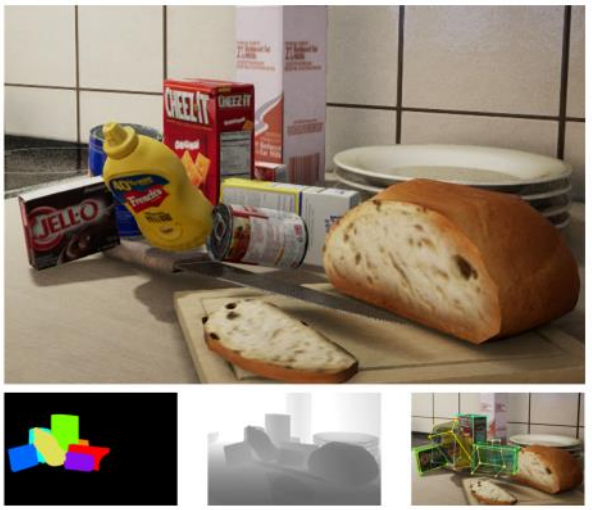
NVIDIA Deep learning Dataset Synthesizer (NDDS) Documentation

Thursday, June 21, 2018 10:13 AM

NDDS is a UE4 plugin from NVIDIA to empower computer vision researchers to export high-quality synthetic images with metadata. NDDS supports images, segmentation, depth, object pose, bounding box, keypoints, and custom stencils. In addition to the exporter, the plugin includes different components for generating highly randomized images. This randomization includes lighting, objects, camera position, poses, textures, and distractors, as well as camera path following, and so forth. Together, these components allow researchers to easily create randomized scenes for training deep neural networks.



Example of an image generated using NDDS, along with ground truth segmentation, depth, and object poses.

Downloading

This repository uses gitLFS -- DO NOT DOWNLOAD AS .ZIP

First, install git LFS (large file storage): https://help.github.com/articles/installing-git-large-file-storage/, then lfs clone.

Motivation

Training and testing deep learning systems is an expensive and involved task due to the need for hand-labeled data. This is problematic when the task demands expert knowledge or not-so-obvious annotations (e.g., 3D bounding box vertices). In order to overcome these limitations we have been exploring the use of simulators for generating labeled data. We have shown in [1,2] that highly randomized synthetic data can be used to train computer vision systems for real-world applications, thus showing successful domain transfer.

Citation

If you use this tool in a research project, please cite as follows:

```
@misc{to2018ndds,
 author = {Thang To and Jonathan Tremblay and Duncan McKay and Yukie Yamaguchi and Kirby Leung and Adrian Balanon and Jia Cheng and William Hodge and Stan Birchfield},
 title = {{NDDS}: {NVIDIA} Deep Learning Dataset Synthesizer },
 note = {\url{https://github.com/NVIDIA/Dataset_Synthesizer}},
 year = 2018
}
```

References

[1] J. Tremblay, T. To, A. Molchanov, S. Tyree, J. Kautz, S. Birchfield. Synthetically Trained Neural Networks for Learning Human-Readable Plans from Real-World Demonstrations. In International Conference on Robotics and Automation (ICRA), 2018.

[2] J. Tremblay, T. To, S. Birchfield. Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation. CVPR Workshop on Real World Challenges and New Benchmarks for Deep Learning in Robotic Vision, 2018.

Health and Safety Information

Thursday, June 28, 2018 11:23 AM

Some individuals may experience a seizure or other complications when exposed to certain visual images, including flashing lights or patterns that may appear. If you or any of your relatives have a history of seizures or epilepsy, consult a doctor before using Nvidia Deep Learning Dataset Synthesizer. Even people who have no history of seizures or epilepsy may have an undiagnosed condition that can cause these “photosensitive epileptic seizures.” Symptoms may include, among others: (1) Lightheadedness, (2) Altered vision, (3) Eye or face twitching, (4) Involuntary movements, (5) Convulsions, (6) Loss of awareness, (7) Confusion, (8) Disorientation, (9) Nausea
If you experience any of these problems immediately stop using Nvidia Deep Learning Dataset Synthesizer and consult a physician. Parents should monitor and ask their children about the above symptoms - children and teenagers may be more likely than adults to experience these symptoms. You may be able to reduce the risk of photosensitive epileptic seizures by taking the following precautions:
>> Do not play when you are drowsy, fatigued or ill.
>> Do not use the Nvidia Deep Learning Dataset Synthesizer for extended periods of time.

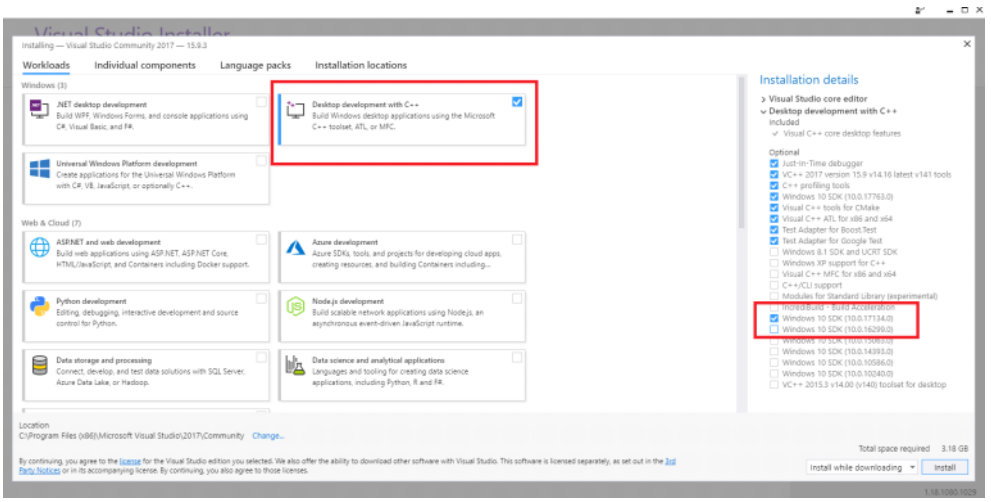
Installing Nvidia Deep Learning Dataset Synthesizer

Monday, June 25, 2018 5:09 PM

Windows Specific:

First, install Visual Studio.

a. If you are installing Visual Studio or Visual Studio Community, use the 2017 version located here . (Other versions may not be compatible.)
   i. We successfully tested NDDS using Visual Studio 2017 version 15.9.
b. When installing Visual Studio, you will need to perform a **CUSTOM INSTALL**. Make sure the following are selected
   - Desktop development with C++
   - Latest Windows 10 SDK
   - Windows 8.1 SDK (if you are still on Windows 8.1)

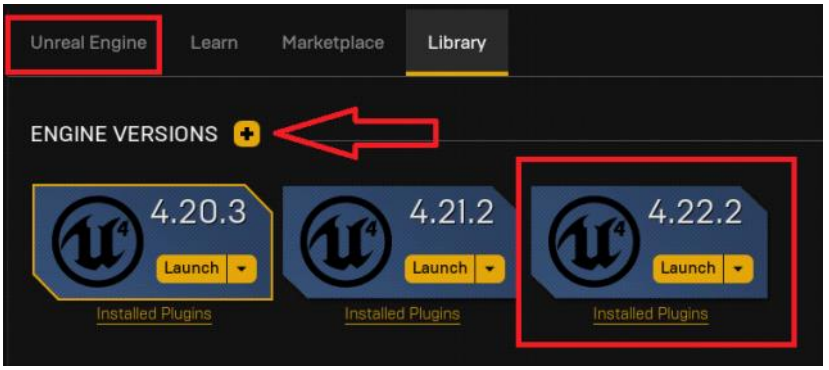


Download the following:

a. Epic's Launcher so that you can install Unreal Engine.
   i. Due to Unreal's requirements, please always ensure that you create your directory starting with a letter and not a number.
b. Installation instructions are found here .

1. Let's start with Unreal. Once you have downloaded Epic Launcher, install it and create an Epic account.

2. Once installed start the **Launcher**, login with your account and go to the **Unreal Engine** tab on the upper left corner of the Launcher. From here, make sure you have **Library** highlighted and then click the " **+** " icon.
   a. A pull down menu will appear with various engine versions. Please select **Version 4.22.2 or 4.22.3**

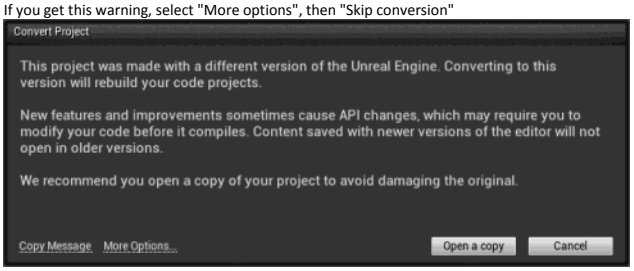


3. With the engine is installed, start the **Editor** by pressing the **Launch** button. You will need to start the **Editor** at least once so that it sets file associations. Creating a blank default project is good enough.

4. Close the **Editor** and **Launcher** for now.

Linux Specific:

a. for Unreal Engine, refer to the UE4 website instructions: https://wiki.unrealengine.com/Building_On_Linux
   Note NDDS uses UE v**ersion 4.22.X,** with respect to the unreal first time setup documentation, use **git clone -b 4.22**
   See also https://docs.unrealengine.com/en-US/Platforms/Linux/GettingStarted
b. Note that NDDS currently supports only Ubuntu 16.04.4 LTS
c. We successfully tested NDDS using NVIDIA driver versions 387.34 and 390.67
d. If you get this warning, select "More options", then "Skip conversion"



Addendum re for NDDS-on-UE4.22 update:

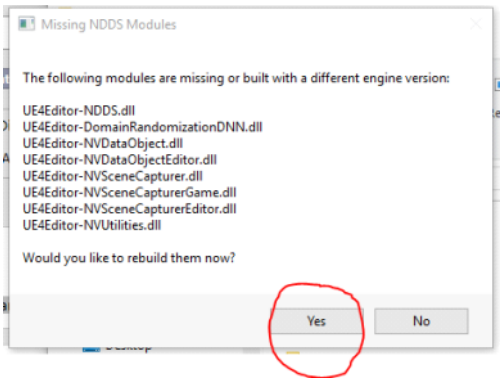
1. Tested with VS2017v15.9 (https://docs.microsoft.com/en-us/visualstudio/releasenotes/vs2017-relnotes), see also https://docs.unrealengine.com/en-us/Programming/Development/VisualStudioSetup
2. Tested with UE4.22.2, 4.22.3

Installation

1. **Download NDDS:**
   a. Ensure LFS is installed: **https://help.github.com/articles/installing-git-large-file-storage/**, and use **git lfs clone** rather than downloading as .zip
   Install the NDDS files for the dataset synthesizer.

2. Navigate to the directory containing the files and find **NDDS.uproject.** This should be under the **\Source** sub-directory.

3. **Run NDDS.uproject** and select "Yes" when it prompts you to rebuild binaries. The compilation will occur behind the scene and open the project wh en completed.



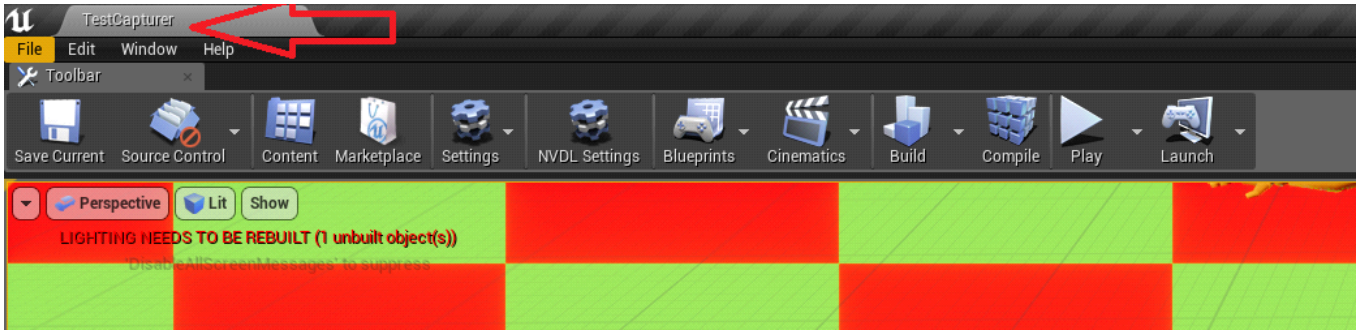
Note: **Nvidia Deep Learning Dataset Synthesizer** plugins may only be used within a project (game) -- hosting as engine plugins not yet supported.

Content Overview

Thursday, June 21, 2018 10:14 AM

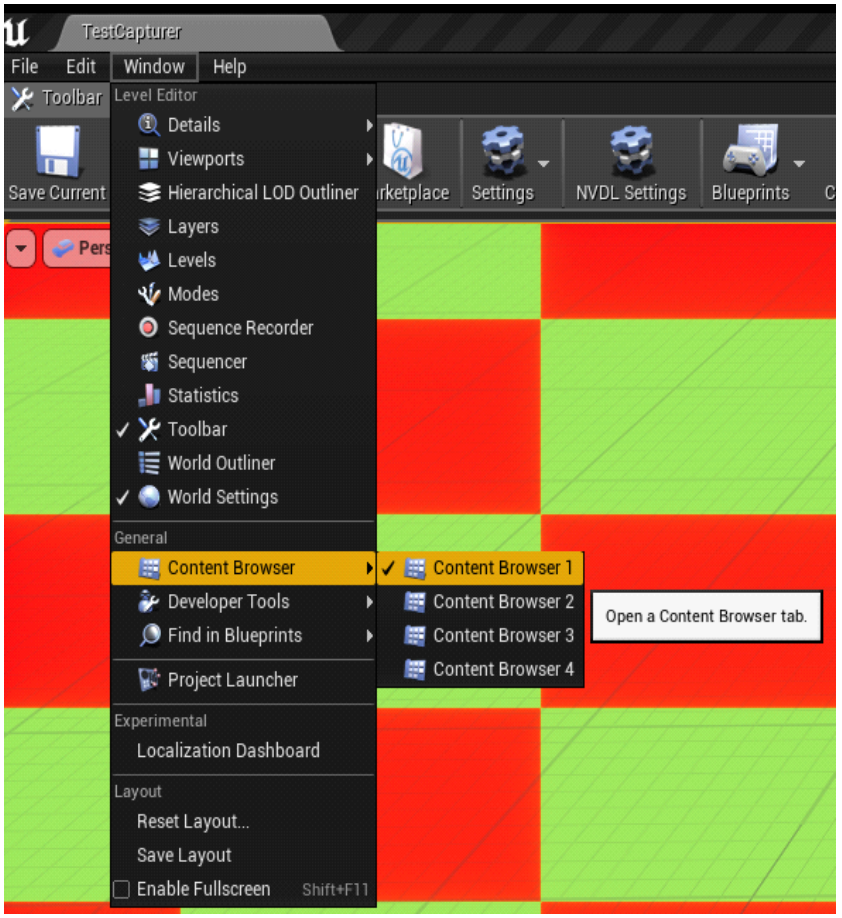
1. The **Nvidia Deep Learning Dataset Synthesizer** comes with demo content of assets to capture simulation data from the Unreal Game Engine. Please familiarize yourself with the [Unreal Editor Basics] so that you will be able to follow along the directions below.

