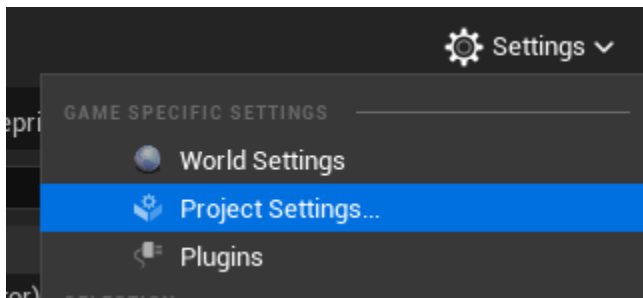


Procedural City Generator

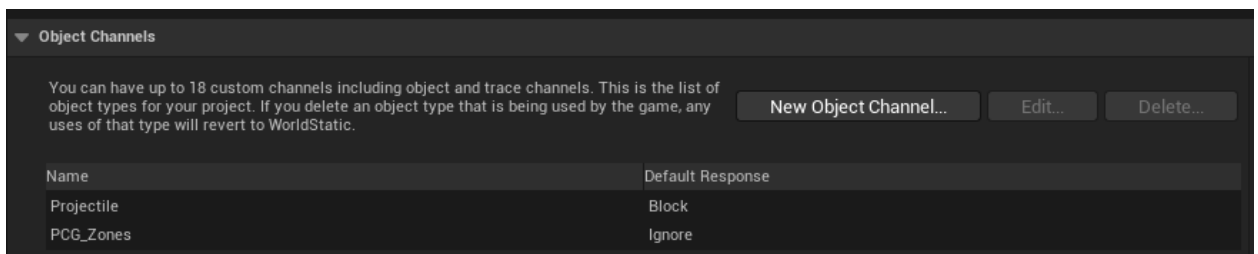
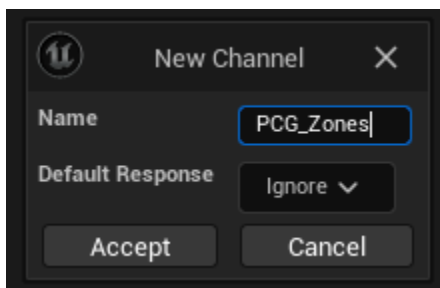
Migration Guide for Unreal Engine 5

Please follow these steps to migrate the **Procedural City Generator** to your own project.

1. Open a fresh copy of the Procedural City Generator
2. Open your custom project
3. Inside your **custom project**, click on **Settings > Project Settings** to open the **Project Settings**.



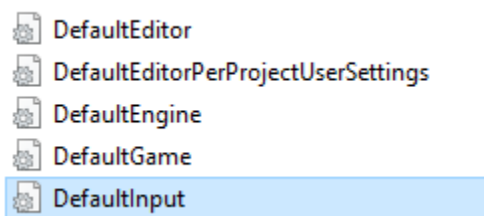
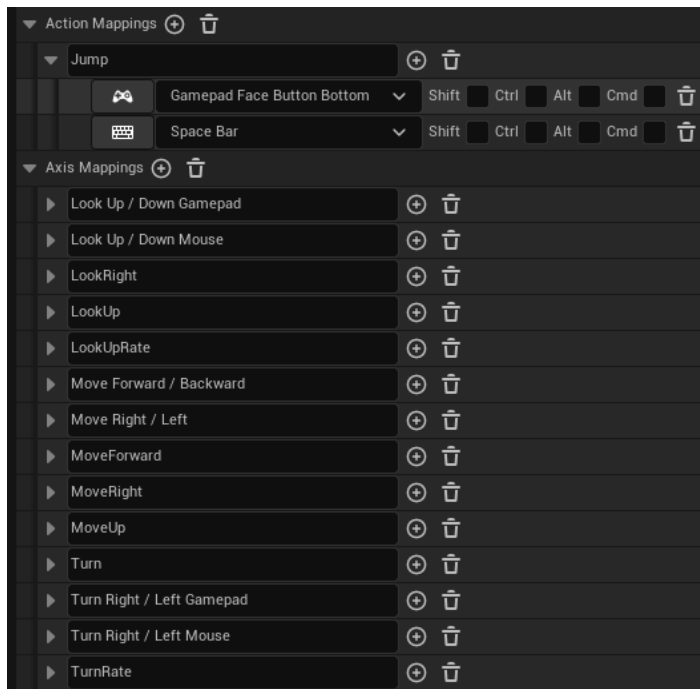
4. Go to **Collision** and click on **New Object Channel** and call it "**PCG_Zones**" with a default response of Ignore. Make sure the spelling is correct.



