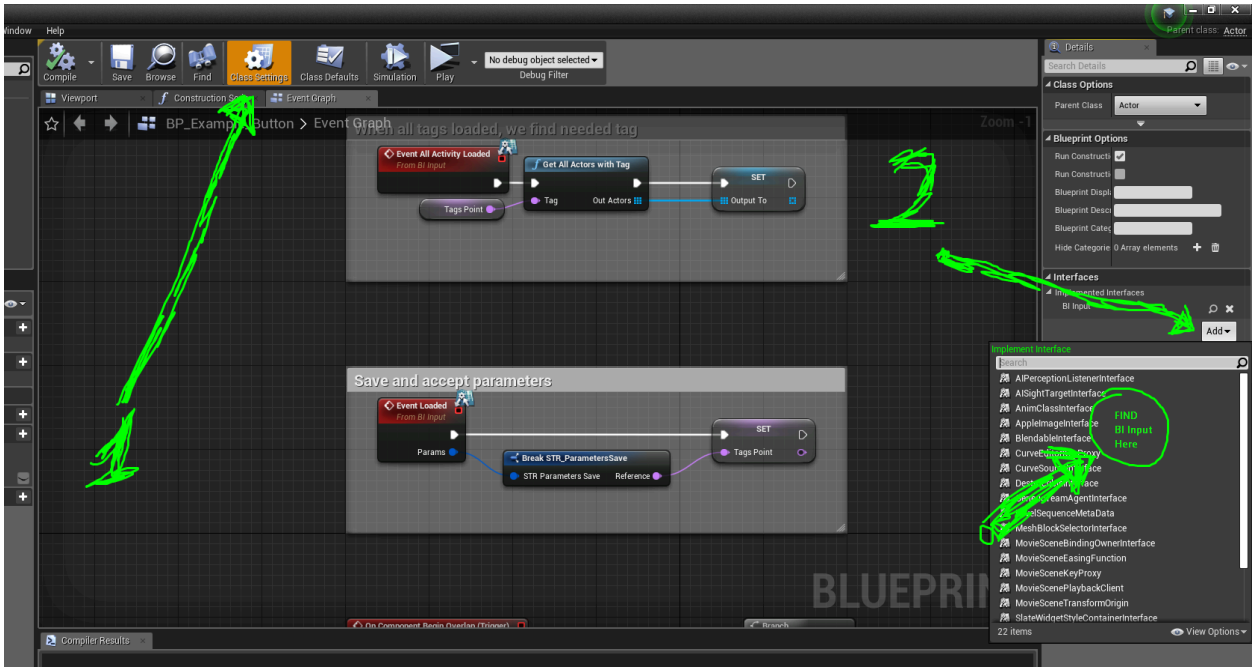
You can go back to the editor with a simple construction. This output will reset all changes made to the level during the game.



6. Saving activity parameters

To save the parameters, a reasonably flexible system was invented, which you can easily expand.

First you need to add the blueprint interface to the desired object.



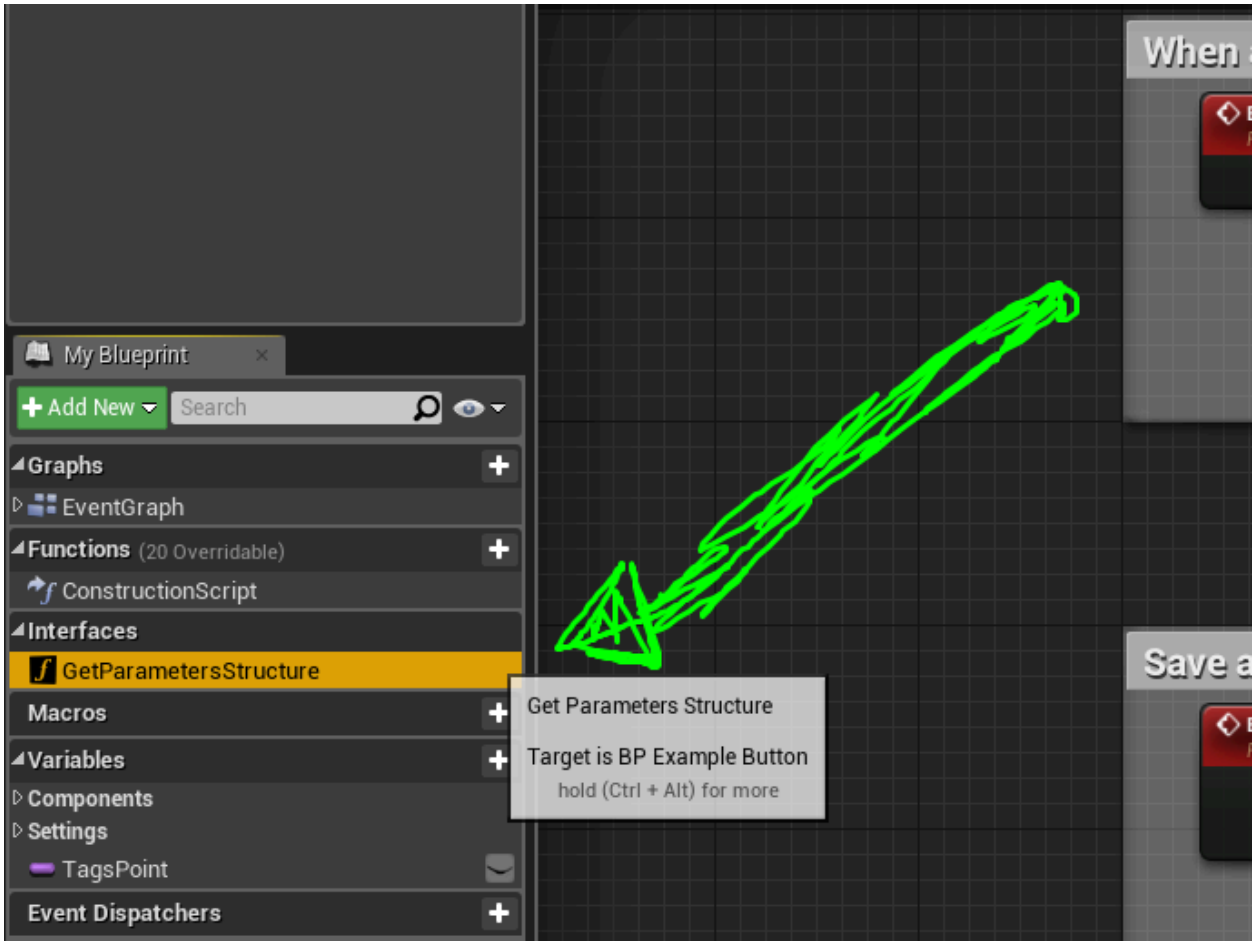
Add an interface called "BI Input"