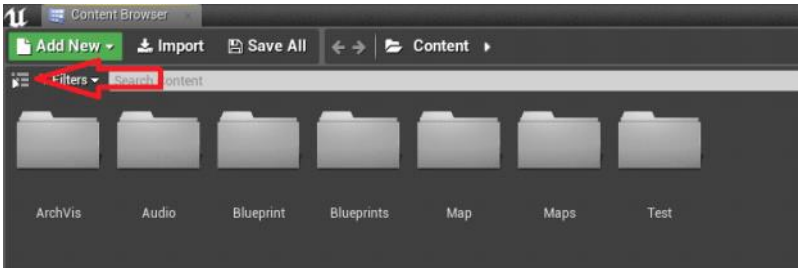
1. Once you open the **Unreal Editor** with the **NDDS.uproject**, a default level called ***TestCapturer*** will load as indicated at the top left hand corner of the 3D view port. This level has a sample scene with a basic simulation capture set up.

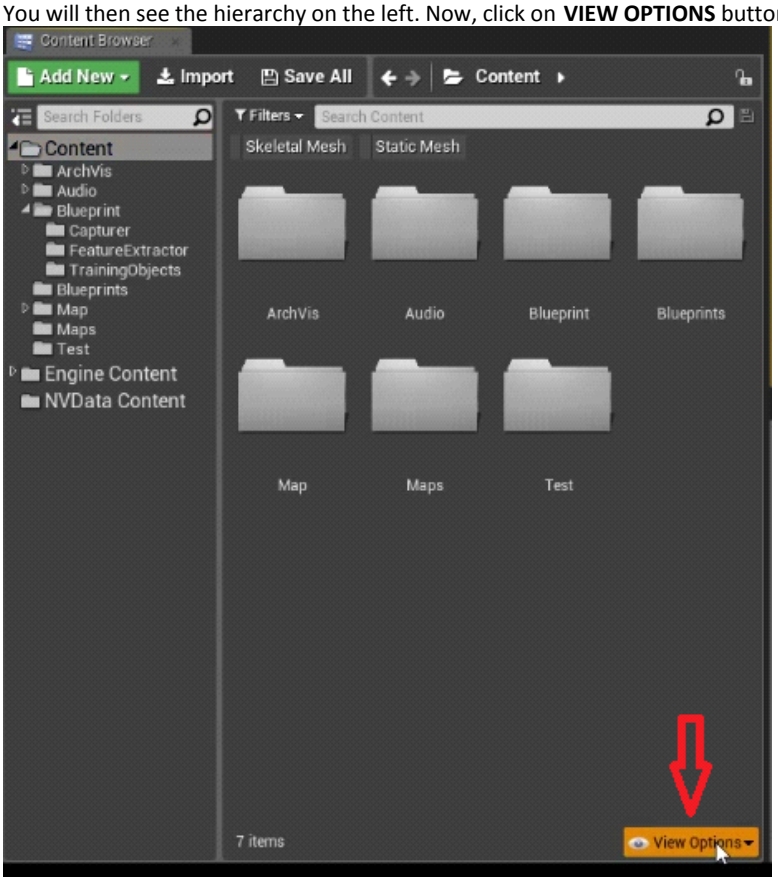


2. We will now go over the available content that makes up the loaded test scene.

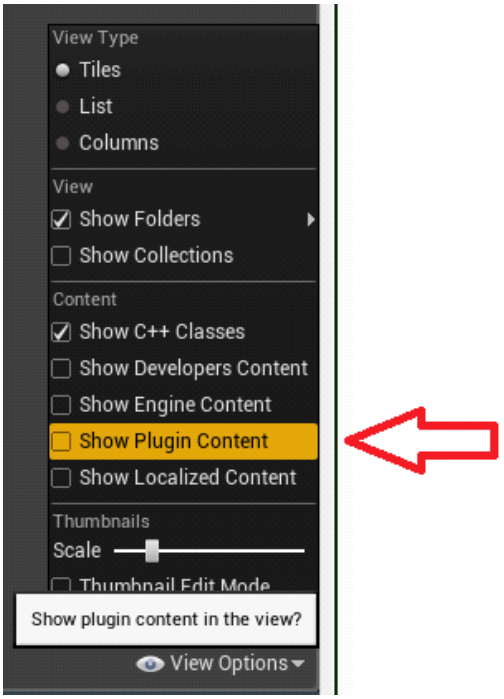
3. Before we start, look for a Tab or Window labelled **CONTENT BROWSER**.
   a. Note, if it is not readily visible, please enable it by going to the Unreal Menu bar and selecting **WINDOW > CONTENT BROWSER > CONTENT BROWSER 1**

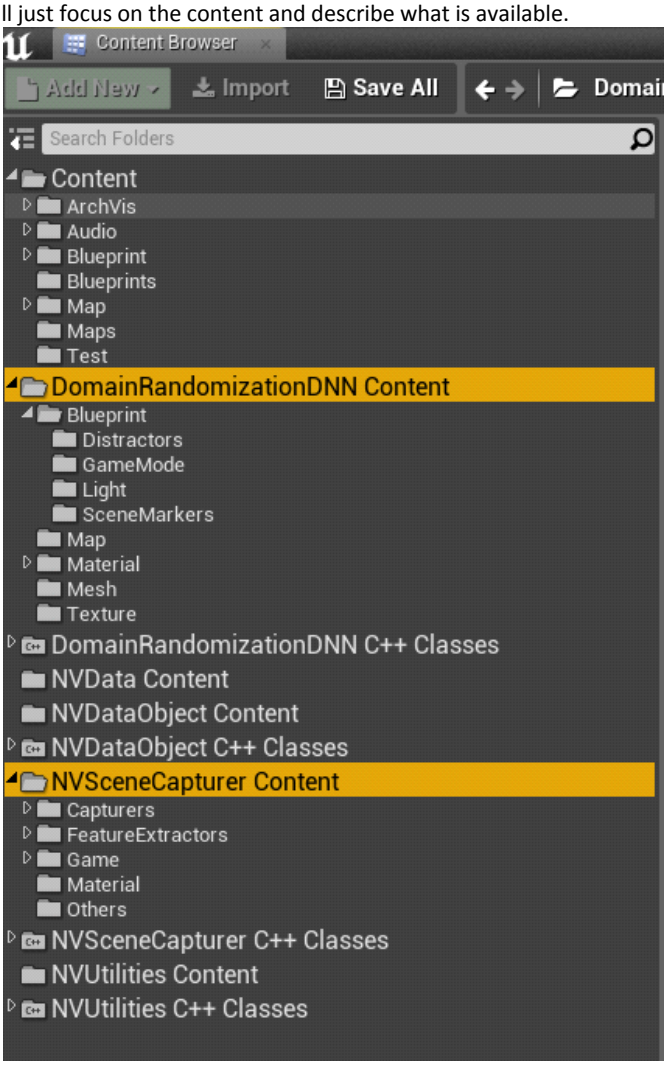


4. Once the **CONTENT BROWSER** is visible, make sure you can see the content directory hierarchy on the left hand side of the **CONTENT BROWSER** window.
    a. If it is not visible click the button here:
    ○ 

    b. You will then see the hierarchy on the left. Now, click on **VIEW OPTIONS** button.
    ○ 

    c. Enable the **SHOW PLUGIN CONTENT** checkbox so that additional folders will appear in the content directory on the left.

- 

5. Once plugin content is visible, you will see a **DomainRandomizationDNN CONTENT** and **NVSceneCapturer CONTENT** folder in the **Content** directory, as well as the source files. For now, we will just focus on the content and describe what is available.
   - 

NVSceneCapturer CONTENT

Capturers

1. This folder contains capturer camera objects (configuration + intrinsics) that can be placed in the scene:
    a. SceneCapturer_Zed
    b. SceneCapturer_IntelRealSense_SR300
    c. SceneCapturer_Simple

FeatureExtractors

Depth

This folder contains difference capturer **Absolute** and **Quantized** depth settings to be used on the Capturer Cameras.

Absolute

These absolute depth feature extractors capture the scene's depth in absolute value.

i. *FE_Depth_micron_32bits*: capture scene depth in micron units, using 32 bits. Max distance is 4km
ii. *FE_Depth_mm_16bits*: capture scene depth in mm units, using 16 bits. Max distance is 65m

Quantized

These depth feature extractors capture the scene's depth and quantize it into range [0, 1]

i. *FE_DepthQuantized_8bits*: capture and quantized scene depth using 8 bits - it map [0, 1] to [0, 255]. To convert the value from the grayscale use the formula: (PixelValue / 255.0) * MaxDepthDistance
ii. *FE_DepthQuantized_16bits*: capture and quantized scene depth using 16 bits - it map [0, 1] to [0, 65535]. This 16 bits quantized feature extractor is more accurate than the 8 bits but it's normally bigger. To convert the value from the grayscale use the formula: (PixelValue / 65535.0) * MaxDepthDistance

Segmentation

This folder contain feature extractor which segmenting the objects in the scene.

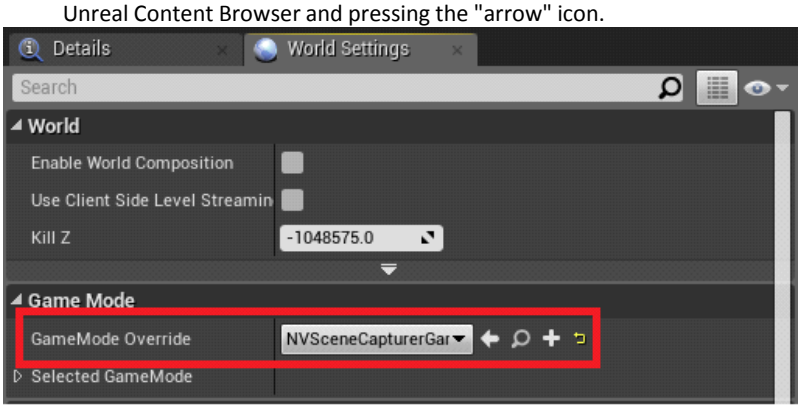
1. *FE_InstanceSegmentation*: capture object segmentation in 32 bits color mode - each object have its own id and color even if they are using the same 3d models.
2. *FE_ClassSegmentation*: capture the class segmentation of objects to grayscale pixel format - different objects which have same tags or using the same 3d models will share the same id.

Game

1. This folder contains Game Mode object and UI actors for the Capturer runtime interface.

    For the purposes of the capturer, Game Mode Override should always be set to "**NVSceneCapturerGameMode_BP.**" **IF IT IS NOT, go to the World Settings tab where you can "GameMode Override" in one of two ways:**

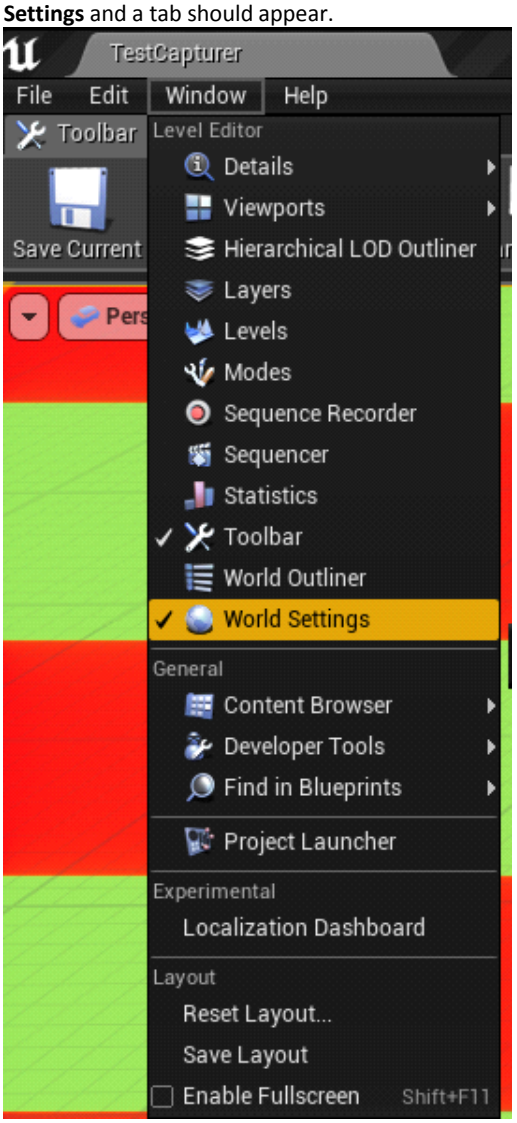
    i. **By pressing on the pull down menu next to "GameMode Override" and searching for "NVSceneCapturerGameMode_BP" in the list.**
    ii. **Selecting NVSceneCapturerGameMode_BP** in the Content Browser under "Dataset_Synthesizer/Source/Plugins/NVSceneCapturer/Content/Game/" in the Unreal Content Browser and pressing the "arrow" icon.



The Game Mode and UI objects are referenced in the **WORLD SETTINGS** of the game level space that is currently loaded.

i. **Please minimize changes to default UE world settings beyond what is documented here, unless you know what you are doing.**

ii. To see the **WORLD SETTINGS** tab, go to the File Menu under **Window > World Settings** and a tab should appear.



Material

- This folder contains the Materials used in the Feature Extractors.

DomainRandomizationDNN Content

BLUEPRINTS

1. Within the **Blueprint** folder under "*NDDS\Dataset_Synthesizer\Source\Plugins\DomainRandomizationDNN\Content\Blueprint*" you will find the following:
   a. **Distractors**
   b. **Game Mode**
   c. **Light**
      i. Contains light actors that can be placed in a scene that allow for scene lighting domain randomization.
   d. **Scene Markers**

2. For a more in-depth view of Unreal's Blueprint system please go to Blueprints Visual Scripting in the Unreal engine documentation.

Basic Export Example

Thursday, June 28, 2018 4:25 PM

In this section we explore the different components needed for doing simple exporting.

Please watch the video first.

VIDEO: https://github.com/NVIDIA/Dataset_Synthesizer/blob/master/Videos/tutorial_1.mp4

The steps are as follows:

1. Create a new scene
   File > New Level > VR-Basic
2. Make sure the **World Settings** is visible
   Window > World Settings
3. Add NDDS UI:
   In the **World Settings** set the **Game Mode Override** in **Game Mode** to *NVSceneCapturerGameMode_BP*
4. Add to the scene the *SceneCapturer_Simple*
   a. In **Content Browser**, under **Add New** click the little button to show the **sources panel (beside the Search Folders bar)**. Under **NVSceneCapturer Content** > **Capturers**, click and drag *SceneCapturer_Simple* onto scene
      i. SceneCapturer_Simple is a premade capturer (like a camera) with default feature extractors. It captures the following
         1) Object Data
         2) True Color
         3) Depth
         4) Instance Segmentation
         5) Class Segmentation
      ii. This was made for common use cases and the user is able to create custom capturers as needed.
   b. In the scene click on the camera to select the *SceneCapturer_Simple* . This will show many controls under **Details**. There are a lot of controls, but we will focus on the most important ones.
   c. Under **Details** > **Capture:**
      i. *Auto Start Capturing*: Check this to start exporting immediately upon pressing **Play**
      ii. *Max Number of Frames*: Specify how many frames you want to export
      iii. *Scene Data Handler|Save Path|Root Captured Path*: Specify the path where you want to save the exported data
   d. Under **Details** > **Feature Extraction**: This is where you can select which meta-data you are interested in. The Section "Feature Extraction Details" below describes the different choices you have there.
5. (Optional) Add to the scene the *SceneManager_BP*
   a. (This step is optional, for control of segmentation and more complex randomization.)
   b. In **Content Browser**, under **Add New** click the little button to show the **sources panel**. Under **DomainRandomizationDNN Content** > **Blueprint**, click and drag *DR_SceneManager_BP* onto scene
6. Add the component *NVCapturableActorTag* to the objects for which you want to export meta-data.
   a. In the scene click on an object to select it
   b. In **Details** > **Add Component**, type "tag", then select **NVCapturableActorTag**
   c. The section Feature Extraction Details gives more details on the meta-data.