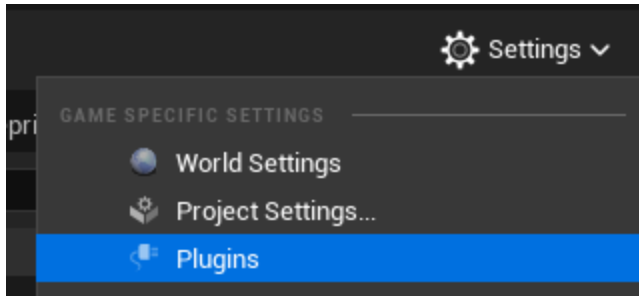
5. Go to **Input** and notice the **Action Mappings** and **Axis Mappings** available. You can recreate these mappings in your custom project by hand, or you can copy the

DefaultInput.ini file (Project Folder > Config > DefaultInput.ini) from the ProceduralCity project to your custom project.

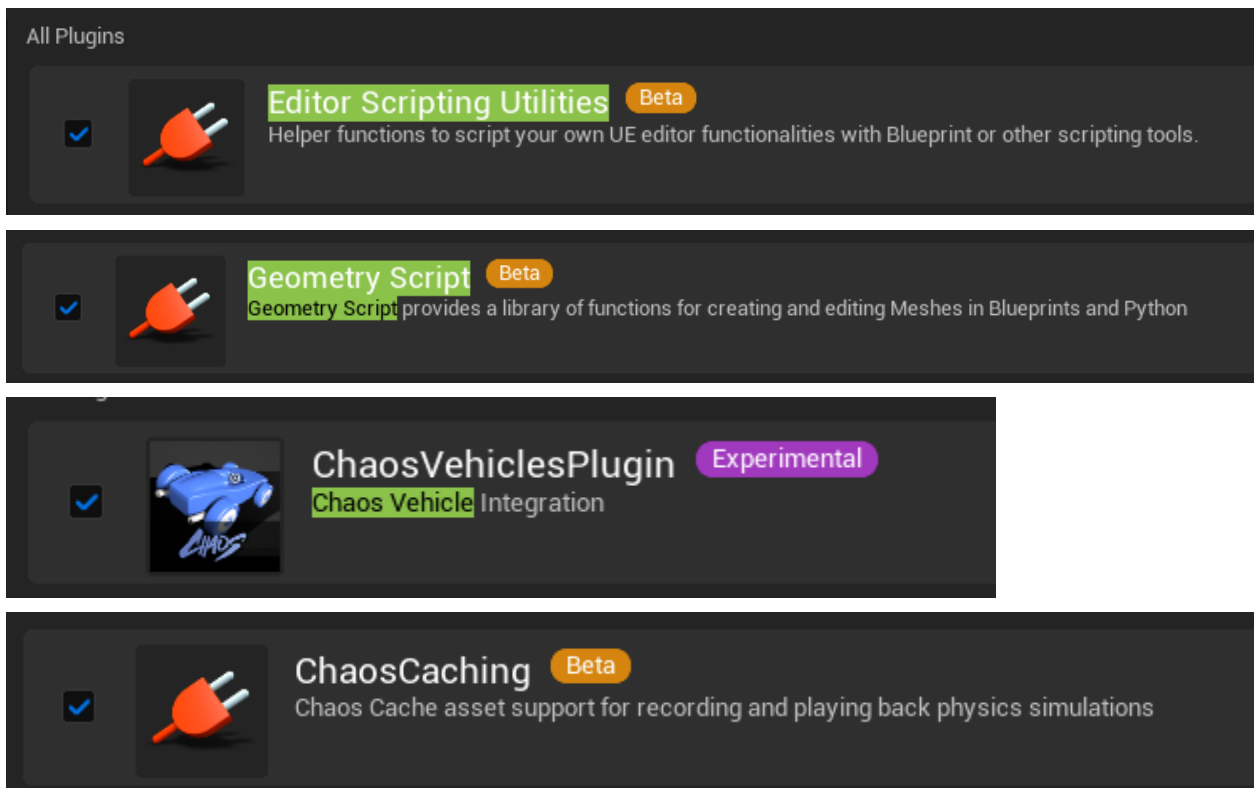
Warning: If you have other inputs already set up in your custom project, do NOT override the file as you will lose your settings. Instead, open the files and simply copy the mappings manually.



