Python Automation for Asset Import

1. Script Overview & Usage

This script automates the import and configuration of static Mesh and texture assets in Unreal Engine using the Python API. It was designed for batch-process assets from a folder and implement required configuration for LODs, collision, Nanite, texture compression, and folder structure.

What It Does:

- Recursively imports all files from a given folder into `/Game/AutoImports`. Creates a new folder if it doesn't exist.
- Automatically categorizes files into **Static Meshes** or **Textures**
- Applies import settings per asset type:
    - Nanite, LODs, Ray Tracing, Collision (for meshes)
    - Compression, Mip Settings, Texture Group (for textures)
- Determines asset category and LOD Groups from file/folder naming

How to Use:

Step 1: Create a new unreal Project or use an existing one.

Step 2: Place the assetImporter.py script inside "content\Python\".

Step 3: Launch the unreal project and open a python shell.

Step 4: execute command **import assetImporter**. (This will load the script in unreal)

Step 5: next run the command **assetImporter.importAssets(Full File Path)**

For example if the files are in C->MyAssets->Files, then

importAssets("C:/MyAssets/Files")

All assets will be imported, categorized, and configured automatically.

2. Unreal Python API Functions Used

Following are some API used and their function

Function	Purpose
`AssetToolsHelpers.get_asset_tools()`	To handle automated imports
`AutomatedAssetImportData()`	Represents the batch import configuration
`EditorAssetLibrary.list_assets()`	Scans imported assets, Requires path
`StaticMeshEditorSubsystem`	Core editing tool for mesh-specific properties
`get_editor_property` / `set_editor_property`	Used throughout for accessing specific property

Specific Feature APIs:

- **Nanite & Auto LOD Distance**: `mesh.get_editor_property("nanite_settings")`, `set_lods()`
- **Collision Configuration**: `body_setup.default_instance.set_editor_property(...)`
- **Texture Compression & Grouping**: `TextureMipGenSettings`, `TextureCompressionSettings`, `TextureGroup`
- **Dynamic Grouping based on Folder Name**: `tex.get_path_name().lower()` used to infer character-related texture groups

3. Input Requirements & Configuration

Folder Structure:

All assets must reside under a `Files/` directory structure

This structure is used to determine asset grouping (e.g. `"Character"` or `"Props"`) and auto-assign LOD groups or texture groups.

Configuration Details:

Static Mesh Settings:

Setting	Value
Nanite	`Disabled`
Generate Lightmap UVs	`Enabled`
Min Lightmap Resolution	`256`
LOD Group	`From asset name`
Auto LOD Distance	`Disabled manually`
Ray Tracing	`Disabled`
Collision Profile	`BlockAllDynamic`
Collision Object Type	`WorldDynamic`
Collision Enabled	`QueryAndPhysics`

Texture Settings:

Type	**Compression**	**Mip Gen**	**Texture Group**
Diffuse	`TC_DEFAULT`	`Average`	`World` / `Character`
Normal	`TC_NORMALMAP`	`Sharpen0`	`WorldNormalMap` / `CharacterNormalMap`
ORM	`TC_MASKS`	`No Mipmaps`	`WorldSpecular` / `CharacterSpecular`

Common Texture Settings:

- **Lossy Compression**: `OodleRD0 20` (medium quality)
- **ASTC Quality**: `ASTC 8x8 block` for optimized GPU performance.

4. Performance Consideration

- Using If…….elif instead of If…....else if to reduce the check if initial conditions are met elif will not do additional checks in the same chain
- Avoiding unnecessary re-assignments via comparison logic (`if current != desired`)
- Using an additional function to increase reusability of code. `setTextureProperty(texture,MipGen,CompSetting,TexGroup)`
- Separate function for Static Mesh `processStaticMesh(meshes)` and Textures `processTextures(textures)` for modularity of code.
- Overwriting existing assets (`replace_existing=True`)
- Case-insensitive tag detection for naming (e.g. `_Arch`, `_Large`) using `str.lower()` method

5. Design Decisions

- **Fully Automated Import**
  Using `unreal.AutomatedAssetImportData` reduces the input requirement during the import and gives the script freedom to configure each asset uniquely.

- **Use of folder-based classification**
  Consistent folder naming (e.g., "Character", "Props") allows scalable logic without manual input. It also asserts folder hierarchy.