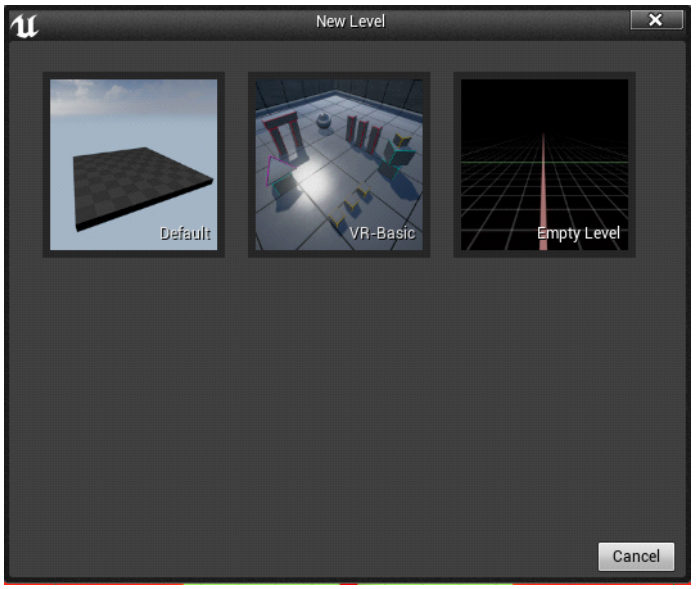
7. Press **Play** to start capturing
8. (If the mouse disappears, press Shift+F1 to make it visible again. This is a feature of UE4.)
9. Navigate to the folder specified above (see *Root Captured Path*) to find your exported synthetic data

Creating a new Domain Randomization map

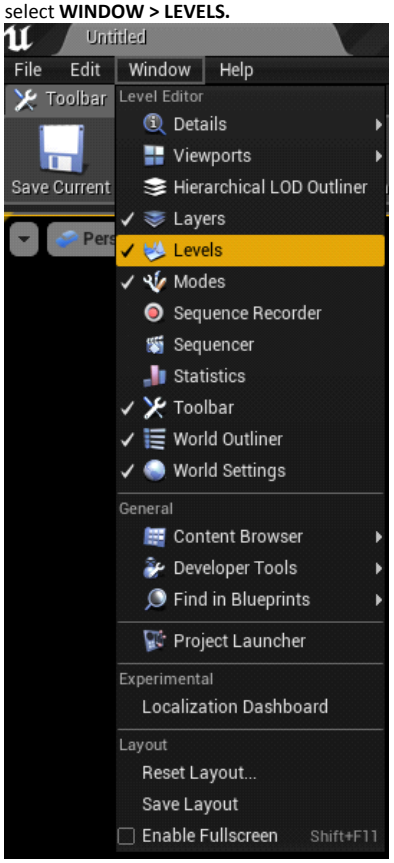
Monday, June 25, 2018 9:06 AM

Note: Before you proceed make sure your CONTENT BROWSER is set to view PLUGIN CONTENT. Directions can be found in the Content Overview page.

1. To create a new map for domain randomization, open the editor and go to **FILE > NEW LEVEL > EMPTY LEVEL.** Once you have selected "**Empty Level**", go ahead and save the new level (**FILE> SAVE CURRENT**). As a suggestion, the default project contains a directory location for levels under "**Content > Map**"

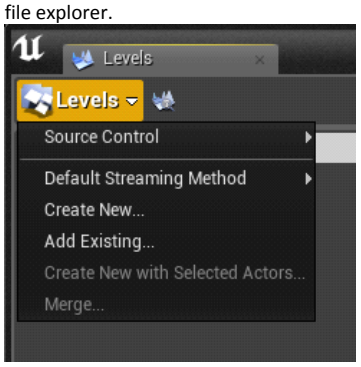


1. Load the **DRMainMap** level located in **/DomainRandomizationDNN CONTENT/Map/** by going to the **LEVELS** tab of Unreal. If the **LEVELS** tab is not visible, go to the File Menu bar of Unreal and select **WINDOW > LEVELS.**



2. Once you have the **LEVELS** tab visible, click **LEVELS > ADD EXISTING…** which will open an Unreal

file explorer.

- 

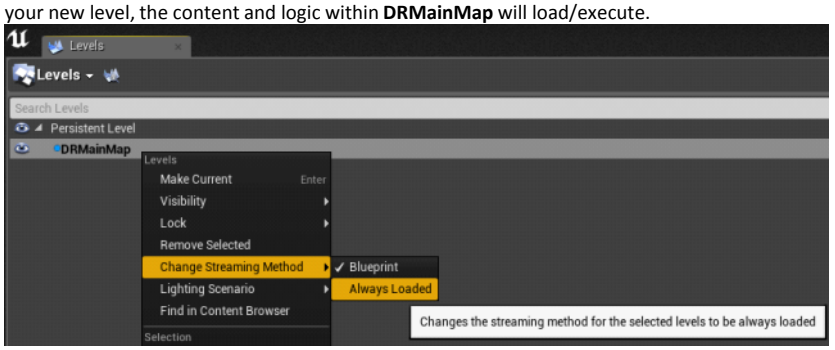
3. Navigate to **DomainRandomizationDNN Content\Map** and select **DRMainMap** and click **OPEN** on the lower right corner.

4. After adding the level, your **LEVELS** tab will look like this:

- 

5. The name of the level, **DRMainMap** has three states visible in the **LEVELS** tab. 1) It Is indented relative to "Persistent Level." 2) It has a blue dot as a prefix. 3) It is highlighted blue/bold. All these things define a state for the level.
    a. The indentation relative to Persistent Level means that **DRMainMap** is a "Sub-Level." The map you created represented as "Persistent Map" means that its content and logic is always loaded at runtime. The **DRMainMap** in this case is a level content reference so it is indented.
        i. For more on this concept visit the Unreal Level Streaming Documentation.
    b. The blue dot prefix denotes a load state for **DRMainMap**. A blue dot means that at runtime, the logic and content within **DRMainMap** is not loaded/executed unless the user adds some blueprint, code, or trigger volume logic.
    c. The highlight blue/bold text means that any actor, mesh/prop, that you add into the level scene will be instantiated and stored within **DRMainMap**.

6. In order to proceed, **DRMainMap** must be loaded. The quickest way to is to always load at the start of runtime. Right-click on **DRMainMap** in the levels tab. Go to and select **CHANGE STREAMING METHOD > ALWAYS LOADED.** This removes the blue dot meaning that when you run your new level, the content and logic within **DRMainMap** will load/execute.

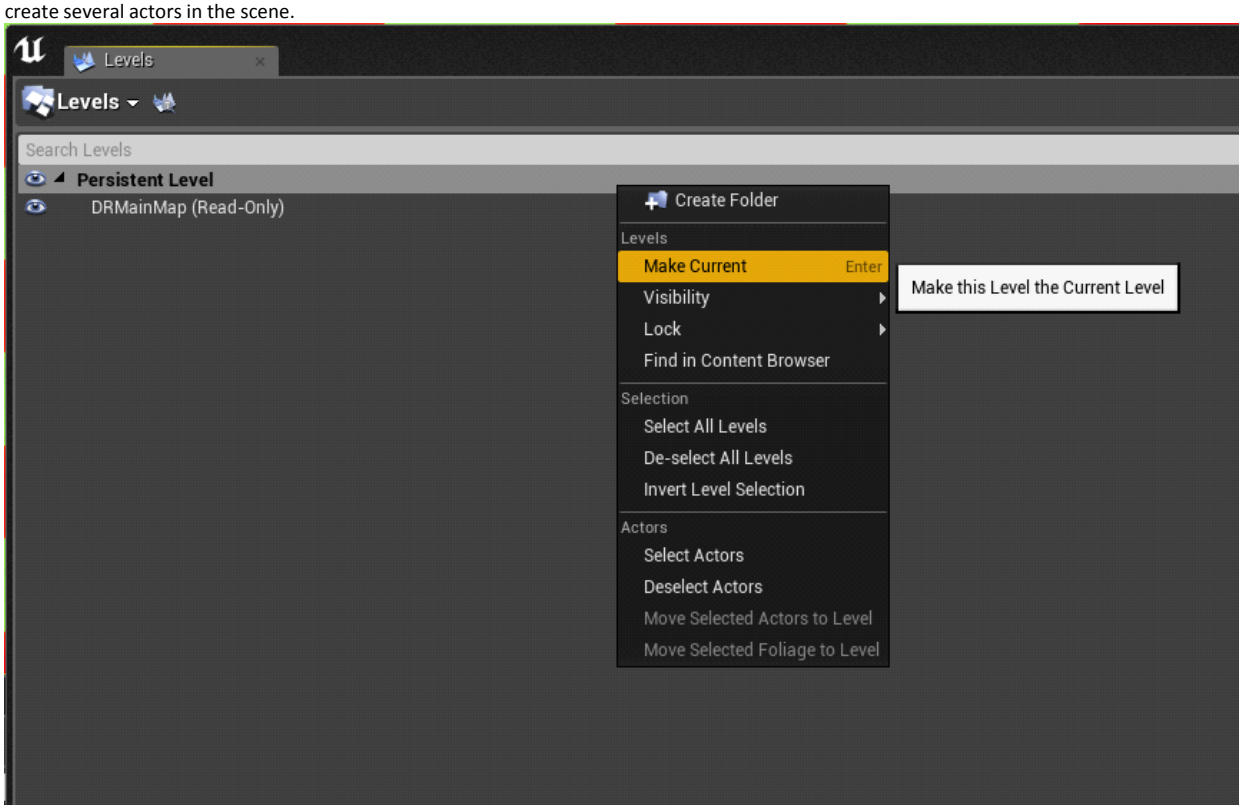


7. Notice there are now two "pencil" icons in the Levels tab corresponding to **Persistent Level** and **DRMainMap**.
    a. **Click the pencil icon on Persistent Map ONLY**.
        i. *Do not save DrMainMap, only save the map you created which in this view is called "Persistent Map.*
        ii. If you accidentally clicked on the "pencil" on DRMainMap, you should revert the file back via source control or from your original source.
    b. Persistent Map needs to be saved to make sure **DRMainMap's** reference load state is saved.

8. Once you save your Persistent Map, open it again via the **FILE > OPEN LEVEL** option. We do this as a precaution to prevent any accidental edits to **DRMainMap**. Note, you may need to navigate back

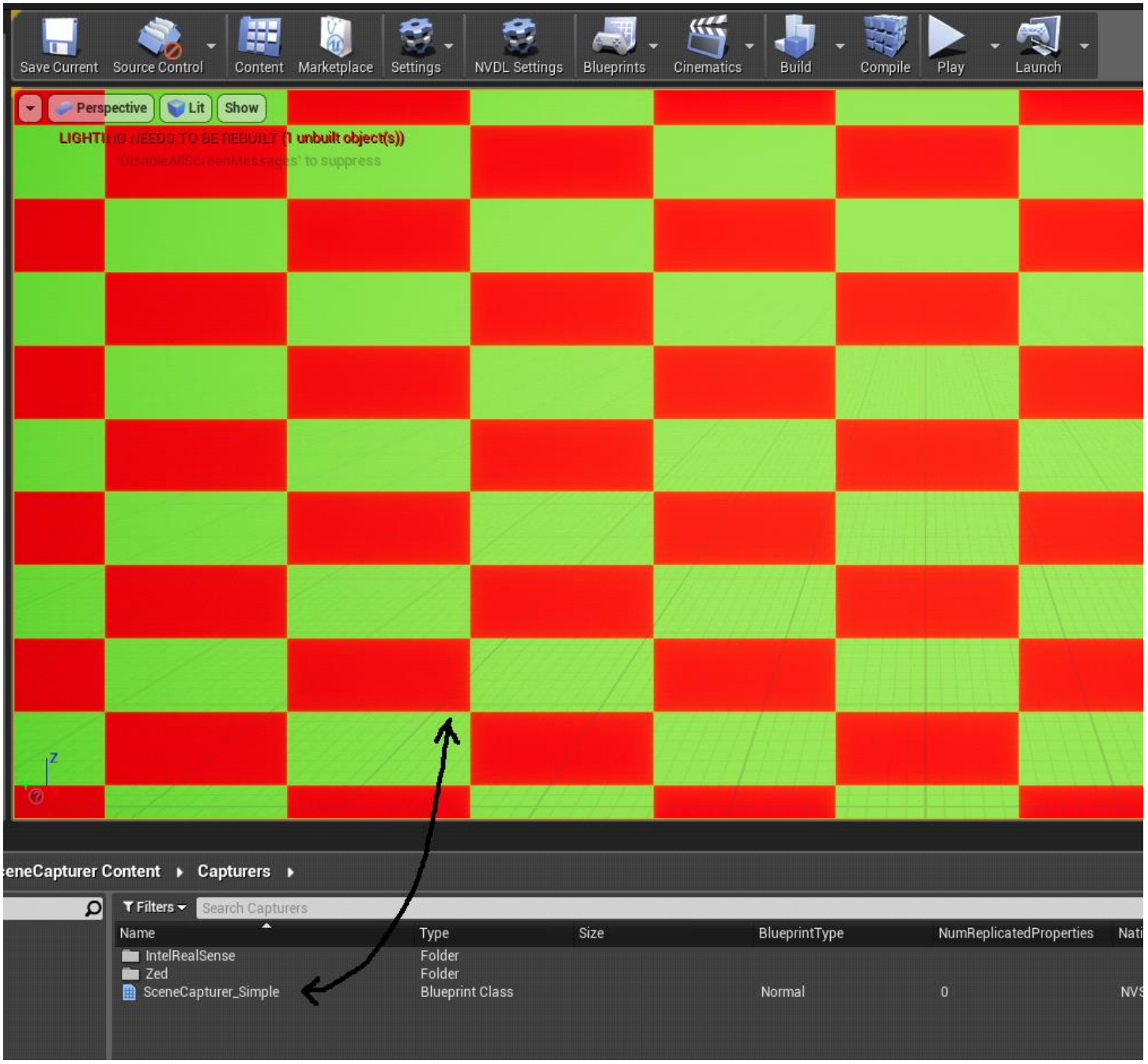
to the directory where you saved the file as the cursor is on the last location highlighted. You will get a pop-up asking if you want to **SAVE CONTENT**, ignore the dialog and press **DON'T SAVE.**

9. When the map opens, go to **LEVELS** tab and make sure that **PERSISTENT LEVEL is blue/bolded**, meaning that any object we create will be in the Persistent Level. If it is not already blue/bold, then right click on it and select **MAKE CURRENT**. We need to do this because we are about to create several actors in the scene.

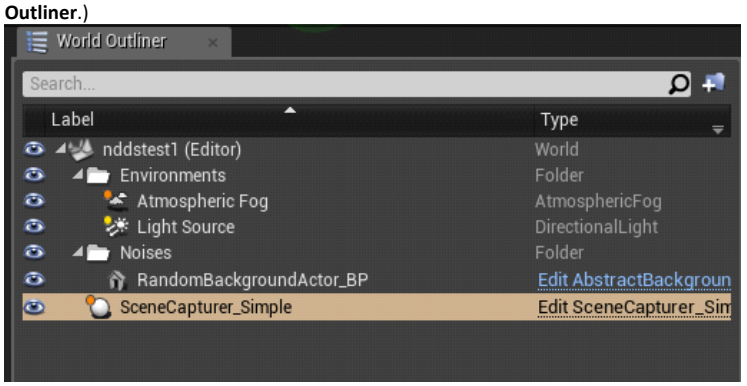


   a. Once this is done, you can go ahead and close this window.

10. With your map opened again, go to the **CONTENT BROWSER** and navigate to **NVSceneCapturer Content/Capturers/** and Click+Hold and drag **SceneCapturer_Simple** to the viewport to place it in the scene.