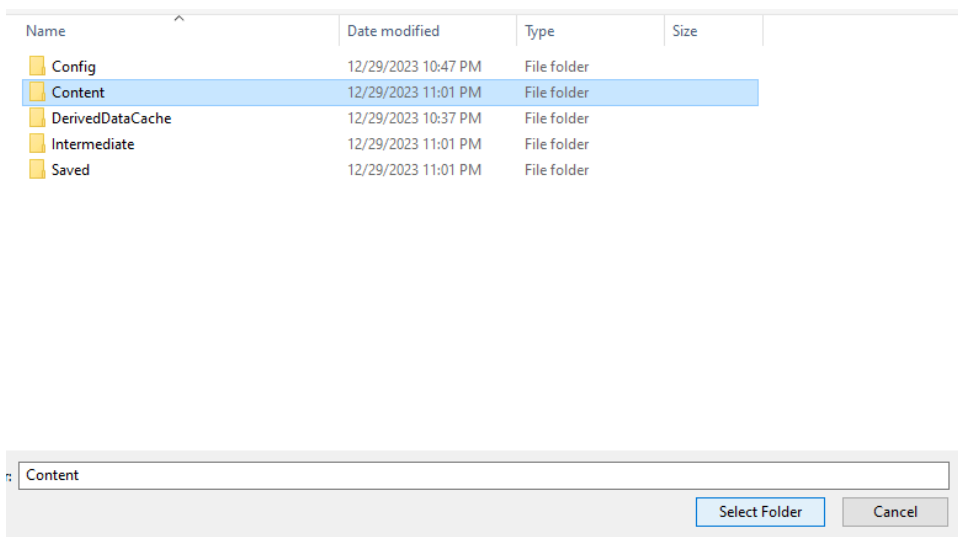
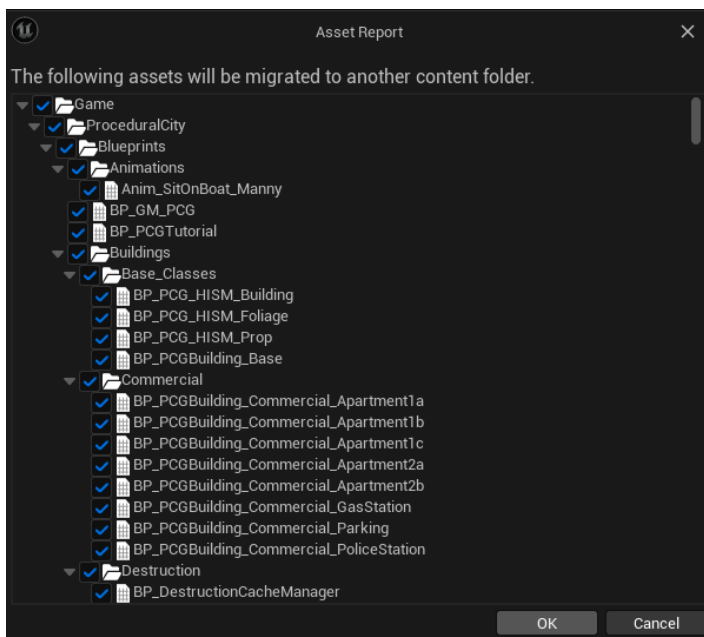
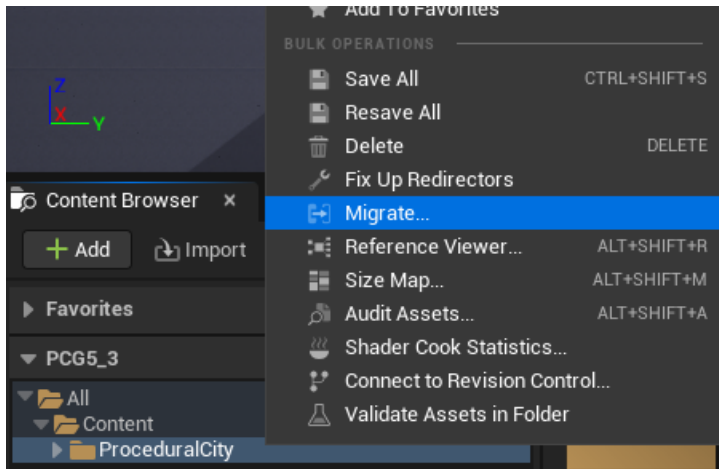
6. Click on **Settings > Plugins** to open the **Plugins** menu.



