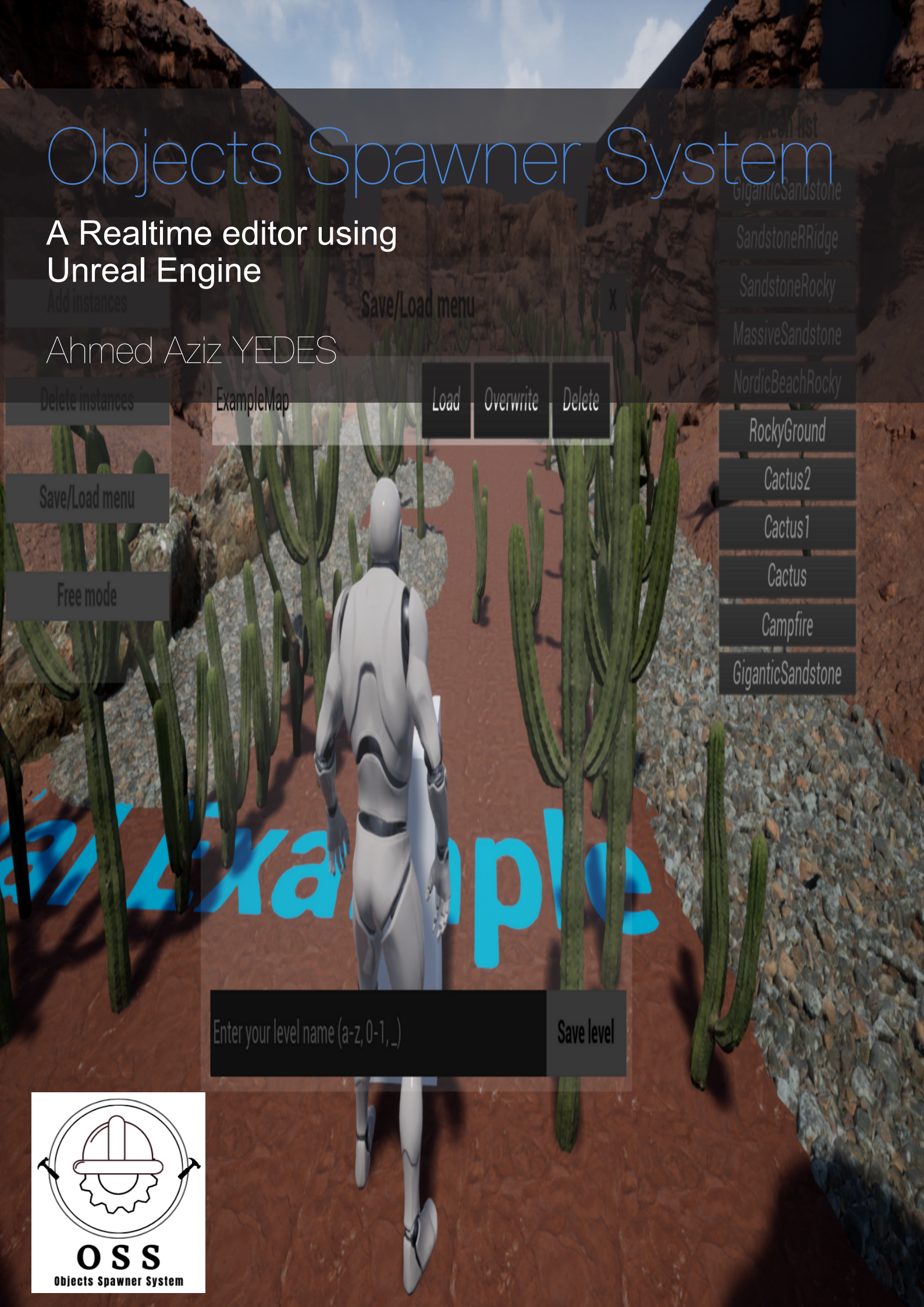
Objects Spawner System

A Realtime editor using Unreal Engine

Ahmed Aziz YEDES



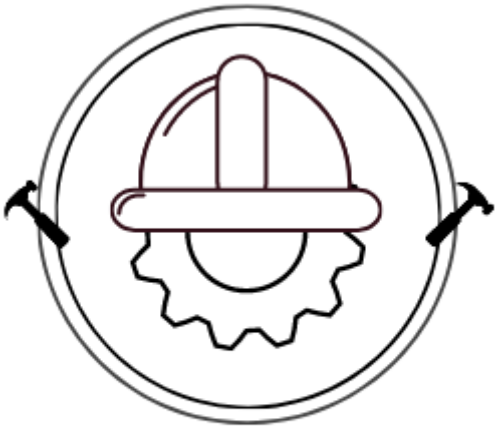
OSS
Objects Spawner System

Objects Spawner System

A Realtime editor using Unreal Engine

by

Ahmed Aziz YEDES



OSS

Objects Spawner System

Summary

Objects Spawner System, aka OSS, is an Unreal Engine content that allows the end-user to place meshes at runtime using the mouse cursor, delete unwanted objects that had been spawned, save scenes and load previously saved ones.