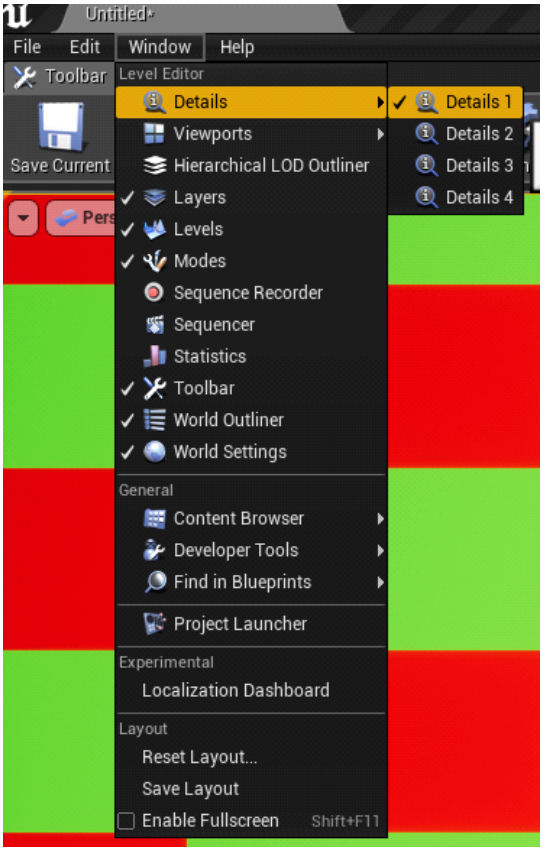
- 

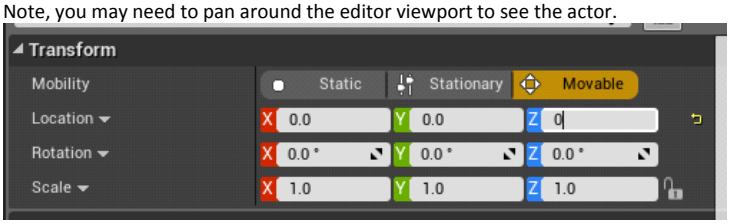
11. The capturer will have a visible camera representation in the space. Make sure it is selected in the scene. To select, either click on the camera in the viewport, or click **SceneCapturerSimple** in **World Outliner.** (If the latter is not visible, then from the **MENU** bar select **Window > World Outliner**.)

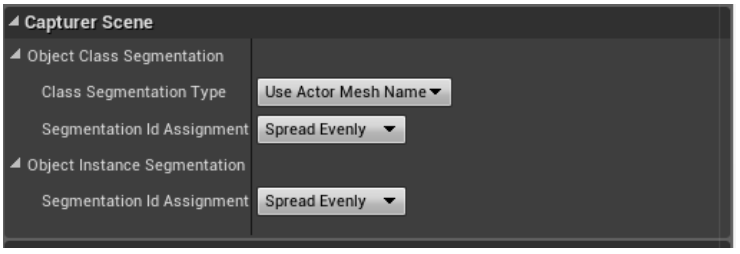
- 

a. Go to the **DETAILS** tab. The **DETAILS** tab will show you all the parameters of the selected object.
b. If the **DETAILS** tab is not visible go to the **MENU** bar and select **WINDOW > DETAILS >**

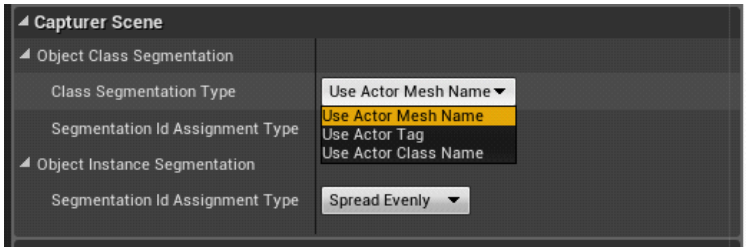
DETAILS 1



12. With the instance of the **SceneCapturer_Simple** actor selected in the 3d viewport go to your Details tab and find the **TRANSFORM > LOCATION** and set it to 0, 0, 0 (type the numbers in the field and hit Return) so the capturer is located at the root of the world. You can also click on the **YELLOW ARROW** to the right as this also resets to default.
    a. Note, you may need to pan around the editor viewport to see the actor.
    ○ 

13. Now, return to the **CONTENT BROWSER** and navigate to **/NVSceneCapturer Content/Others** and select **SceneManager_BP** and drag it into the scene. When you look at **DETAILS**, for this actor you will see that it has the following. You may need to press the little arrow key to unhide the folder details.
    ○ 

    a. Object Class Segmentation
        i. Use the actor's mesh name for its class type name, actor with no visible mesh will be ignored
            1) NOTE: Since each actor can have multiple mesh components, we only use the first valid mesh's name for the mask UseActorMeshName
        ii. Use the actor's Tags for its mask name, actor with no tags will be ignored
            1) NOTE: Since each actor can have multiple tags, we only use the first one in the Tags list for the mask UseActorTag
        iii. Use the actor's class (either C++ or blueprint) name for its class name. All the actor instances of the same class/blueprint will have the same mask UseActorClassName



b. Segmentation Id Assignment Type: See [Sequentially vs Spread evenly actor IDs]

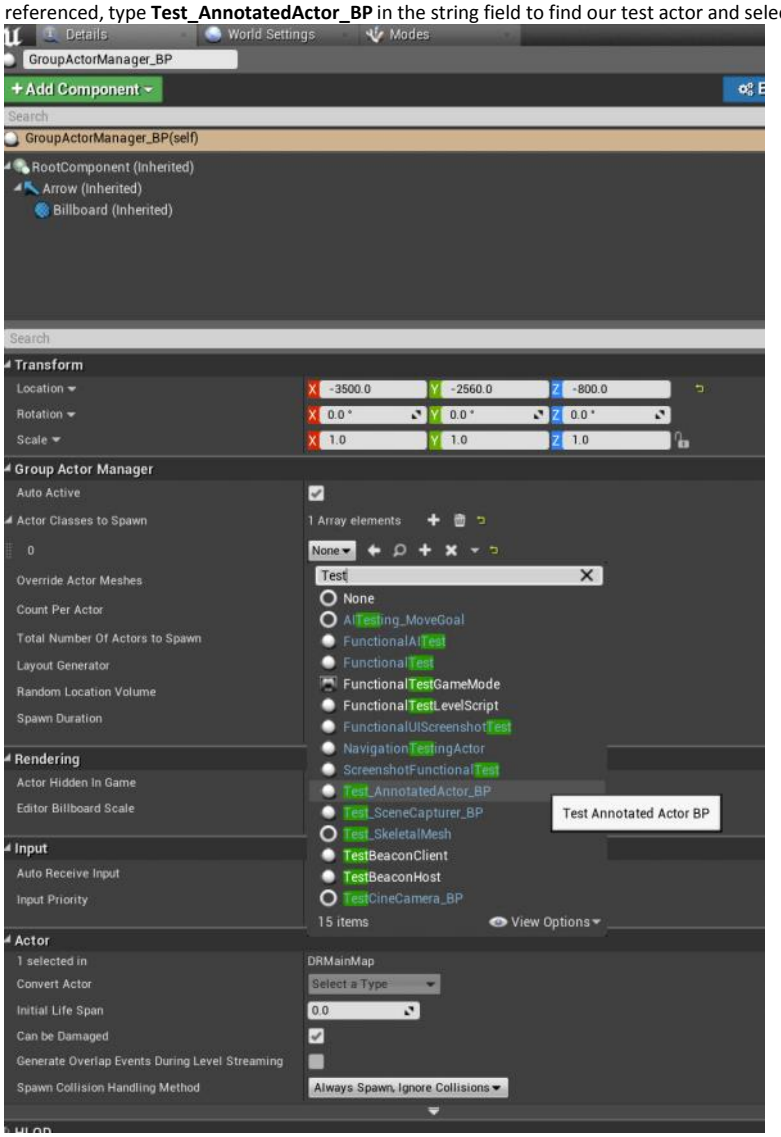
c. Object Instance Segmentation: Segmentation IDs are assigned per object instance rather than with respect to the object's class type.

d. Note, if you do not add in the **SceneManager_BP**, upon running the simulation, the **SceneManager_BP** will be automatically created.

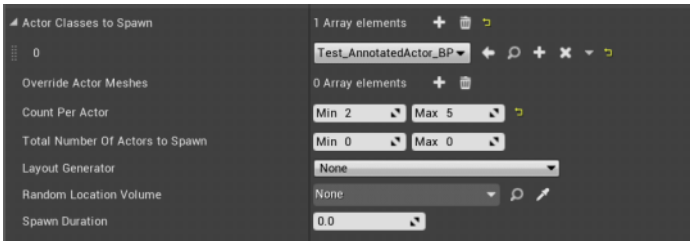
14. We also need to add the **GroupActorManager_BP** into the scene and it is located in **/DomainRandomizationDNN Content/Blueprint/**

15. Select your placed **GroupActorManager_BP** instance and go to **DETAILS**. Go to the **ACTOR CLASSES TO SPAWN** array and add an element by clicking the "+" icon.

16. A pulldown menu button called **NONE** appears, click on this to show the valid objects that can be referenced, type **Test_AnnotatedActor_BP** in the string field to find our test actor and select it.

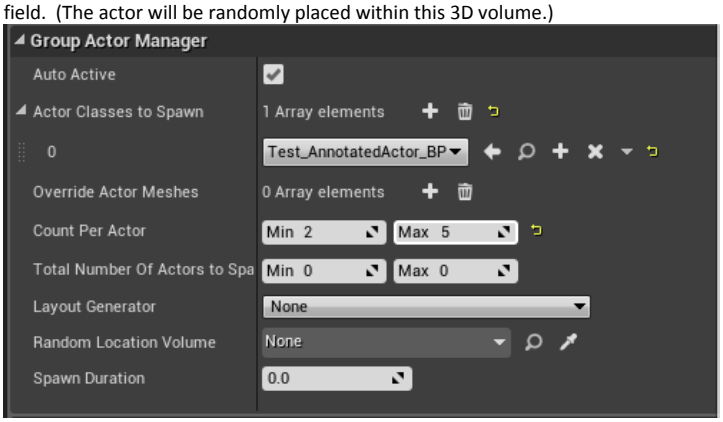


a. Selected, it should look like this:

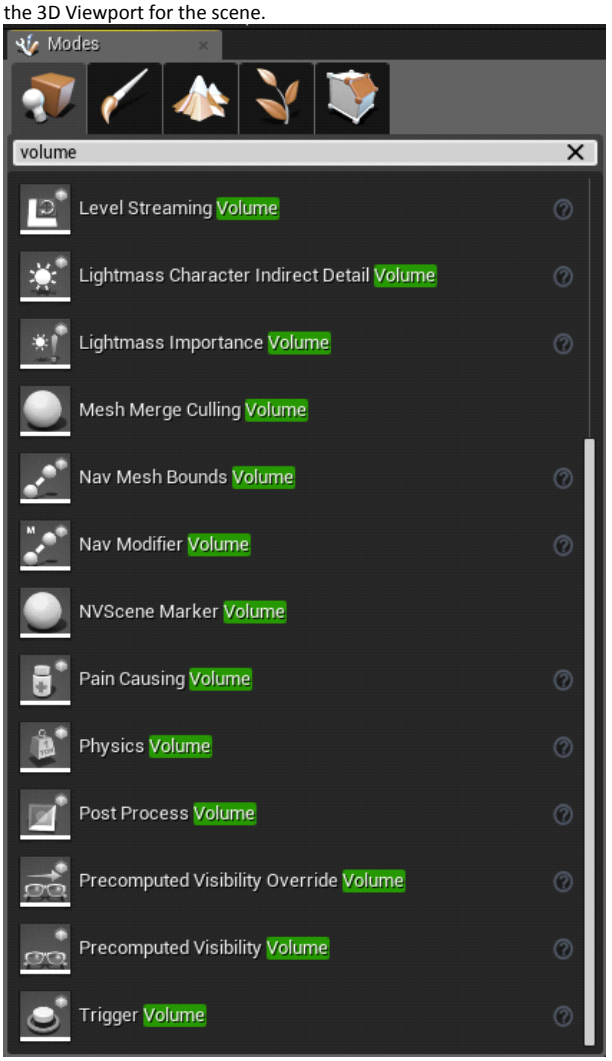


17. As an example, set **COUNT PER ACTOR** to **Min of "2" and Max of "5."**

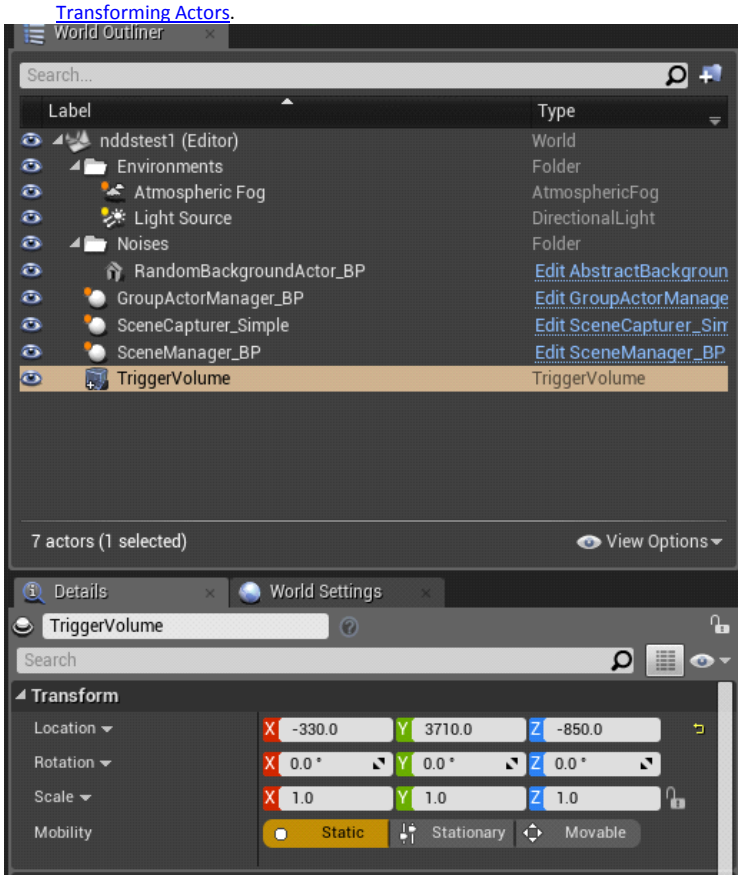
18. Now, we need to add a 3D volume to the scene to reference in **RANDOM LOCATION VOLUME** field. (The actor will be randomly placed within this 3D volume.)



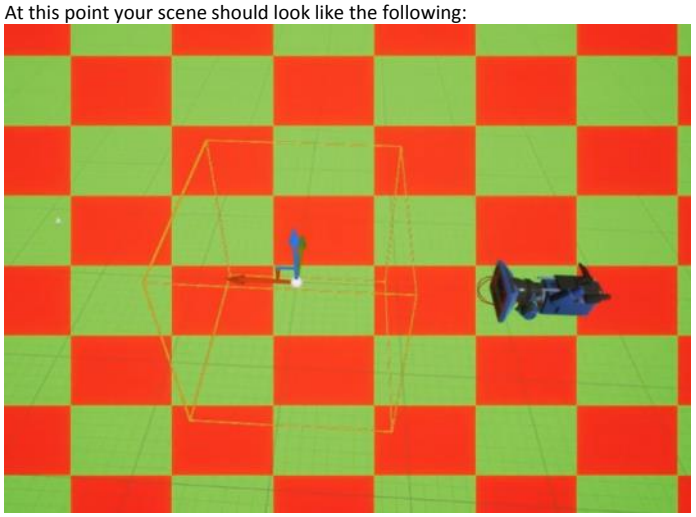
19. To do this go to the Unreal **MODES** tab. If it is not visible, go to **WINDOW > MODES.** Once the tab is accessible, type "volume" in the string search and Click+Hold and drag **TRIGGER VOLUME** into the 3D Viewport for the scene.