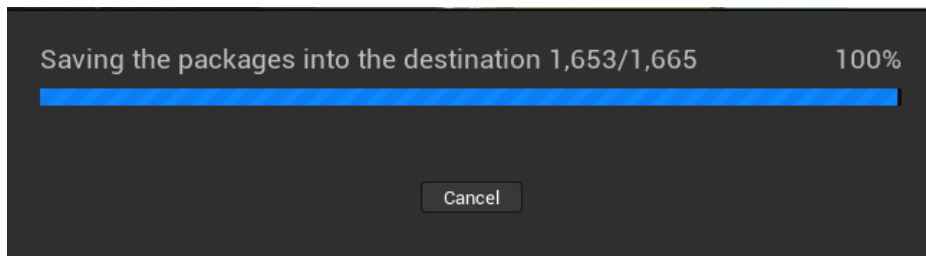
7. Search for and **enable** the following plugins:
 - a. Editor Scripting Utilities
 - b. Geometry Script
 - c. Chaos Vehicle Plugin
 - d. Chaos Caching



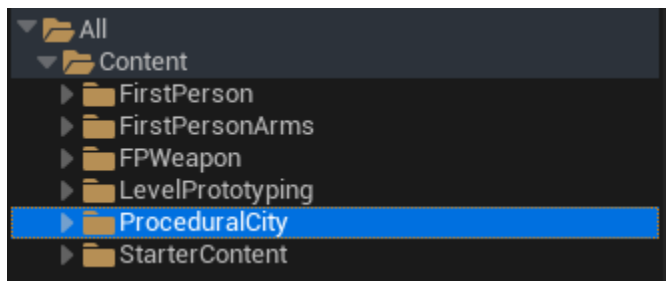
8. **Restart** the editor if needed.
9. Right-click the **ProceduralCity** folder (top level) and select **Migrate**. Click **OK** in the **dialogue box**, then navigate to your custom project and select the **Content** folder as the folder destination. Click **Select Folder** to start migration.