- **Additional checks to avoid re-assignment**
  The script checks if existing properties are the same as what we want to assing. This can increase the performance of script especially when reimporting. There is an additional action that Overwrites existing assets (`replace_existing=True`)

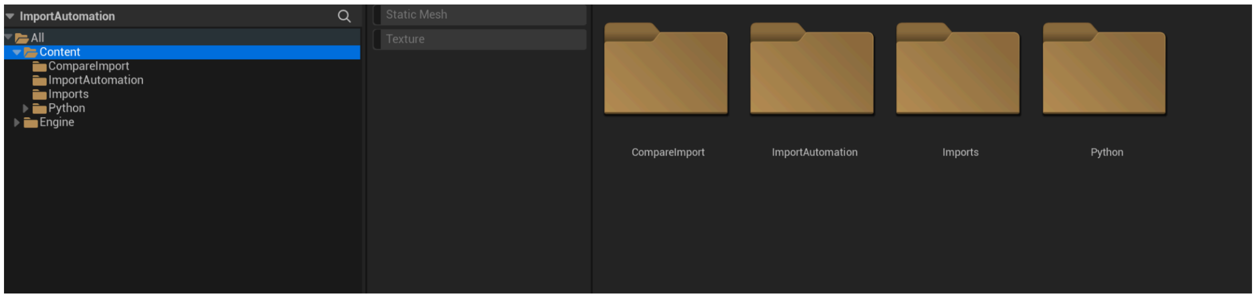
- **Classifying textures via name**
  File naming conventions (e.g., `_N`, `_D`, `_ORM`) are a standard for big projects. This also forces best practices which can be extremely helpful later in the project.

- **Setting `Sharpen0` and `No Mipmaps` for Normal and ORM maps**
  This is based on the research and explanations found in this topic say these MipGen Setting works best for these maps. Also `Mask(no SRGB)` compression setting for ORM.

- **Wrapping common `set_editor_property` assignment inside other function**
  This increases code reusability and does all the comparison and assigning of property inside another function.

6. Screenshots

Before Import

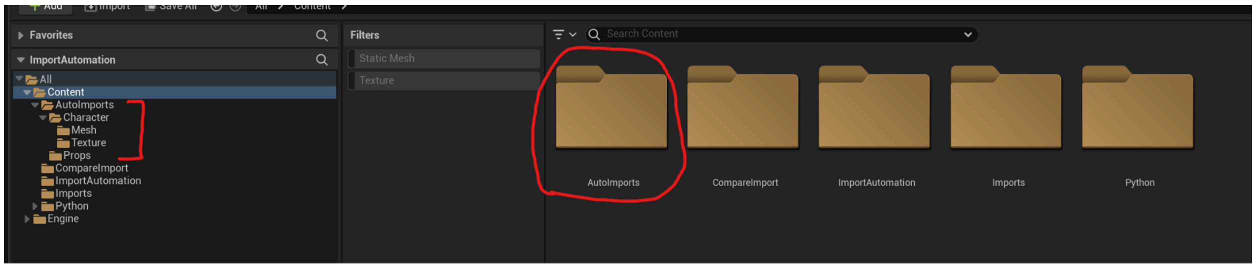


ImportAutomation
- All
  - Content
    - CompareImport
    - ImportAutomation
    - Imports
    - Python
  - Engine

Static Mesh
Texture

CompareImport ImportAutomation Imports Python

After Import



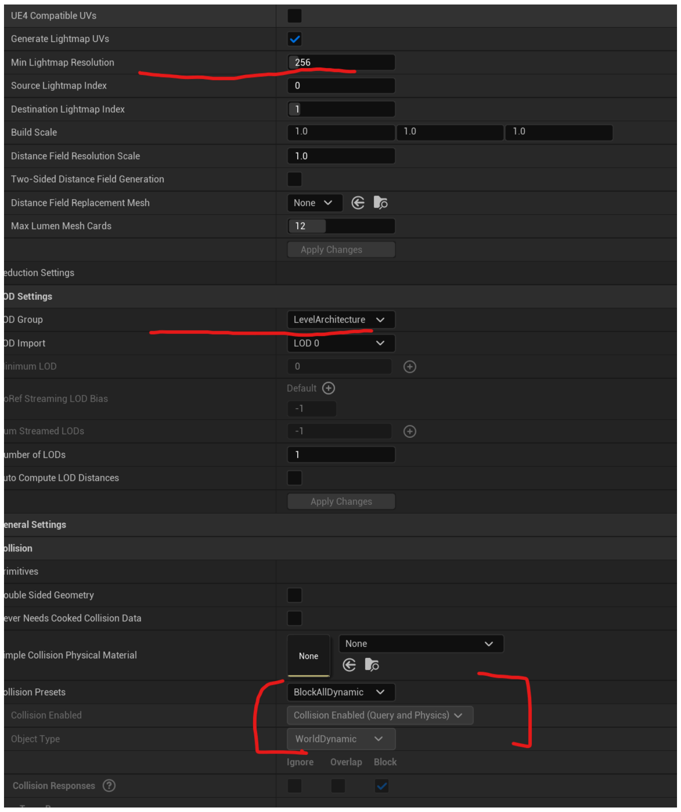
Favorites

ImportAutomation
- All
  - Content
    - AutoImports
      - Character
        - Mesh
        - Texture
      - Props
    - CompareImport
    - ImportAutomation
    - Imports
    - Python
  - Engine