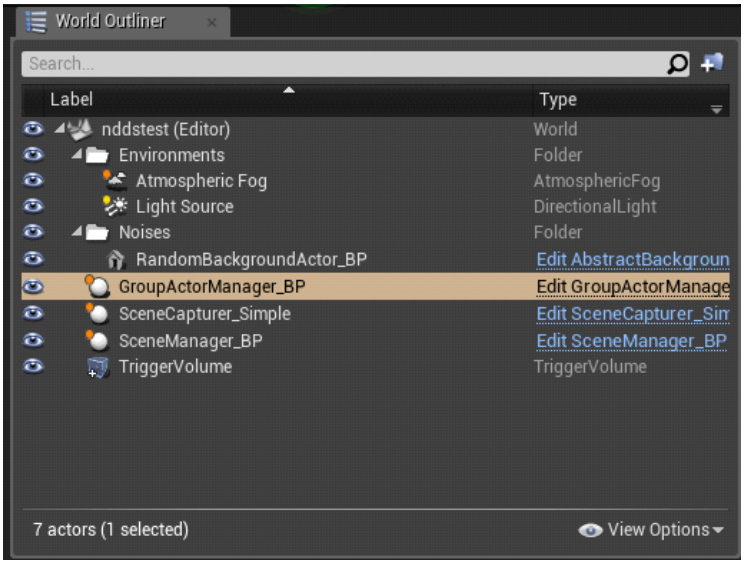
20. Go to the **DETAIL** tab of the **Trigger Volume** instance you created. Set its **TRANSFORM > LOCATION** to 0,0,0 and drag it in front of the "**camera**" actor (**SceneCapturer_Simple**) so that it is visible to the camera (see below).
    a. If you need a refresher on manipulating objects in Unreal's 3D viewport refer to Transforming Actors.



21. At this point your scene should look like the following:

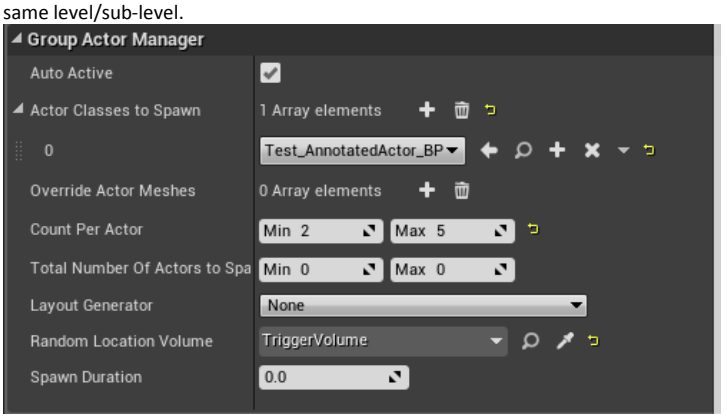


22. ***RECOMMENDED**: For clarity you can select the **GroupActorManager_BP** in the **WORLD OUTLINER** and rename it to "**Group_TrainingObjects**" by pressing F2 to edit the name. Note this changes the name of your instance of GroupActorManager_BP.*
    a. ***To find the WORLD OUTLINER go to WINDOW > WORLD OUTLINER.** The **WORLD OUTLINER** is the list of all the actors/objects/meshes in your level and provides a quick way to select objects without having to physically find them in the scene.*



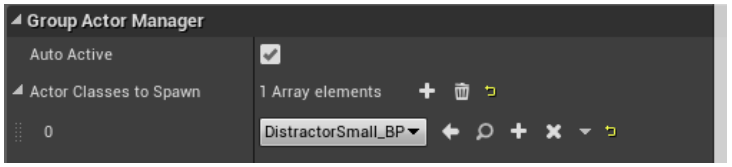
23. Now that you have a volume in position within the scene, you can go back to the **GroupActorManager_BP** (or **Group_TrainingObjects** if you renamed it) and find **RANDOM LOCATION VOLUME** (under **Details > Group Actor Manager**) and set it to the volume you placed in the scene.

24. To do this either, select the dropper icon and select your volume (in this case, **TriggerVolume**) in the 3D viewport. Or click the empty field pulldown bar and select the volume. Unreal usually creates lists for valid selections.
    a. Note: If you followed this tutorial up to this point, you should not have any issues selecting the volume with the dropper. However, if you do, it is likely that your actor instances in the scene do not reside in the same level or sub-level. In our case, we are trying to reference a Trigger Volume in the **GroupActorManager_BP**.
        i. Please refer to Step 7 or read "Moving Actors Between Levels" section in the Managing Multiple Levels documentation.
    b. In Unreal, having an actor reference another actor must be within the same level. There are complex ways to mitigate this but usually objects referencing each other have to be in the same level/sub-level.



25. We will now set up the Distractors:
    a. In the **WORLD OUTLINER** select the instance of **GroupActorManager_BP (Group_TrainingObjects if renamed)** and ***Ctrl+C*** then ***Ctrl+V*** to copy it. Notice it will have a suffix of "**2**"
    b. Rename it to "**Group_Distractors**"

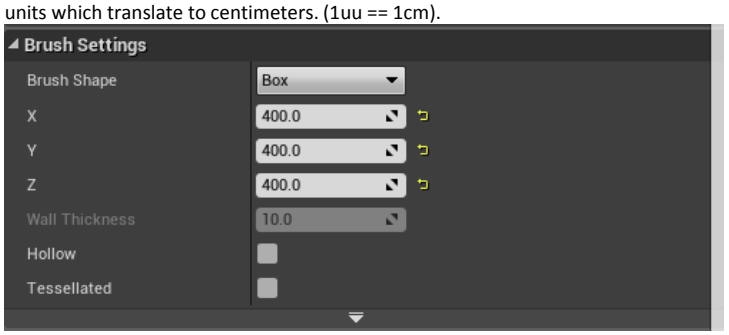
26. In the **DETAILS** for the copied/renamed object go to the **GROUP ACTOR MANAGER** section and replace **ELEMENT 0** of Actor Classes to Spawn to "**DistractorSmall_BP**". Click on the pulldown menu for the option.



27. In the same section, set **Min and Max of Count Per Actor** to **40** to **60** respectively.

28. Below that, set **Min and Max TOTAL NUMBER OF ACTORS TO SPAWN** to **40** and **60** as well.

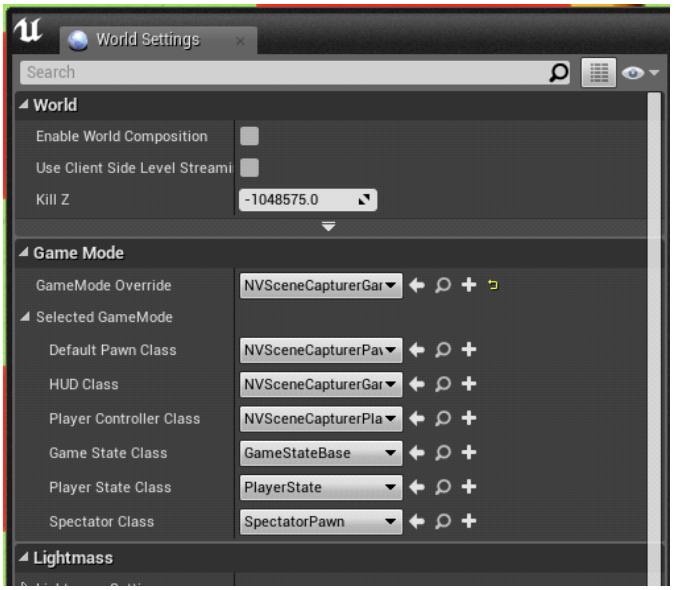
29. Now, in the **World Outliner**, select the **TriggerVolume**, copy and paste it. Now change the size of this second volume (**TriggerVolume2**) by going to **DETAILS > BRUSH SETTINGS** and type the new dimensions in X, Y, Z. In our sample video we set it to 400uu, 400uu, 400uu. Note "uu" is Unreal units which translate to centimeters. (1uu == 1cm).



30. Once you have resized the second volume (**TriggerVolume2**) go to the **WORLD OUTLINER** and select the **Group_Distractors** and within its **DETAILS** go to **Group Actor Manager > Random Location Volume** to your larger volume. If you have kept the original naming, it should be **TriggerVolume2.**

31. We now need to set the **GAME MODE**. To do this go to the **WORLD SETTINGS** tab. (If not visible go to menu bar **WINDOW > WORLD SETTINGS**.)

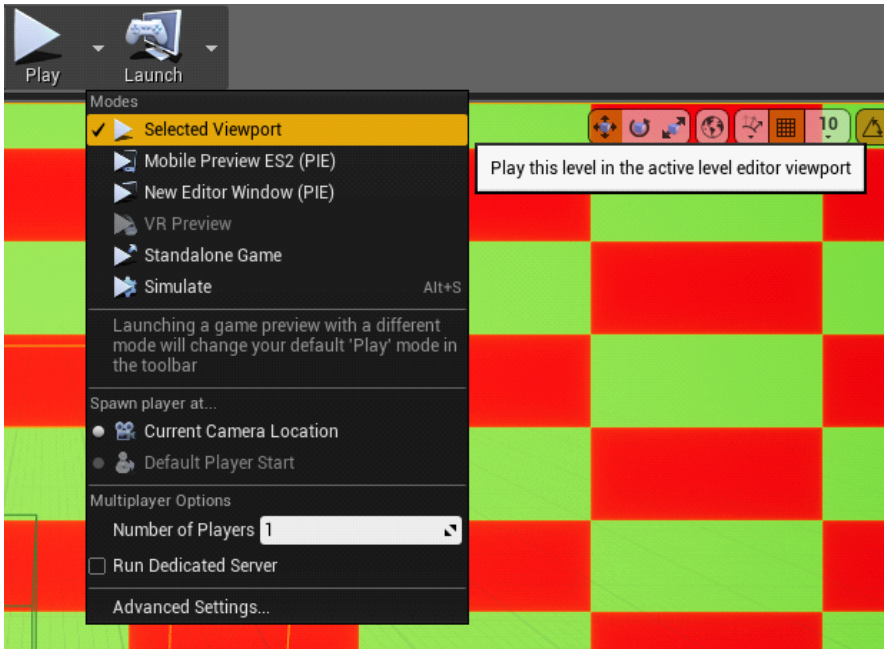
32. Within World Settings tab go to the **GAME MODE** section of **WORLD SETTINGS**, and set **GameMode Override** to **NVSceneCapturerGameMode_BP** by doing a string search or finding it in the list. It should fill in the "**Selected GameMode**" fields and look like this:



33. Save your progress by saving "**Persistent Map**" in level or Ctrl+S or **FILE MENU > SAVE**.

34. Before proceeding to the next step, note that when you press **PLAY,** the simulation will cause a rapid change to the display for the duration and is not intended for direct viewing. Please refer to the **HEALTH AND SAFTEY INFORMATION** section.

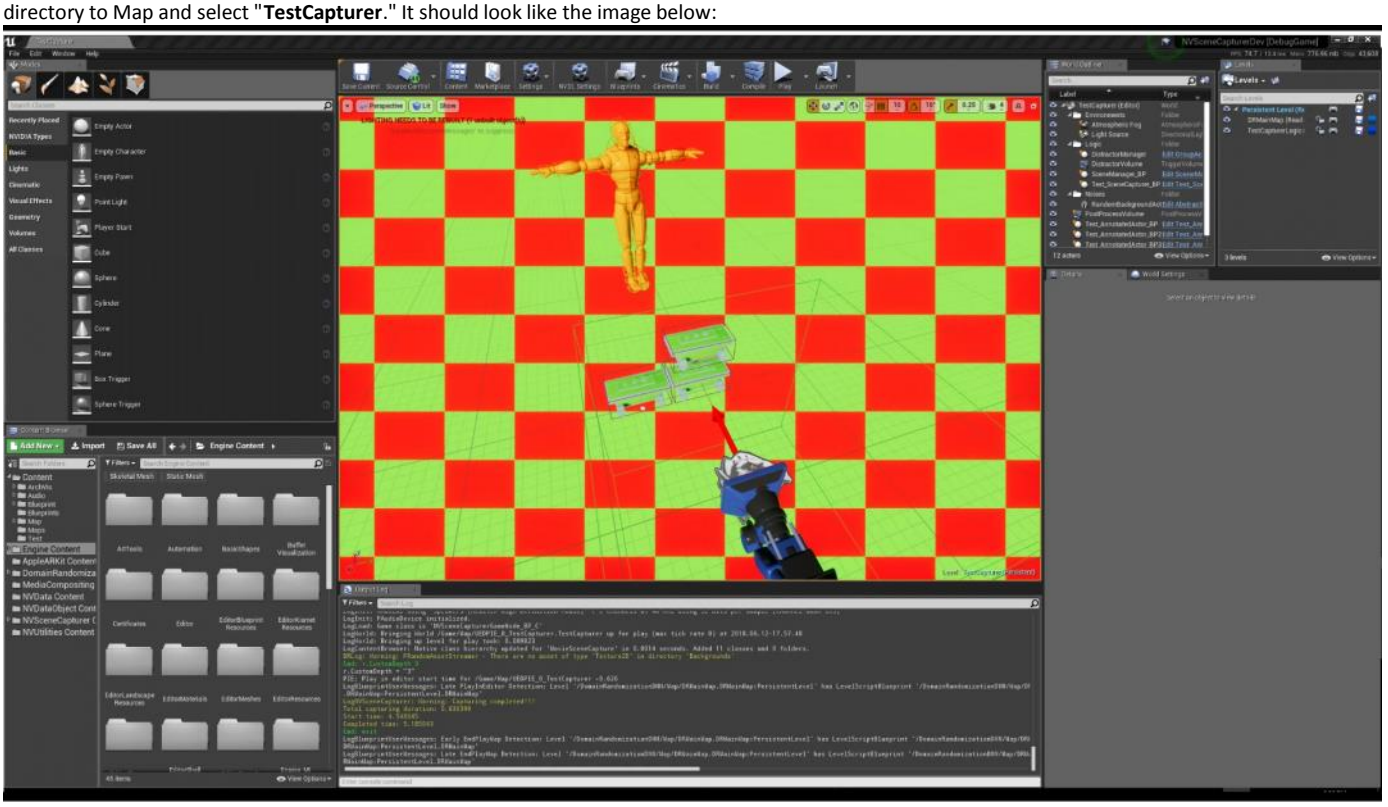
35. Go to the Viewport bar and look for the **PLAY** button icon and press **PLAY** to start. **Note that once you do this your cursor will disappear**. You can either exit by pressing **ESC** or display the mouse by pressing **SHIFT+F1** to regain cursor control. An in depth look at these features in Unreal can be found here.