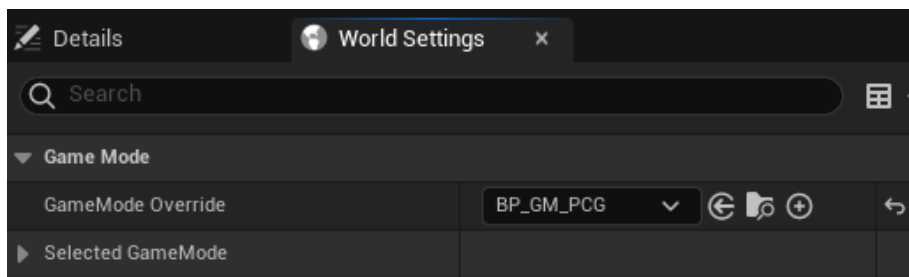
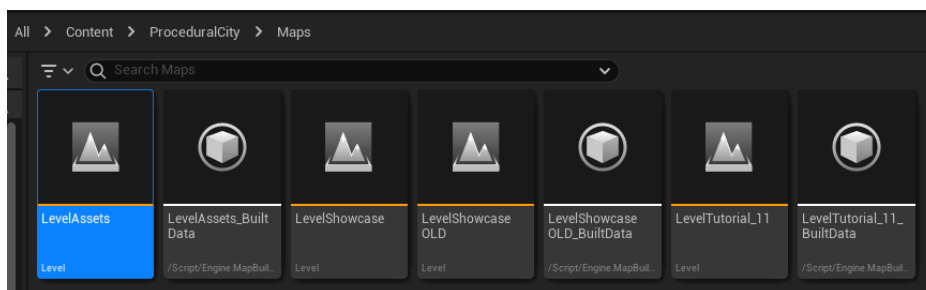
10. Wait for the files to be copied over to your custom project.



11. Once the files are copied, go to your custom project and verify that the **ProceduralCity** folder is now available in the **Content Browser**.



12. Go to **ProceduralCity > Maps** and open **LevelAssets**. Then under **GameMode Override** select **BP_GM_PCG**.



13. Press **Play** and verify that the default PCG character appears and you can walk around the level.

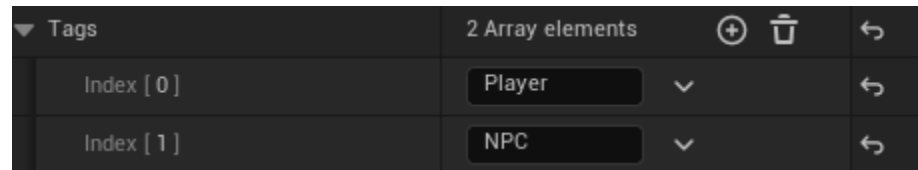


14. Lastly, if you are using a custom character Blueprint you will need to do the following:

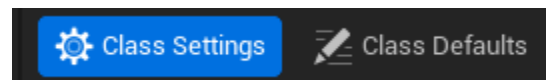
a. Open **BP_ThirdPersonCharacter** under **ProceduralCity > Blueprints > Characters**.

b. Open your **Custom Character** and do the following:

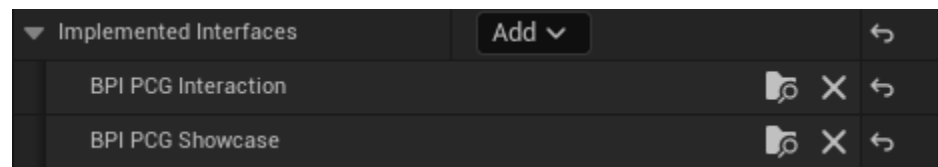
i. Add the actor tags **Player** and **NPC**.



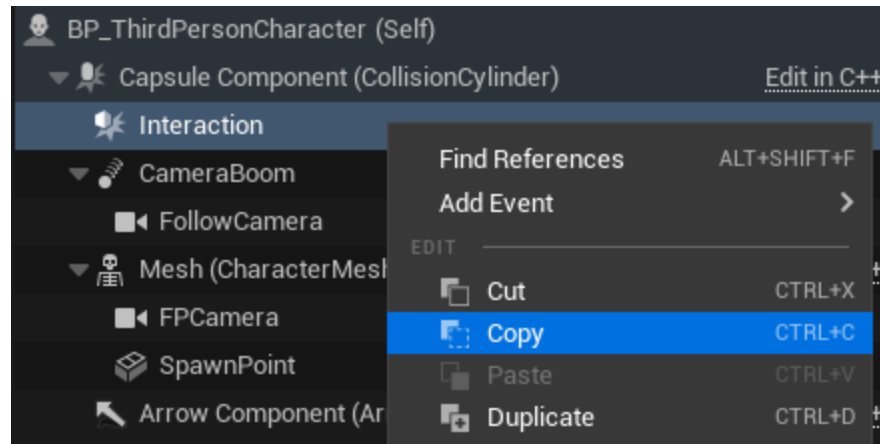
ii. Click on **Class Settings** at the top