OSS can spawn unlimited types of meshes, all you need to do is create your list of meshes and it will produce its proper components from them.

OSS uses **Hierarchical Instanced Meshes** to reduce the number of draw calls made by the meshes.

With OSS, you can save your scene and you can also load your previous ones.

Contents

Summary i

List of Figures iii

1 Introduction 1

2 Getting Started 3

2.1 Main actors . . . 3
2.2 Setting up the actors . . . 4
2.3 Importing and using meshes with OSS . . . 4
2.3.1 Adjusting collisions . . . 4
2.3.2 Adjusting materials . . . 5
2.4 Using OSS . . . 6
2.4.1 Add instances mode . . . 6
2.4.2 Delete instances mode . . . 7
2.4.3 Save/Load mode . . . 7
2.4.4 Free mode . . . 8
2.4.5 Help button . . . 9

3 Notes 10

List of Figures

1.1 Empty level 1
1.2 Level Example1 2

2.1 Content browser 3
2.2 BP_Spawner variables 4
2.3 Setting up collisions 5
2.4 Mesh without material 5
2.5 Adjust material 6
2.6 Main menu 6
2.7 Add instances 7
2.8 Add instances 7
2.9 Save/Load menu 8
2.10 Help message 9

1

Introduction

OSS allows users to place objects in the scene at runtime. It also allows them to start with a base level and design it as they wish. They are able to save the current level, stop the game, run the game over again and load previous levels.

Example of a base level that can be designed to be a more complex one:

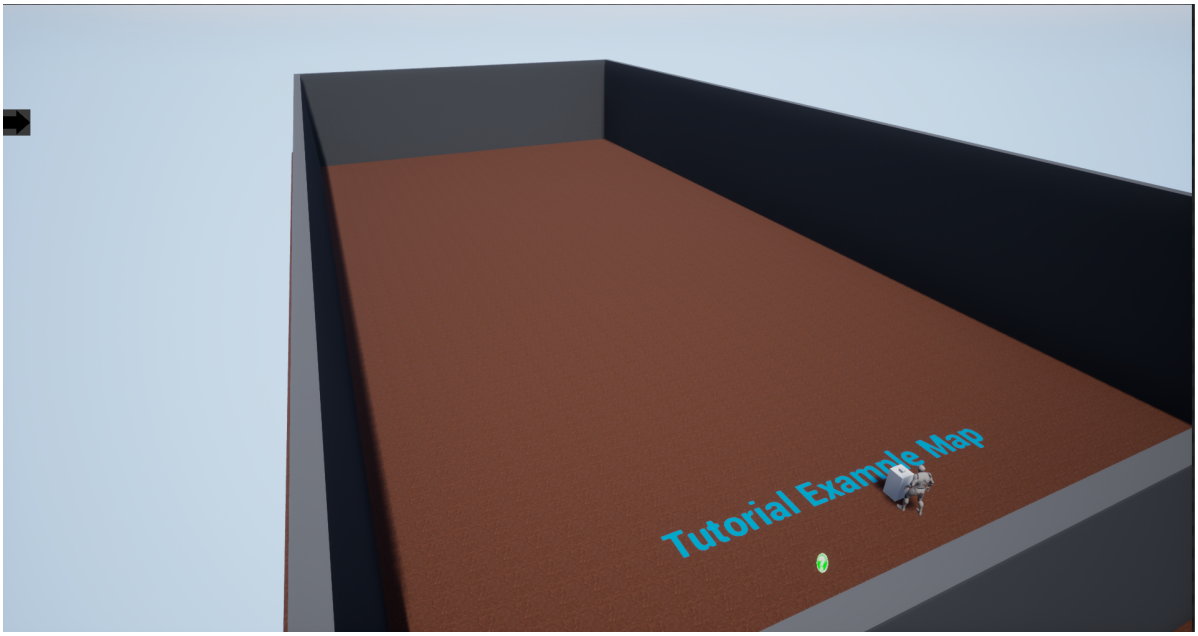


Figure 1.1: Empty level

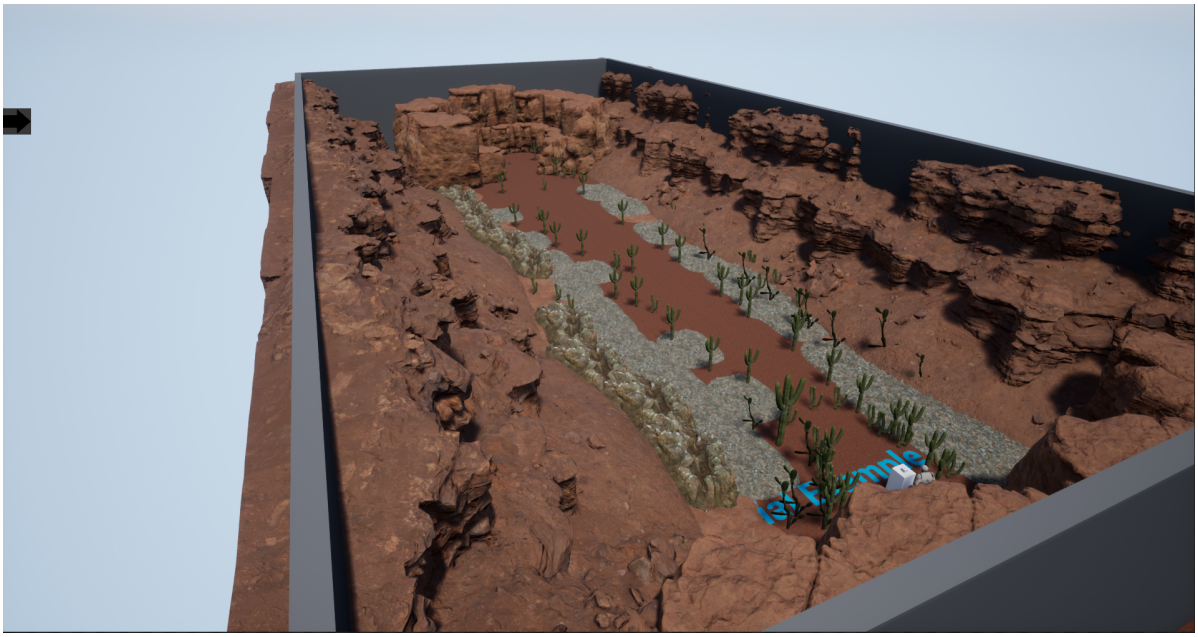


Figure 1.2: Level Example1

2

Getting Started

Watch **this video** for a quick showcase of the content utility.
The **video** will show you how to:

- Place the **BP_Spawner** in the scene.
- Create your list of meshes.
- Add instances of objects to the scene.
- Save your level to a save slot
- Load your saved level.
- Remove instances from the level.

The content includes two example maps:

- **TestMap** : A basic map with a plane and the spawner. This map allows you to place the objects the second you start the game.
- **ExampleMap** : This map is the Third Person tutorial map, with the third person as the main character. It shows an example of how to use the content with a third person character and a trigger button that puts you in the level editor mode. In this mode you can edit your scene by placing your objects, and then switching to your player to play in the edited level.

2.1. Main actors

OSS comes with mainly two actors: the **BP_Spawner** and the **BP_Pawn**.