Play
Launch
Modes
Selected Viewport
Mobile Preview ES2 (PIE)
New Editor Window (PIE)
VR Preview
Standalone Game
Simulate
Alt+S
Launching a game preview with a different mode will change your default 'Play' mode in the toolbar
Spawn player at...
Current Camera Location
Default Player Start
Multiplayer Options
Number of Players
1
Run Dedicated Server
Advanced Settings...
Play this level in the active level editor viewport

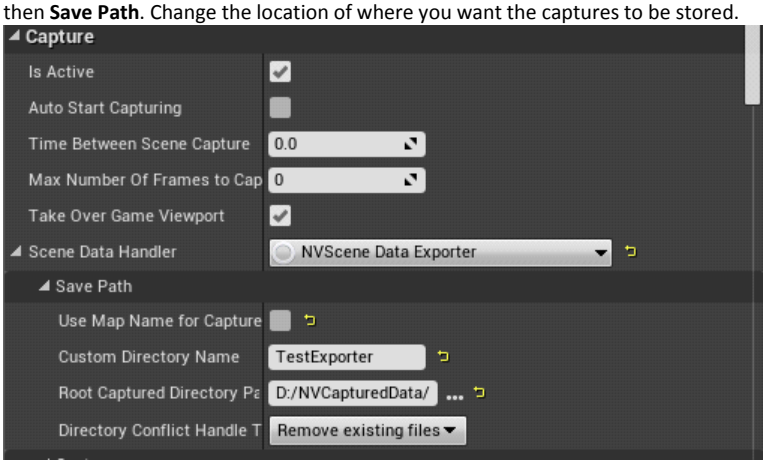
Export Captures from a Level with Domain Randomization

Thursday, June 21, 2018 11:12 AM

1. Open your level.
   a. For the purposes of this tutorial we will use our sample level called **TestCapturer** which is the default start up level for the NDDS project. Note, the level name is indicated on the upper left corner of the Editor window.
   b. If **TestCapturer** is not open go to the **FILE MENU > OPEN LEVEL**. Navigate the Content directory to Map and select "**TestCapturer**." It should look like the image below:



2. You must select a directory for where you want the files to be stored. To do this, go to **WORLD OUTLINER** and select **Test_SceneCapturer_BP**. Under **DETAILS**, click on **Scene Data Handler** and then **Save Path**. Change the location of where you want the captures to be stored.

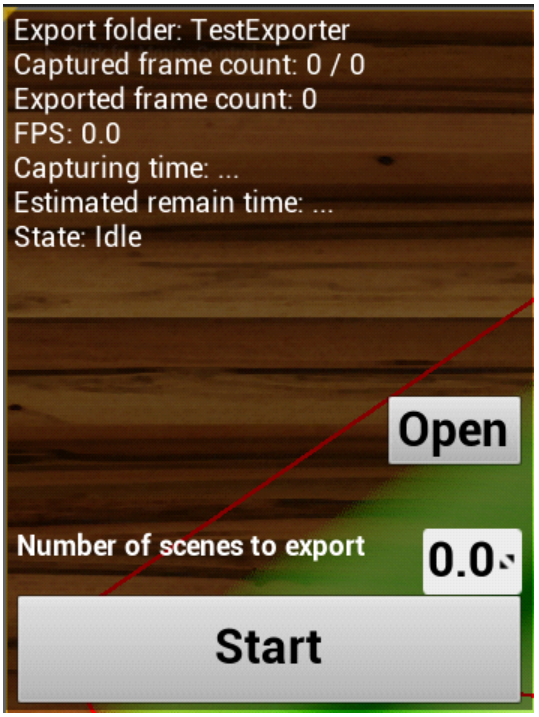


3. Before proceeding to the next step, note that when you press **PLAY,** the simulation will cause a rapid change to the display for the duration and is not intended for direct viewing. Please refer to the **HEALTH AND SAFTEY INFORMATION** section.

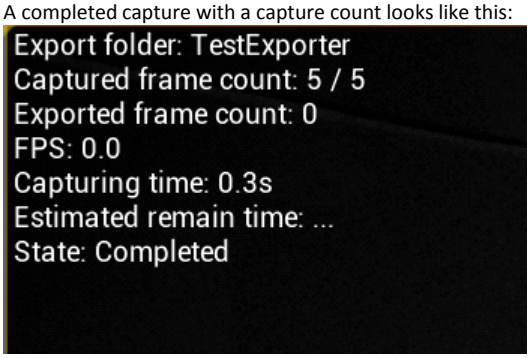
4. Press the "**Play**" button on the **Toolbar**. Ensure that you do not have "**Simulate**" option highlighted. To check, press the small arrow button beside "**Play**". The checkmark should be on the top selection. You can press Stop when you want to stop the capturer but it will interrupt an active capture. Only stop when you are done capturing.

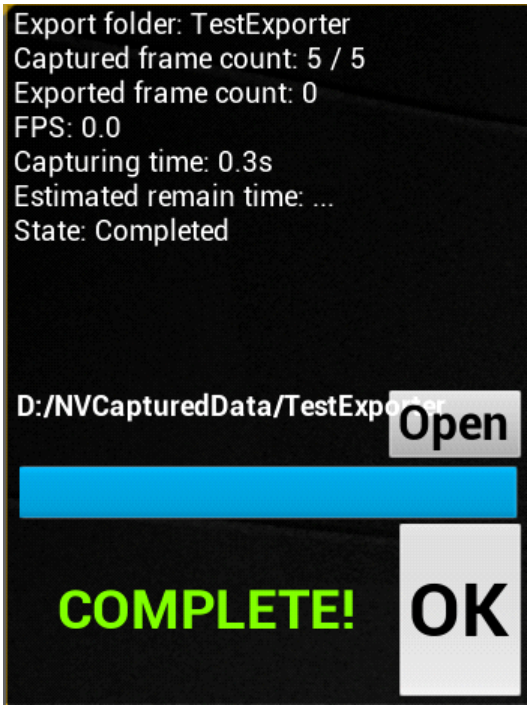
5. Notice your mouse cursor will not be usable in the simulation, to resolve this press **Shift + F1** to toggle it. Unreal will show this message on the upper left hand corner but it overlaps with the capture interface.

6. On the upper left hand corner of the viewport there are a few interactable buttons and fields:
   a. **BUTTON: Open** - A button that opens an explorer window to the location of your captured frames usually at the **<project folder>\<level name>** directory. This is valid if you completed a set of captures. The button will open a default empty location if you did not.
   b. **INPUT FIELD: Number of scenes to export** - An input field to set the number of frames to capture.
      - If you set a specific number, the capturer will run until the number of frames set is reached and will show a "**COMPLETE**" status message.
      - **If you set it to 0 a STOP and PAUSE button will appear. This is because you are capturing and will continue to capture until you stop it. Always use an integer greater than 1 for this setting.**
   c. **BUTTON: Start** - A button to start the capture process.
      - If you had set a specified amount a progress bar will update until it reaches the amount set.
      - **If you had set a specified amount, after it completes an "OK" button will appear. Press the OK button to start a new capture process.**



6. A completed capture with a capture count looks like this:



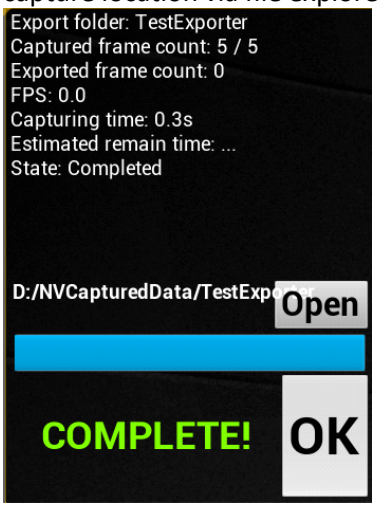

Export folder: TestExporter
Captured frame count: 5 / 5
Exported frame count: 0
FPS: 0.0
Capturing time: 0.3s
Estimated remain time: ...
State: Completed
D:/NVCapturedData/TestExp
Open
COMPLETE!
OK

Reviewing the Captured Data

Wednesday, June 27, 2018 9:41 AM

1. Obtain some capture data.
    a. For a quick guide to this, review the instruction page "**Export Captures from a Level**"

2. Either press the **Open** button on the interface from the Capture Interface or navigate to your capture location via file explorer.



3. When reviewing capture data, you should see the following files:
    a. **A color image named <frame#>.png**
    b. **A depth image named <frame#>.depth.png**
    c. **A class segmentation image named <frame#>.cs.png**
    d. **An instance segmentation image named <frame#>.is.png**
    e. **An object property data text file named <frame#>.json**

Feature Extraction Details

Thursday, June 28, 2018 4:27 PM

In this section we are going to give a deeper look to the different meta-data the tool can export.

Feature Extraction Details

For any *SceneCapturer* present in the scene, you can add as many features to extract as you need, although it will impact performance and the captured data will be bigger

[https://github.com/NVIDIA/Dataset_Synthesizer/blob/master/Videos/overview_scene_capturer.mp4]

Here are an overview of the different choices:

- Object Data: Meta data linked to any objects with a *NVCapturableActorTag* see below for more information
- True Color (GroundTruth): The RGB frame seen by the capturer

Depth

Absolute

These absolute depth feature extractors capture the scene's depth in absolute value.

1. *FE_Depth_micron_32bits*: capture scene depth in micron units, using 32 bits. Max distance is 4km
2. *FE_Depth_mm_16bits*: capture scene depth in mm units, using 16 bits. Max distance is 65.535m (65535mm)
3. *FE_Depth_cm_16bits*: capture scene depth in cm units, using 16 bits. Max distance is 655.35m (65535cm)
4. *FE_Depth_cm_8bits*: capture scene depth in cm units, using 8 bits. Max distance is 2.55m (255cm)

Quantized

These depth feature extractors capture the scene's depth and quantize it into range [0, 1]

1. *FE_DepthQuantized_8bits*: capture and quantized scene depth using 8 bits - it map [0, 1] to [0, 255]. To convert the value from the grayscale use the formula: (PixelValue / 255.0) * MaxDepthDistance
2. *FE_DepthQuantized_16bits*: capture and quantized scene depth using 16 bits - it map [0, 1] to [0, 65535]. This 16 bits quantized feature extractor is more accurate than the 8 bits but it's normally bigger. To convert the value from the grayscale use the formula: (PixelValue / 65535.0) * MaxDepthDistance

Segmentation

1. *FE_InstanceSegmentation*: capture object segmentation in 32 bits color mode - each object have its own id and color even if they are using the same 3d models.
2. *FE_ClassSegmentation*: capture the class segmentation of objects to grayscale pixel format - different objects which have same tags or using the same 3d models will share the same id.

Metadata

In this section we are looking at the meta data exported when a *NVCapturableActorTag* is added to an object.

[https://github.com/NVIDIA/Dataset_Synthesizer/blob/master/Videos/data_overview.mp4]

Each frame exported contains the camera position/rotation in the virtual environment, _camera_data. Each object present is stored in an array from which you can retrieve the class, visibility (0 means fully

visible, 1 is completely occluded), pose (location with quaternion in camera space), the cuboid centroid (virtual world and camera), bounding_box, cuboid in the virtual world (8 coordinates), and projected_cuboid (8 coordinates). The indicies of the cuboid are as follows (corner id = index) :

FrontTopRight = 0
FrontTopLeft = 1
FrontBottomLeft = 2
FrontBottomRight = 3
RearTopRight = 4
RearTopLeft = 5
RearBottomLeft = 6
RearBottomRight = 7

NOTE: in the 'bounding_box' field, the "top_left" and "bottom_right" coordinate format is (Y, X) instead of (X, Y) like other projected 2d points. *To make it more consistent, we will switch this format to (X, Y) in a future release.*

Randomization Components Details

Friday, March 22, 2019 3:41 PM

NDDS comes with additional actor components that provide randomization behaviors. For more information about actor components please refer to the official Unreal 4 documentation at the following location:
https://docs.unrealengine.com/en-US/Gameplay/HowTo/AddingComponents/Blueprints

Randomize Lights

The following components randomize lights attached to the same actor.

RandomLight

This component randomizes the light intensity and color for the generic light sources:

- IntensityRange - randomize intensity within this range
- ColorData - specify how the random light color is generated (see details in RandomMaterialParam_Color)

RandomLightComponent_SpotLight

On top of the generic light randomization, this adds specific settings for spot lights

- InnerConeAngleRange, OuterConeAngleRange - randomization settings for the inner and outer cone of the spot light

Randomize Mesh

RandomMesh

This component randomizes the assigned mesh of all the mesh components on the same actor

These settings are mutually exclusive:

- MeshDirectories - pick a random mesh from all meshes located inside these directories
- StaticMeshList - pick random a mesh from this list

NOTE: Right now we only support StaticMesh, not SkeletalMesh

Randomize Transforms

RandomMovement

This component randomizes the location of the actor

- RandomLocationData - (mutually exclusive with RandomLocationVolume) random location is selected around the actor's original location
  - X/Y/Z axes can be individually randomized
  - UseObjectAxesInsteadOfWorldAxes - if true, the location will be picked along the object's axes instead of the world axes
- RandomLocationVolume - (mutually exclusive with RandomLocationData) if this is set, random location is sampled within the volume
- ShouldTeleport - whether the actor will be teleported to the random location or move towards it (speed set using RandomSpeedRange)
- CheckCollision - whether to prevent collision when either teleport or move towards the random location (actor will stop as close as possible to the selected location before penetrating other objects)

RandomLookAt

This component changes the rotation of the owner actor to orient towards a random target in the list. The timing of which target switching occurs is determined by the RandomizationDurationInterval (and other random timing settings) set in the general random component settings.

- FocalTargetActors - the list of actors to randomly choose from to look at
- RotationSpeed - rotation speed that determines how fast the orientation changes

NOTE: Since there is a rotation speed, the actor only changes target after it is fully oriented to look at the previous target.

RandomRotation

This component changes the actor's rotation randomly

- RandomConeHalfAngle - choose a random orientation within the cone angle around the actor's +x axis (forward direction)
  - This option is useful for the light source
- Pitch/Roll/YawRange - pick each specific random value for pitch roll and yaw
- RelatedToOriginRotation - if true, the owner will be rotated around its original rotation, otherwise the random rotation will be around the world origin

RandomScale

This component randomizes the scale of the actor

- UniformScaleRange - scale all 3 axes with the same random value pick within this range
- X/Y/ZAxisRange, scale each axis individually

NOTE: Do not use this component on the object-of-interest if you want to capture its 3d pose; only use it to detect the object's segmentation or 2d bounding box. The reason is it will change the object's cuboid dimension and thus invalidate the captured cuboid in the frame where it's scaled. Just use the RandomMovement to move the object closer/further from the camera.