Filters
Static Mesh
Texture

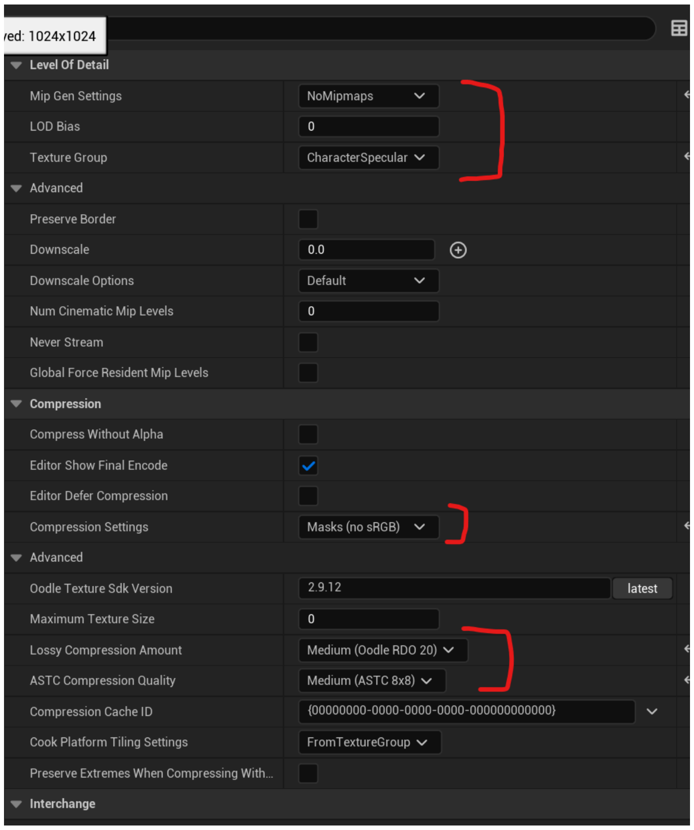
AutoImports CompareImport ImportAutomation Imports Python

Static Mesh Custom Properties



UE4 Compatible UVs	☐
Generate Lightmap UVs	✓
Min Lightmap Resolution	256
Source Lightmap Index	0
Destination Lightmap Index	1
Build Scale	1.0 1.0 1.0
Distance Field Resolution Scale	1.0
Two-Sided Distance Field Generation	☐
Distance Field Replacement Mesh	None
Max Lumen Mesh Cards	12
	Apply Changes
eduction Settings	
OD Settings	
OD Group	LevelArchitecture
OD Import	LOD 0
inimum LOD	0
oRef Streaming LOD Bias	Default -1
um Streamed LODs	-1
umber of LODs	1
uto Compute LOD Distances	☐
	Apply Changes
eneral Settings	
ollision	
rimitives	
ouble Sided Geometry	☐
ever Needs Cooked Collision Data	☐
imple Collision Physical Material	None
ollision Presets	BlockAllDynamic
Collision Enabled	Collision Enabled (Query and Physics)
Object Type	WorldDynamic
	Ignore Overlap Block
Collision Responses	☐ ☐ ✓

ORM Texture Inside Character Folder



ved: 1024x1024

Level Of Detail	
Mip Gen Settings	NoMipmaps
LOD Bias	0
Texture Group	CharacterSpecular
Advanced	
Preserve Border	☐
Downscale	0.0
Downscale Options	Default
Num Cinematic Mip Levels	0
Never Stream	☐
Global Force Resident Mip Levels	☐
Compression	
Compress Without Alpha	☐
Editor Show Final Encode	✓
Editor Defer Compression	☐
Compression Settings	Masks (no sRGB)
Advanced	
Oodle Texture Sdk Version	2.9.12
Maximum Texture Size	0
Lossy Compression Amount	Medium (Oodle RDO 20)
ASTC Compression Quality	Medium (ASTC 8x8)
Compression Cache ID	{00000000-0000-0000-0000-000000000000}
Cook Platform Tiling Settings	FromTextureGroup
Preserve Extremes When Compressing With...	☐
Interchange	