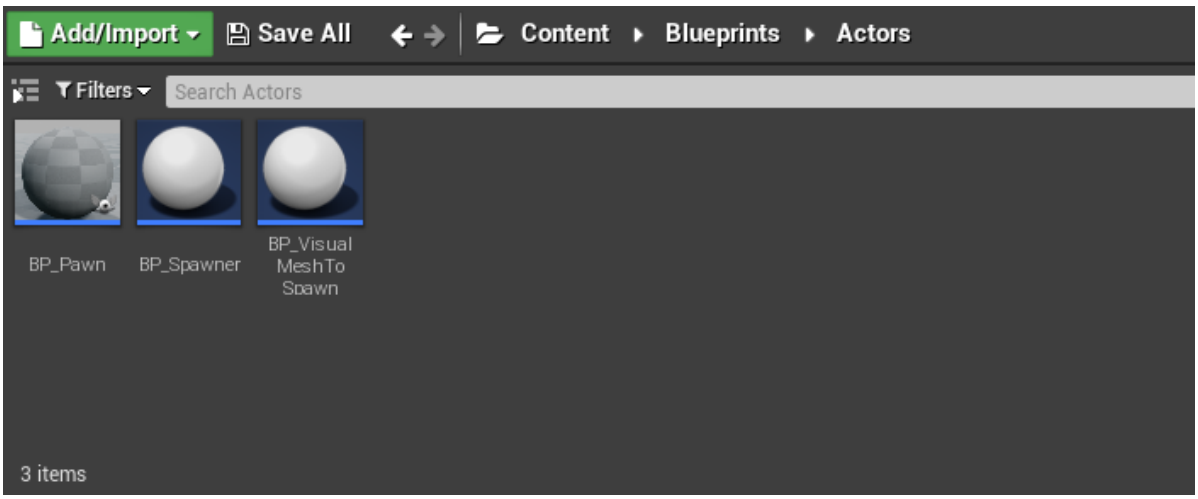


Figure 2.1: Content browser

The **BP_Spawner** is the actor in charge of accomplishing the major work. THis actor, takes the list of meshes and creates the different components used to spawn instances with all the projections from the 2D to the 3D environment.

The **BP_Pawn** is the default pawn used by the game mode in the level **TestMap**. Its role is to receive the input events from either the mouse or the keyboard and signals the **BP_Spawner** to do the work.

2.2. Setting up the actors

You would need to place the **BP_Spawner** in the scene. Then, you would have to fill the parameter **Static Meshes** which contains the list of meshes you would want to use to edit your scene.
The second variable you may want to adjust, is the **Show UI**. It's a Boolean that decides whether or not to show the **user interface** at the beginning.

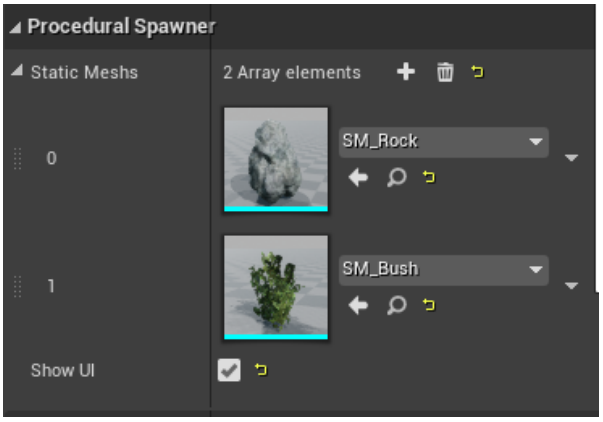


Figure 2.2: BP_Spawner variables

You can also fill the list of meshes at run-time and update the Spawner and the user interface. For example, in the level **ExampleMap**, the third person character is the one who spawns the **BP_Spawner and the BP_PawnExample** and then fills their lists of meshes using his own list of meshes. You can follow the example given in the map **ExampleMap** that shows another way to use the spawner.

2.3. Importing and using meshes with OSS

OSS can be used by any type of mesh, you only need to import your own meshes to the editor, adjust some settings and then start using them with **OSS**.
OSS uses **Hierarchical Instanced Static Mesh Components, HISMC** to reduce the number of calls and achieve better performance. Therefore, some major adjustments need to be done on the meshes to get them to function with HISMC.
To adjust the meshes, there are two major points that we would need to cover:

- **Collisions**
- **Materials**

2.3.1. Adjusting collisions

To be able to generate collisions with your instances, you would need to:

- Set the mesh's **Collision Complexity** to **Use Complex Collision As Simple**.
- Set the mesh's **Collision Presets** to **Block All**.