Randomize Material Parameters

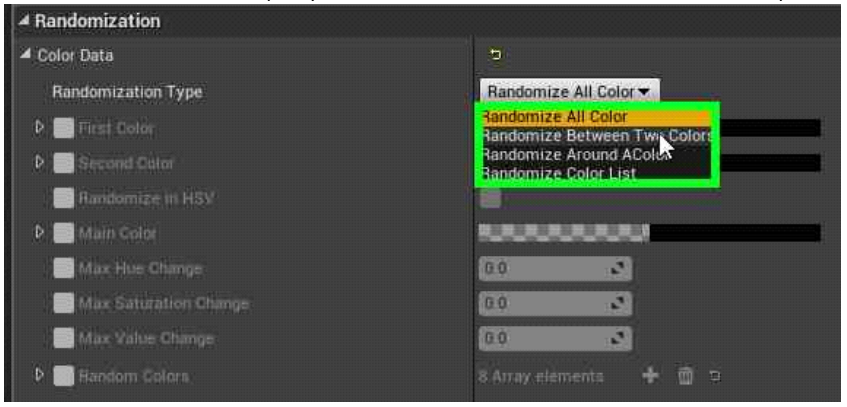
The following components all randomize certain material parameters:

RandomMaterialParam_Color

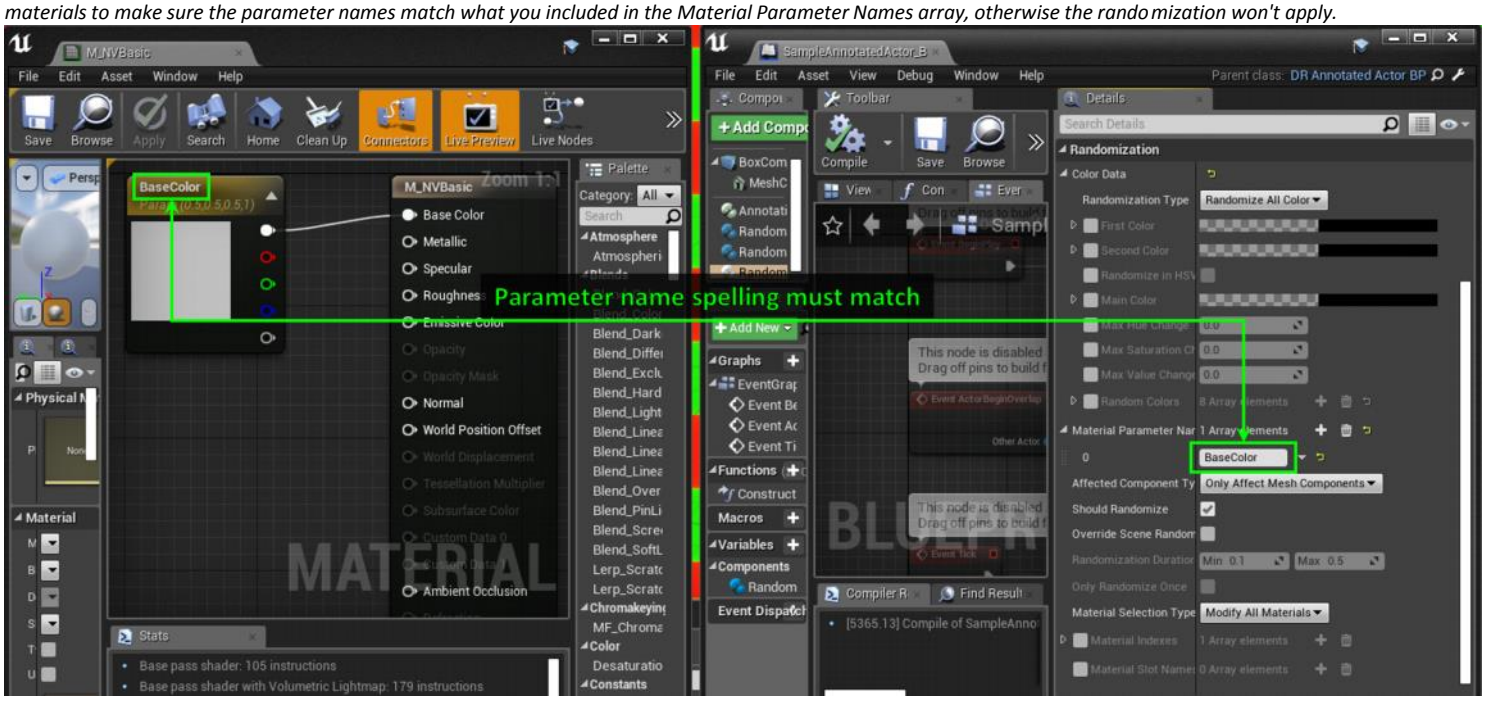
This component can randomize the color values used by parameters of the materials (known as "Vector" parameters within the material logic). *Note: Make sure the materials on the actor's mesh are using vector parameters to determine the relevant colors and take note of the parameter names.*

Choose the **RandomizationType** from the dropdown to determine how to generate the random color:

- All Color - uniformly select a color from entire color space
- Between Two Colors - uniformly select a color in between two colors (linearly interpolated)
- Around A Color - set a main color and generate a color within the deviation
  - Main Color - the color which the random color deviates from
  - Max Hue/Saturation/Value change - determines the deviation
- Color List - randomly pick a color in a custom list of colors
  - Random Colors - specify a list of custom colors to choose from, *Note: no interpolation among colors*



In the **Material Parameter Names** array, add the parameter names for each color (vector) parameter in the mesh's materials that should be randomized. *Note: Check the materials to make sure the parameter names match what you included in the Material Parameter Names array, otherwise the randomization won't apply.*



For more information about making Materials with Material Parameters, please reference the Unreal Engine documentation: https://docs.unrealengine.com/en-us/Engine/Rendering/Materials/HowTo/Making_Parameters

Relevant Sample Content: The sample content of NDDS includes a very simple material called "M_NVBasic", found in the '../NVSamples/Materials' folder . The M_NVBasic material has a "BaseColor" parameter that determines the color of the material. To get running quickly with the RandomMaterialParam_Color component, you can use this material (or a material instance based on it) for your actor's mesh instead of making your own material.

RandomMaterialParam_Scalar

This component randomizes scalar parameters (specified by the material parameter names setting) of the materials

- ValueRange - random scalar value selected within this range

RandomMaterialParam_Texture

This component randomizes texture parameters (specified by the material parameter names setting) of the materials
These settings are mutually exclusive:

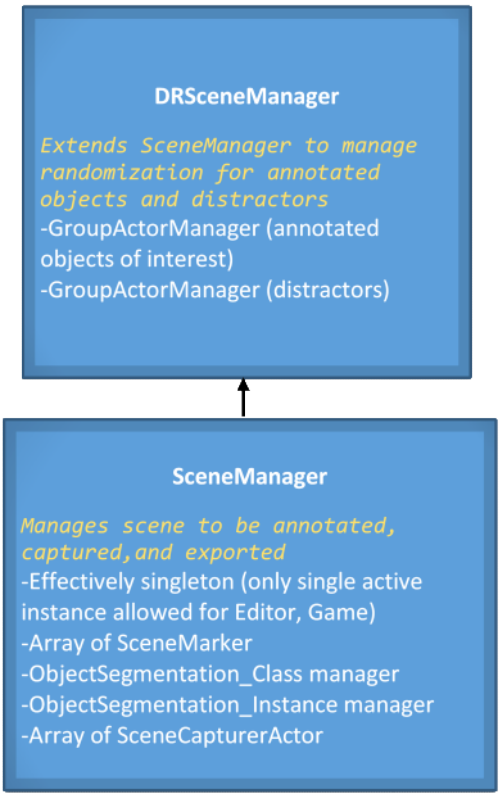
- TextureDirectories - textures are randomly selected from the ones located in these directories
- TextureList - textures are randomly selected from this list

Class Summary

Monday, June 25, 2018 10:46 AM

The following is a summary of the major classes in NDDS, focusing on the most important relationships and fields.
For more detail see: https://github.com/NVIDIA/Dataset_Synthesizer/tree/master/Documentation/ClassDetails
To view, download this folder and navigate to index.html

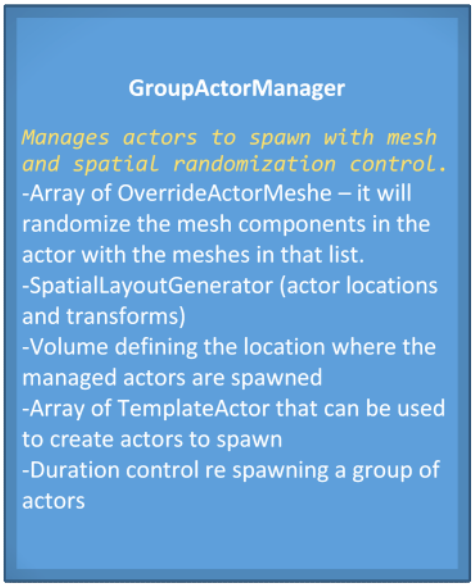
Scene Manager



Notes on usage:

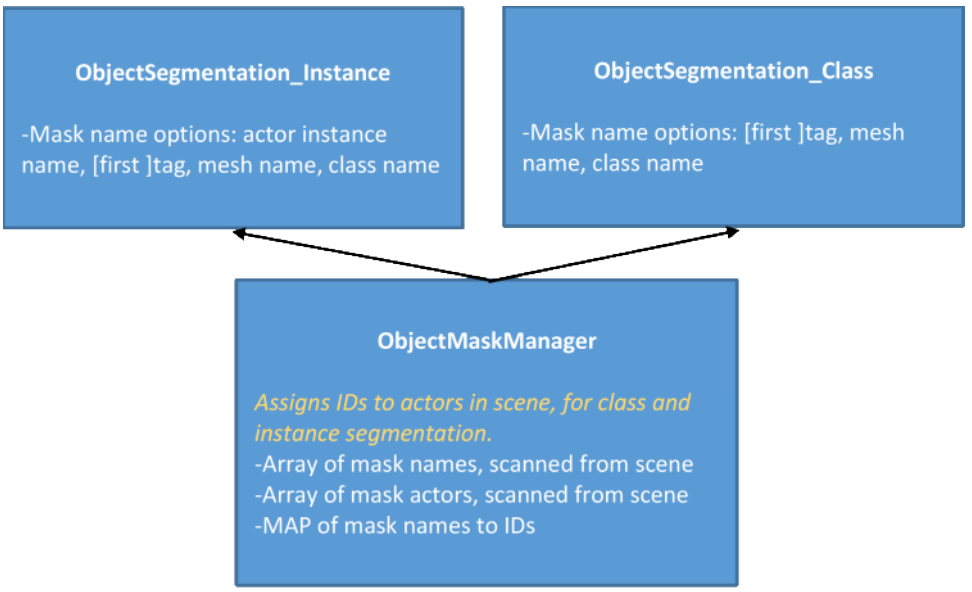
There is a 1:N relationship between *SceneMarker* and *SceneCapturer* -- for each marker, every active capturer is invoked.

GroupActorManager



ObjectMaskManager

Notes on usage:

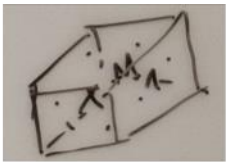


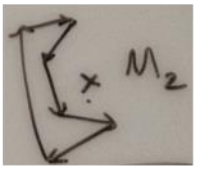
ObjectMaskManager

Sequentially vs Spread evenly actor IDs:

The reason for the "spread evenly" is for easy visualization. IDs are translated to color linearly (greyscale 8 bits for stencil mask, RGBA8 for vertex color mask). **When the number of objects is small, the mask images all look black and thus are hard to distinguish**. In this case "spread evenly" is useful to emphasize the colour difference for visibility. However, for all other scenarios, such as DL training, the colour doesn't matter and a sequentially allocating IDs may be easier to debug. It also helps in cases where on is manually assigning additional IDs -- one need only increase the max ID.

Scene Marker





SceneMarker_Volume

-Controls how capturer moves re anchor: Move randomly within a volume about anchor
-SceneMarkerComponent

SceneMarker_FocalPoint

-Controls how capturer moves re anchor: Move randomly in orbit around anchor
-SceneMarkerComponent

SceneMarker_Path

-Controls how capturer moves re anchor: Move between waypoints from the anchor
-SceneMarkerComponent

SceneMarkerInterface

-Anchor for point of interest in map

SceneMarkerComponent

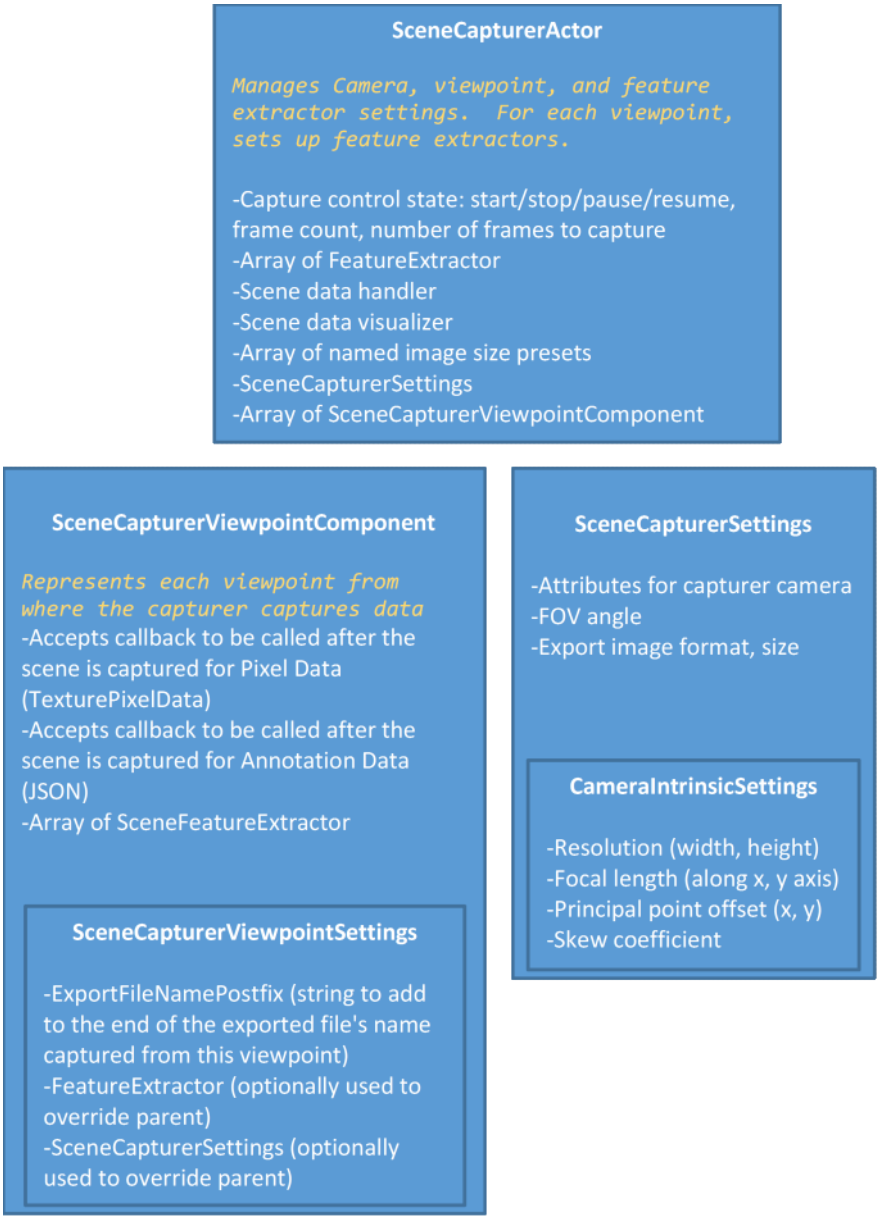
-Attributes for anchor point
-Display name
-Description
-IsActive
-Array of observers (SceneCapturerActor)

Notes on usage:

SceneMarkerComponent

These can be manually deactivated via toggling the *IsActive* attribute to allow the user to control which subset of scene markers gets captured and exported.

SceneCapturerActor

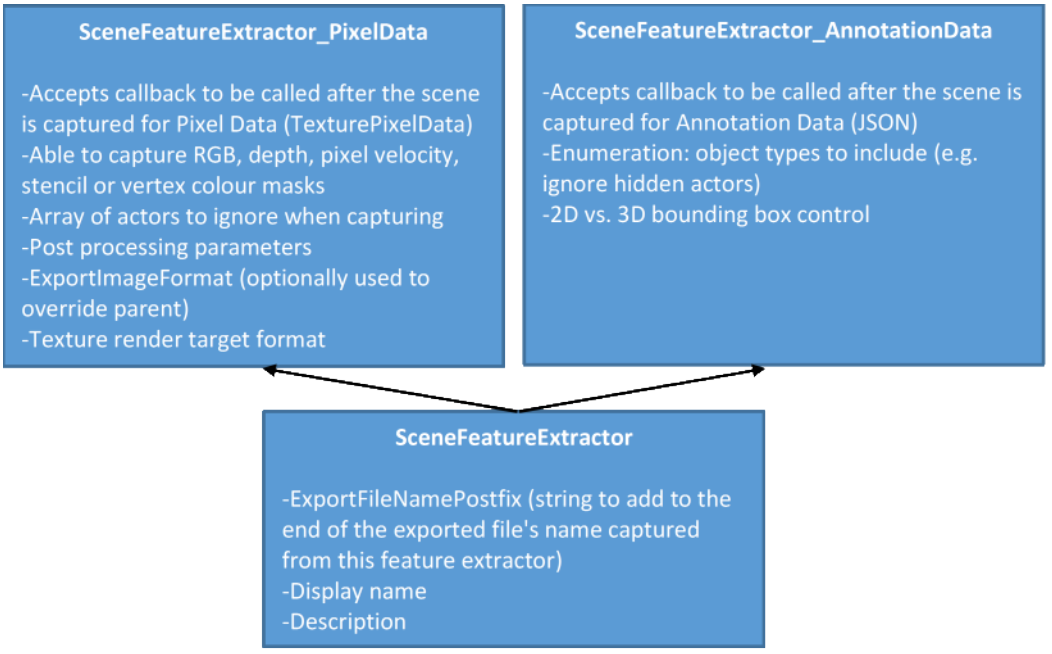


Notes on usage:

SceneCapturerActor

-Capture state can be controlled manually via Start/Pause/Resume/Stop, or automatically for batched offline use via setting *MaxNumberOfFramesToCapture*.