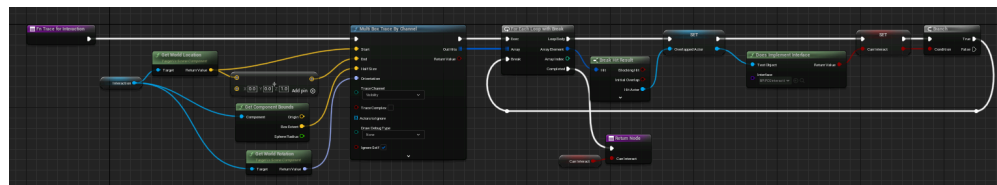
iii. Go to **Implemented Interfaces** on the **Details** panel and add **BPI_PCG_Interaction** and **BPI_PCG_Showcase** by clicking **Add**.



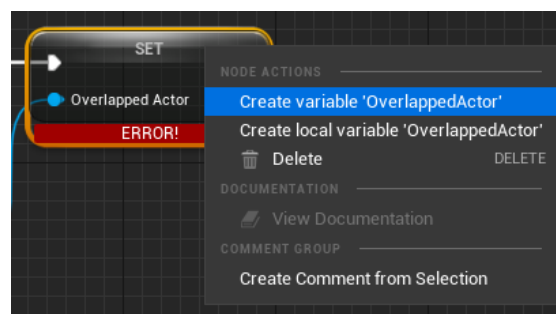
c. Go to the **BP_ThirdPersonCharacter** and **copy** the **Interaction box** under the **Components** window.



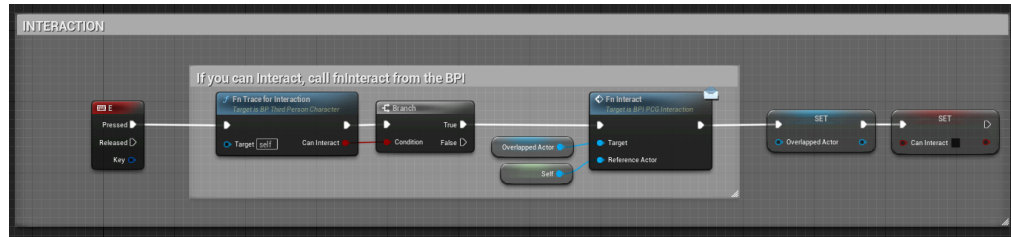
- d. Go back to your **Custom Character** and **paste** it. Make sure you adjust its location and rotation appropriately.
- e. In your Custom Character create a new function called **fnTraceForInteraction**. Then, go to **BP_ThirdPersonCharacter** and copy the nodes inside that same function. Then paste them in the function created for your Custom Character.



- f. Compile your Custom Character and notice the errors. Right click the **OverlappedActor** node and select **Create variable 'OverlappedActor.'** Do the same for the node **CanInteract**. Compile the BP again. You should not see any errors.



- g. Go to the **Event Graph** of **BP_ThirdPersonCharacter** and **copy** the nodes under **Interaction**.



- h. Paste these nodes inside the **Event Graph** of your **Custom Character**. **Compile** your **Custom Character** again and make sure you see no errors.
 - i. [Optional] You can copy other nodes from **BP_ThirdPersonCharacter** if you want additional functionality, such as spawning vehicles at runtime or changing the time of day.
15. Migration is now complete.