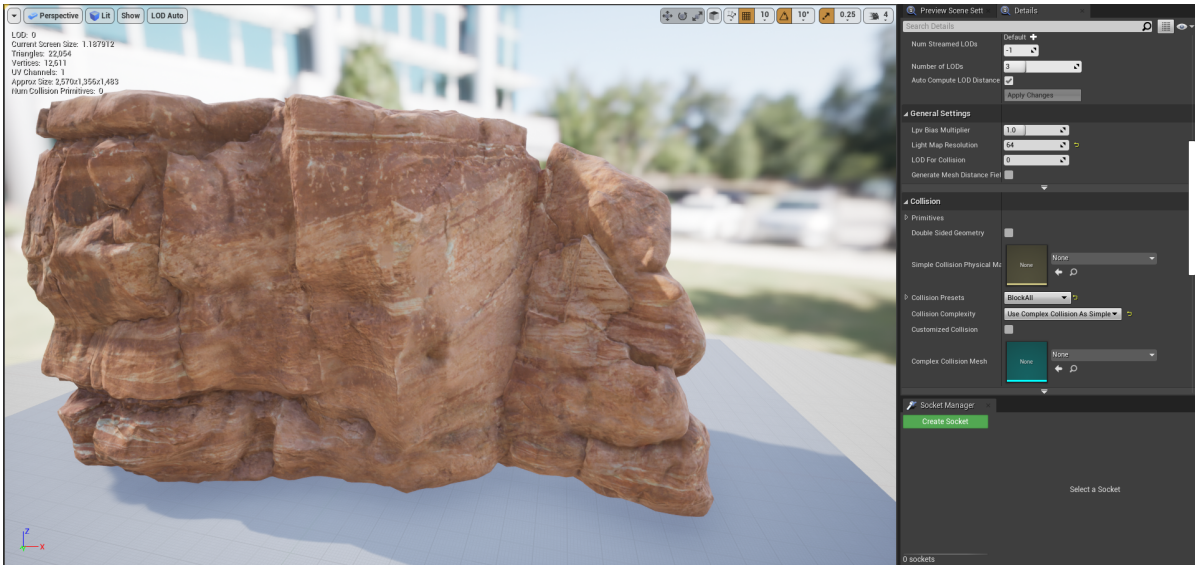


Figure 2.3: Setting up collisions

2.3.2. Adjusting materials

Sometimes, when you add instances to your scene using **OSS**, you may notice that your meshes are without any materials as you can see in this example:

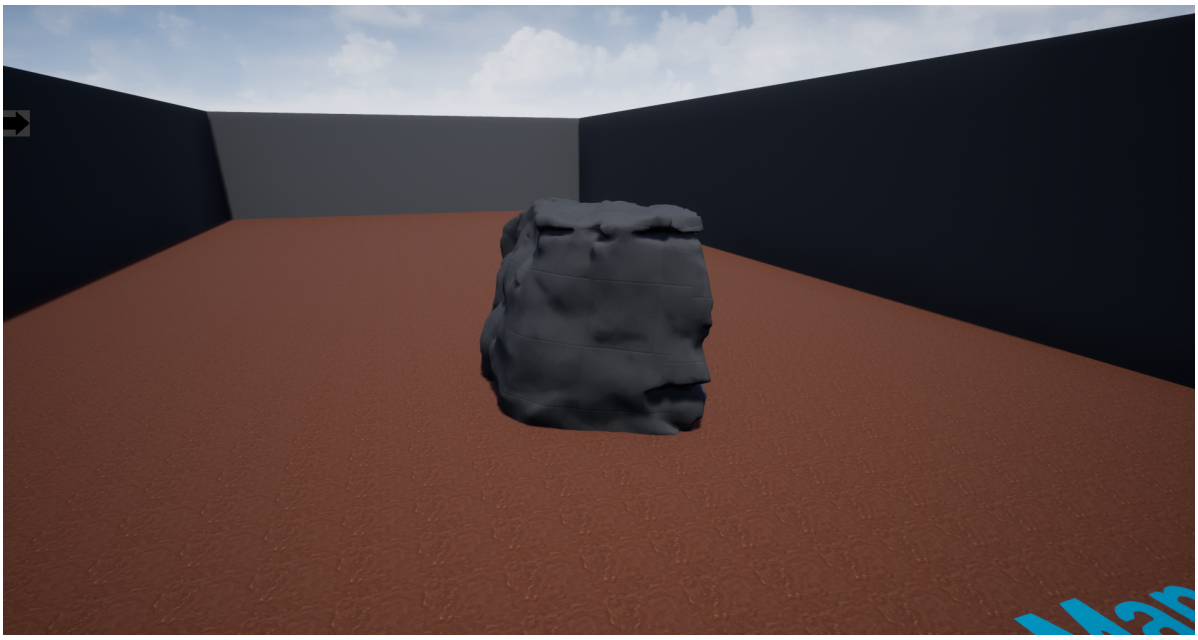


Figure 2.4: Mesh without material

This means that your mesh material is not adjusted to be working with **Instanced Static Meshes**. In this case, no panic, all you need to do is:

- Open your material
- Go under **Details -> Usage -> Used with Instanced Static Meshes**
- Or simply search for **Used with Instanced Static Meshes** with the search bar.
- Check that box
- Apply and Save your material