After that, three new events will be available to you.

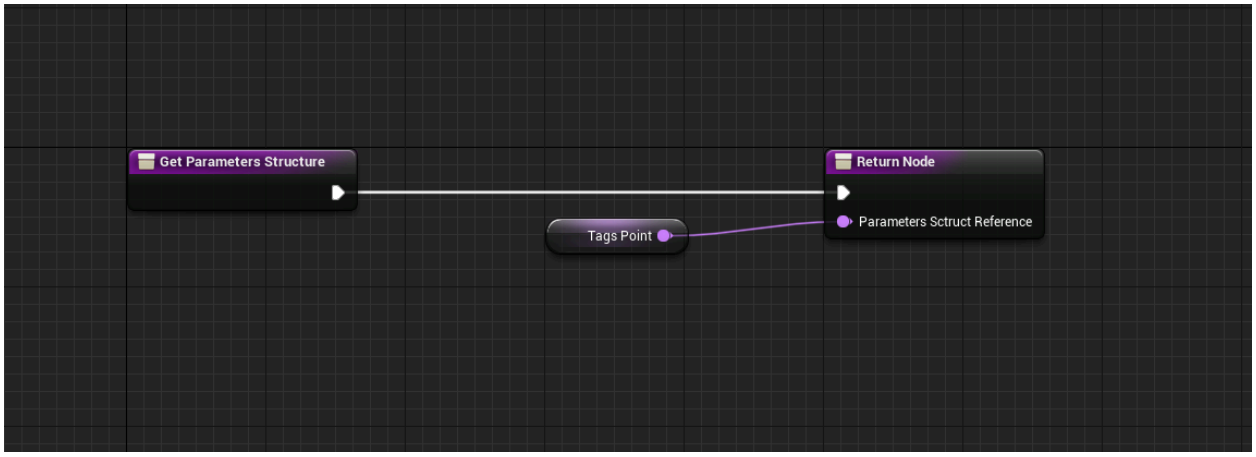
Events for LOAD:
1. **Event Loaded** - Called when your object has loaded. But this does not mean that the rest have loaded. He also gives you saved parameters (which can be configured. How to do this is described below). Here you can simply overwrite the variables in the object or even save the structure in a variable. You can do something else if necessary.
2. **Event All Activity Loaded** - It is called only when **ALL** activity is loaded. *This is useful, for example, when one object is dependent on another, as is done in the examples. In order for the button to open the door, it is necessary that both the door and the button are already created and tags are applied to them.* If you do not need to work with other objects, you can not use this event.

Events for SAVE:

To do this, you need to create an override event "**GetParametersStructure**". This event will be called by the **save algorithm <u>when you will actually save</u> the level.**



You will receive the next event. Now you just need to insert your variables. In the example, only one is used - this is the target tag.



Try it and you yourself will understand that this is done very simply. I just described in detail what is being done and what it is for.

To add your parameters, open the file "**STR_ParametersSave**", which is located in the "**Blueprints/Data**" folder. After that, just add the necessary parameters there.

I described the basic principles, but you can always see how this is implemented in the examples, since they exist.

7. Add your backgrounds and floors (Added in v.4.0).

Just open the file "DT_Environment" and add the necessary materials to the array.

As an example, standard panorama textures, 1:4 ratios are used (1024x4096). Backgrounds use the default sky sphere as a model. Therefore, you should have no problems creating your own better and more optimized materials.