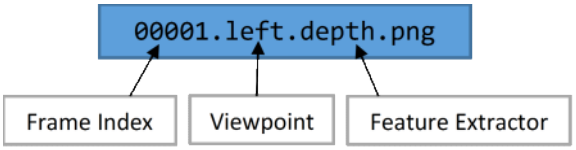
FeatureExtractors



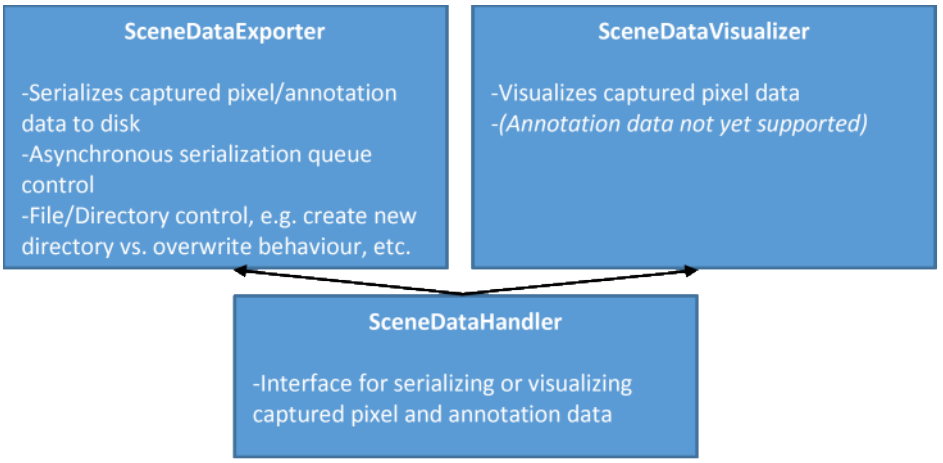
Notes on usage:

ExportFileNamePostFix

Considering the hierarchy of actor -> viewpoint -> feature extractor, *ExportFileNamePosFix* values are accumulated. For example, a combined resultant filename could be:



DataHandler and DataVisualizer



Command line arguments

Thursday, August 23, 2018 3:02 PM

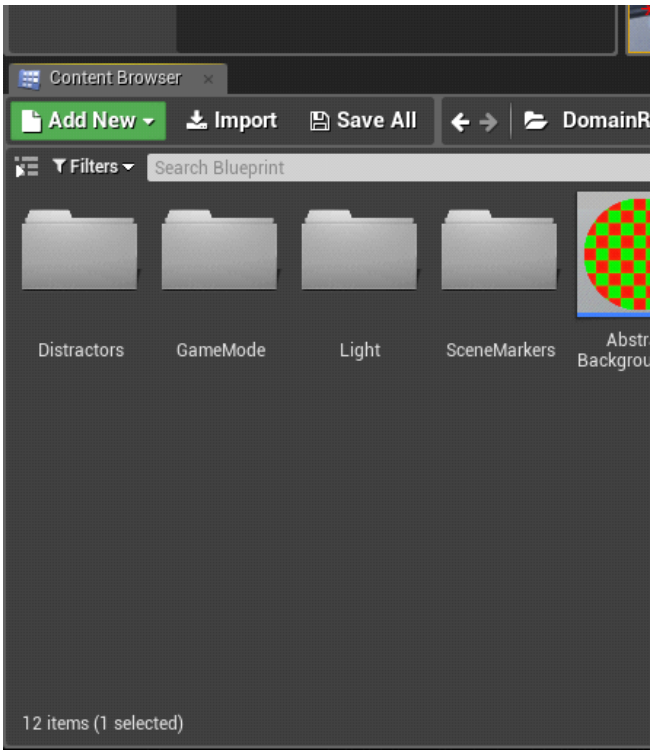
NOTES: THESE COMMAND LINE ARGUMENTS ARE NOT FINALIZED AND MAY BE CHANGED

-OutputPath="DirectoryWhereYouWantTheExportedFilesAre"
-NumberOfFrame="NumberOfFrameToCapture"

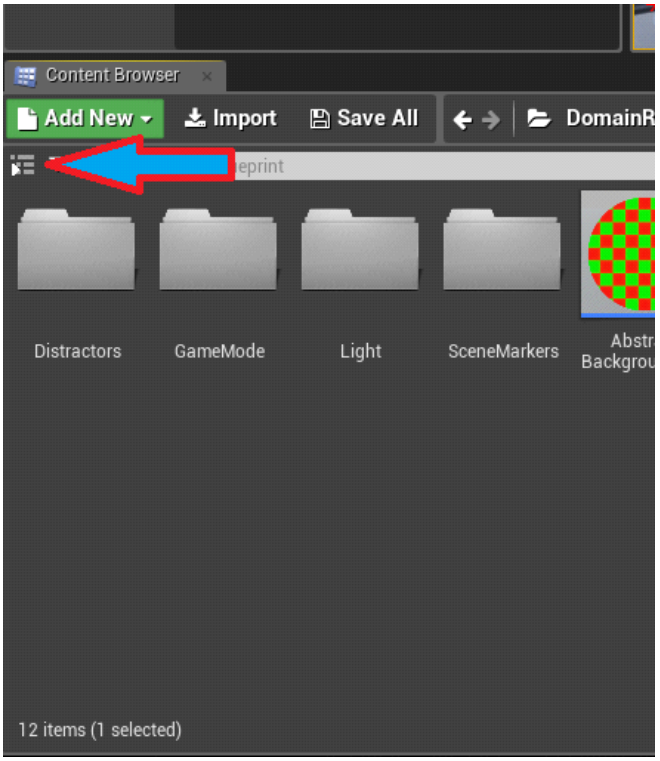
Addendum for Tutorial_1.mp4

Tuesday, December 18, 2018 2:46 PM

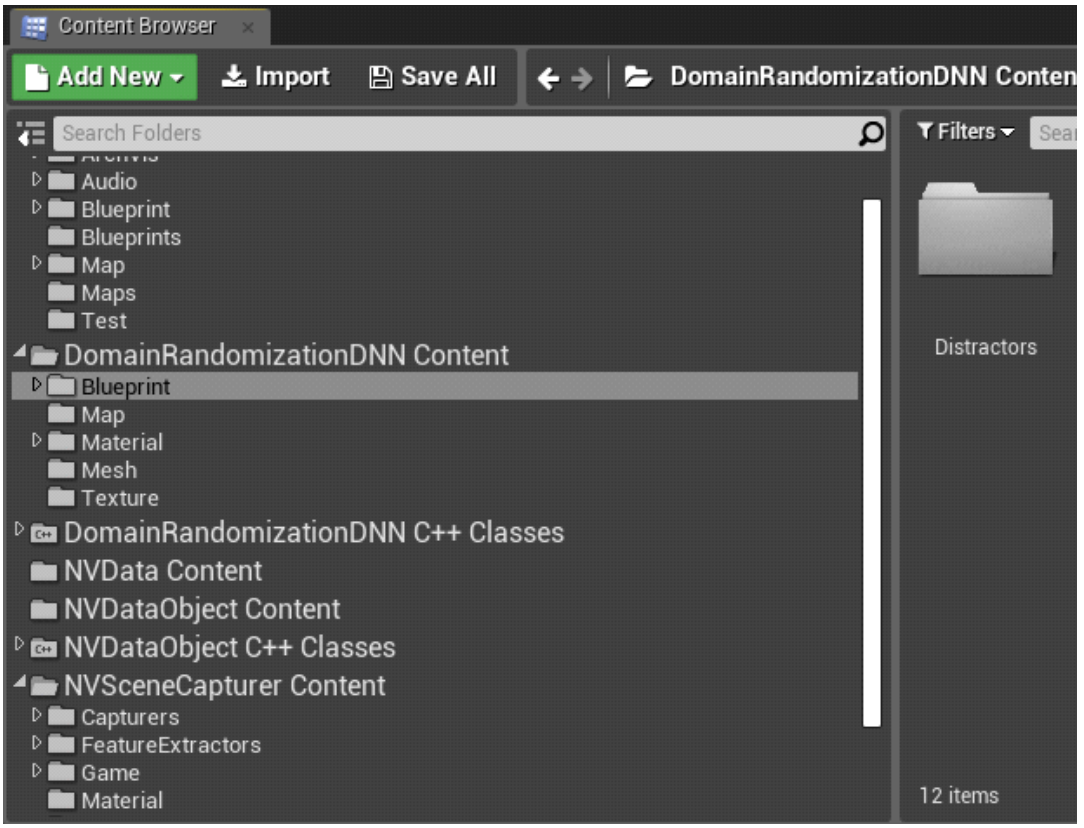
1. Some users has noted that the content browser is not visible or looks empty when following the video. Upon a new installation of the Unreal 4.22.2, you may notice that your screen looks like the following:



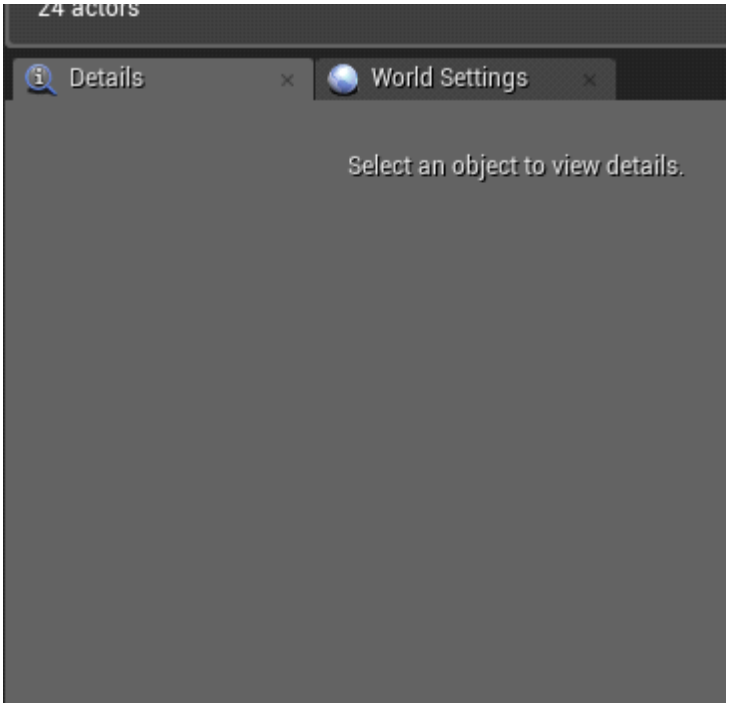
To open the source panel, all you need to do is click on the following button shown in the screenshot



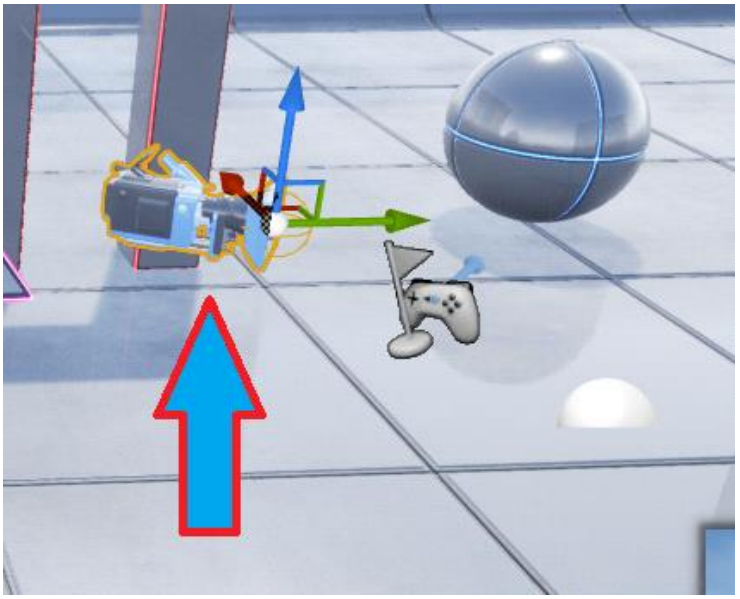
Once that is opened, your Content Browser should like the following, which is the same as on the video

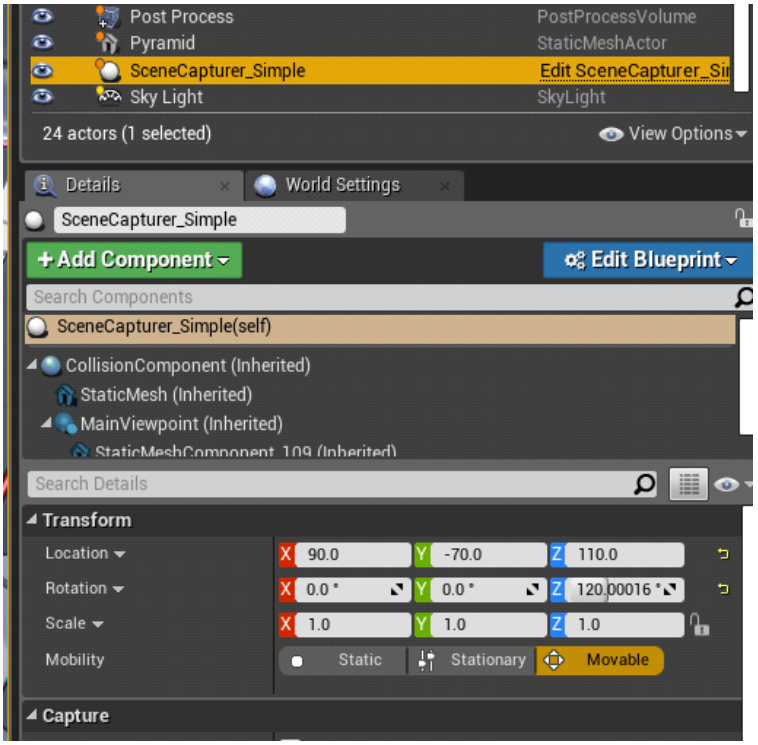


2. If you notice that the details panel is empty as shown below



This is probably because there is not an object selected in the scene. Simply highlight the object you want to configure and the Details panel will show the proper configuration options




Post Process
PostProcessVolume
Pyramid
StaticMeshActor
SceneCapturer_Simple
Edit SceneCapturer_Si
Sky Light
SkyLight
24 actors (1 selected)
View Options
Details
World Settings
SceneCapturer_Simple
+ Add Component
Edit Blueprint
Search Components
SceneCapturer_Simple(self)
CollisionComponent (Inherited)
StaticMesh (Inherited)
MainViewpoint (Inherited)
Search Details
Transform
Location
X 90.0
Y -70.0
Z 110.0
Rotation
X 0.0 °
Y 0.0 °
Z 120.00016 °
Scale
X 1.0
Y 1.0
Z 1.0
Mobility
Static
Stationary
Movable
Capture