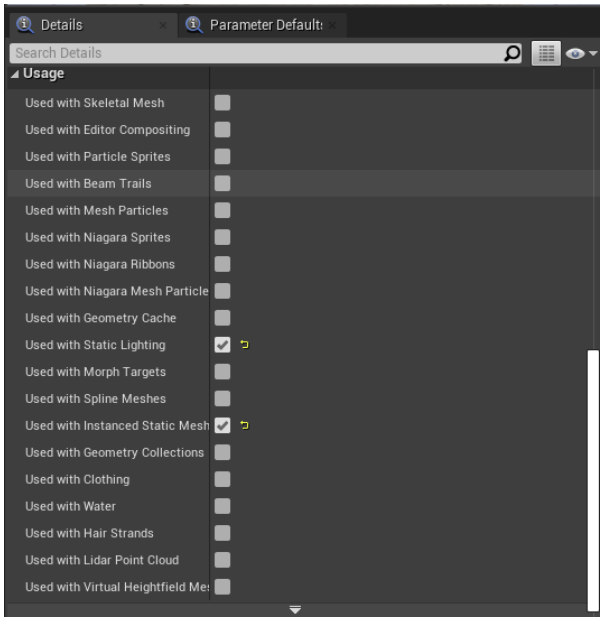


Figure 2.5: Adjust material

2.4. Using OSS

Once everything is set up, you can now use **OSS**.
When you hit play, if your variable **Show UI** is set to **True**, an arrow will appear at the view-port left edge.
Once you click on the arrow, the main menu containing the different usage modes will appear.

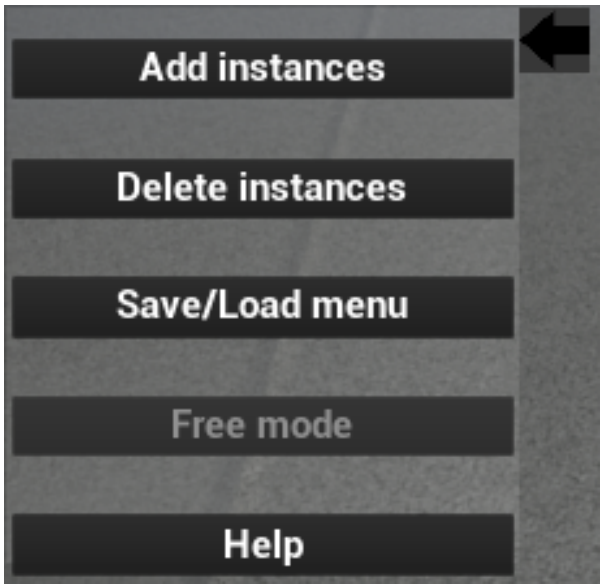


Figure 2.6: Main menu

2.4.1. Add instances mode

In this mode you will be able to add objects to your scene. When you are in this mode, a list containing all your meshes will appear on the right. You can select your mesh from this list.
Use your mouse to choose where you want to add the selected object. A yellow object with the shape of your selected mesh will appear to show you exactly where and how your object will appear on the scene.
In this mode you can:

- Select a mesh from the mesh list
- Rotate your mesh clockwise and counter-clockwise by an angle of 15°
- Scale up and down your mesh by a scale factor of 0.1. (Scaling is by a percentage of the selected mesh)

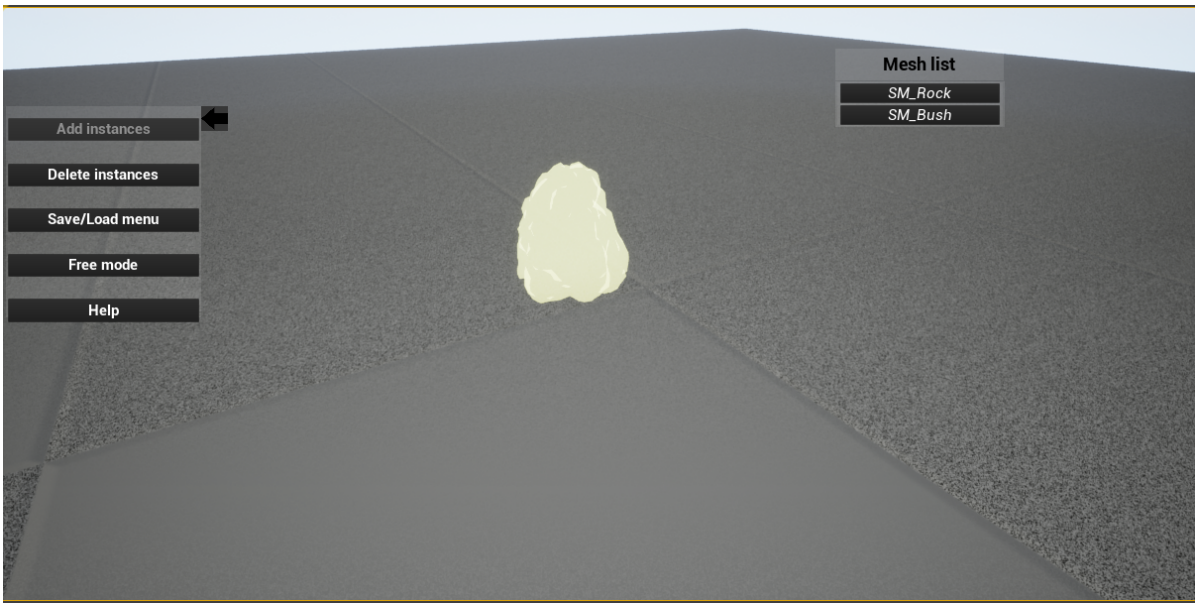


Figure 2.7: Add instances

2.4.2. Delete instances mode

In this mode you will be able to delete one added instance at a time. To delete an object, you need to click on it using your mouse. You will notice a small sphere of a **1 cm** radius. This sphere is displayed to be sure that we detect the wanted instance.

Figure 2.8: Add instances

2.4.3. Save/Load mode

This mode allows you to save your edited levels and load previous saved ones.

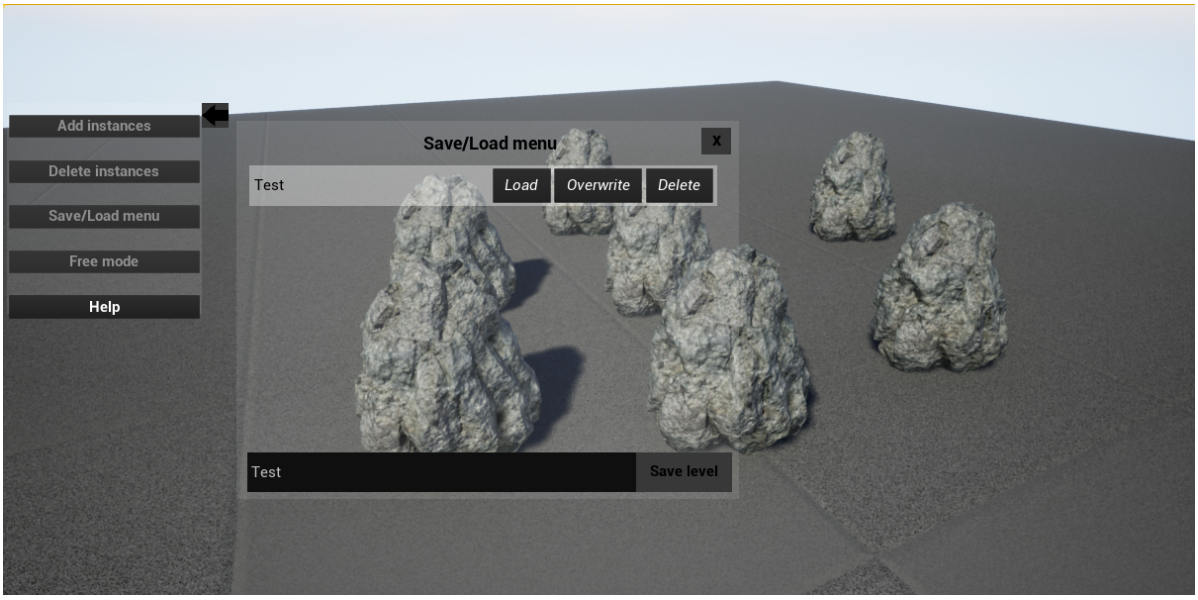


Figure 2.9: Save/Load menu

Save levels
You can save as many scenes as you want. To save a scene, you would need to open the **Save/Load menu**, and then click to save the scene.
Of course a save slot must have a name, and the name must:

- Have a non **empty** name, if not, an error message will pop up.
- Be unique.
- Have less then 23 characters (This is an aesthetic choice), if not an error message will pop up.

When you save your level, a **.sav** file will be saved in your **Saved folder in the project directory**. If you stop the game and then re-run it, you will find your previously saved levels.
If you want to overwrite an existing slot, all you need to do is click on the **overwrite** button instead of creating a new slot.

Load levels
You can load your saved levels by clicking on the **Load** button of the wanted slot.
When loading a scene, your actual level will be lost forever. All instances will be removed and the ones from the save slot will be placed instead. Also, if the saved slot has meshes that are not included in your current mesh list, these ones will be added to your list so you can add and remove instances of that type of mesh.
Remember, even if you stop the game then re-run it, you can still load your levels (as long as the .sav files exist).

Remove levels
You are also able to delete unwanted slots by clicking on the **Delete** button of the slot that is meant to be deleted. However, you have to keep in mind that: **Once deleted, it's gone forever. Even the .sav file will no longer exist**.

2.4.4. Free mode
The free mode, is the mode where no actions are taken. You can move your pawn, check the scene or implement your own mode. In this mode, you are able to roam freely in your map without having to worry about mistakenly deleting your instances or having the visual mesh to spawn as such actions cannot be taken while you are in the free mode.

2.4.5. Help button

This button will show you Default inputs to use with the pawn to send actions to the spawner:

- **Mouse Left:** Add instances and Delete instances (it depends on which mode you are using).
- **Mouse Middle:** Rotate the object by 15° clockwise.
- **Shift + Mouse Middle:** Rotate the object by 15° counter-clockwise.
- **Mouse Scroll Up:** Scale Up the object by 10% its own scale.
- **Mouse Scroll Down:** Scale Down the object by 10% its own scale.

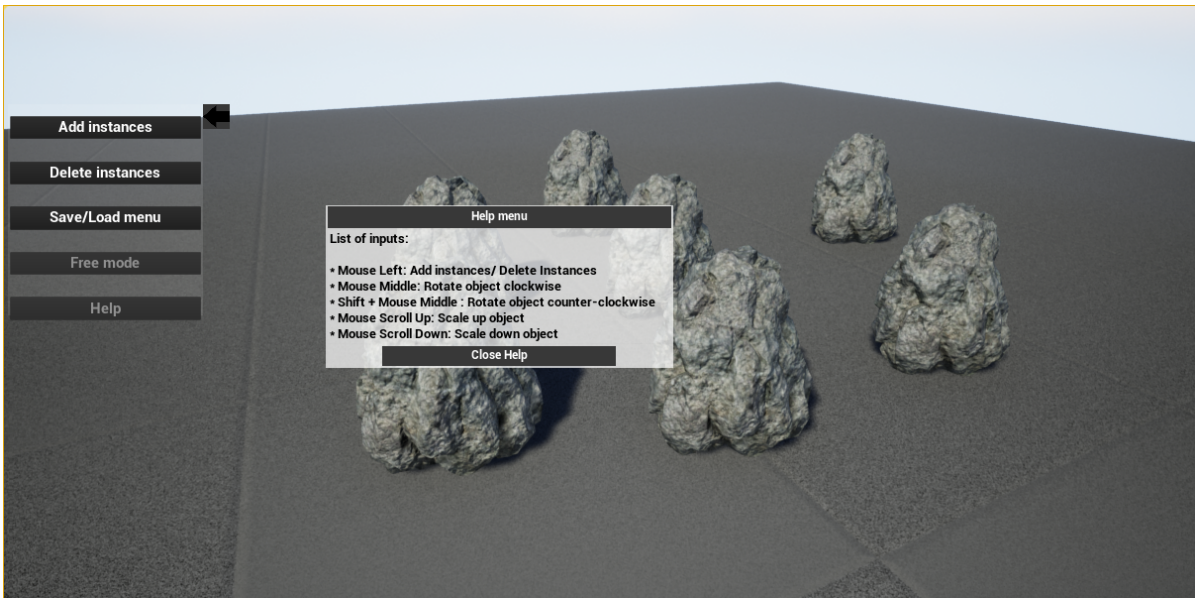


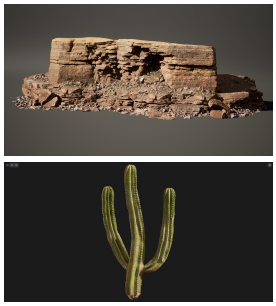
Figure 2.10: Help message

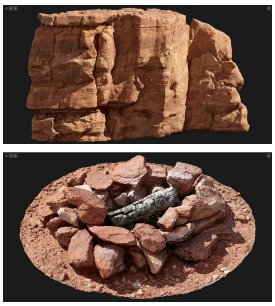
3

Notes

There are some notes worth mentioning about **OSS**.

• **OSS** comes with some basic content from the Unreal Engine's starter content. All the content used for showcase in the documentation, the **Test Demo** and the **showcase videos** are not included. All those assets are available for free on **The megascan contents**.





• **OSS** comes with two maps for two different use cases. The **TestMap** is used in case we would like the spawner to be the immediate level editor. Whereas the **ExampleMap** is in case we integrate the spawner to a Third Person game and interact with the level.