TNM091 — Media Production for Immersive Environments
4. Real-time rendering

September 28, 2021

1 Introduction

In this session you will learn how to build and design a real-time application that operates on a computer cluster in the Dome in Norrköping. You will get familiarized with Unreal Engine 4 and the nDisplay cluster plugin to achieve this goal.

1.1 Layout

In this lab you will have completed:

1. understand the basics of Unreal Engine,
2. understand the setup of the dome using nDisplay,
3. understand how to program both in code and in Blueprints,
4. placing 3D objects in the scene and alter their behavior,
5. add various forms of interactions to the 3D objects

2 Prerequisites

Unreal Engine 4 is a big commercial game engine, with all sorts of capabilities to create stunning 3D visual for multiple purposes. To get started, we assume you have read and understood the terminology in Unreal. `https://docs.unrealengine.com/en-US/GettingStarted/Terminology/index.html`

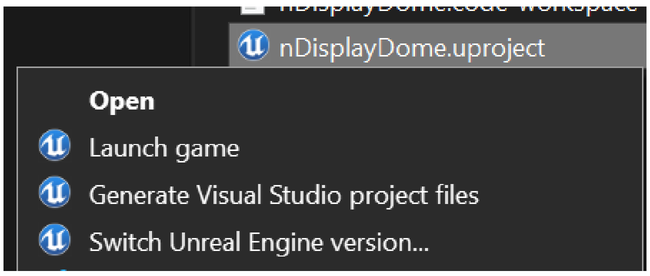
3 Get the project files

Start off by getting the code, by downloading the unreal project files for from the TNM091 webpage. You may need to work with them locally during the lab for it to work, but **do not forget to copy over the result to your personal drive**.

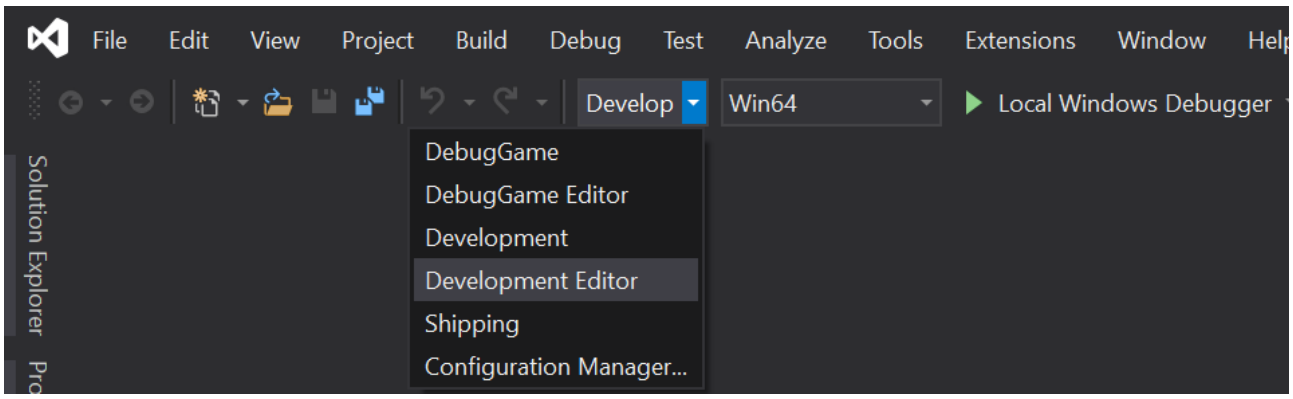
The Unreal Engine project we will be using as a starter template is located in the nDisplayDome folder in the zip file.

For a cluster setup, we will be using a plugin for Unreal Engine. In the nDisplayDome-folder you will find the nDisplayDome.uproject file. Since this is a C++ project of UE4, we will need to first generate the Visual Studio project. Do this by right clicking the nDisplayDome.uproject file and select

“Generate Visual Studio project files”


nDisplayDome.uproject
Open
Launch game
Generate Visual Studio project files
Switch Unreal Engine version...

This will generate a visual studio solution(.sln)-file which can be used to build and run the editor. In order to run the editor, make sure either the “Development Editor” or the “DebugGame Editor” build target is set in visual studio.



Building (Ctrl+Shift+B) this target ensures that the editor is setup correctly. You can also start (Ctrl+F5 or F5 for debugging) the editor through visual studio.

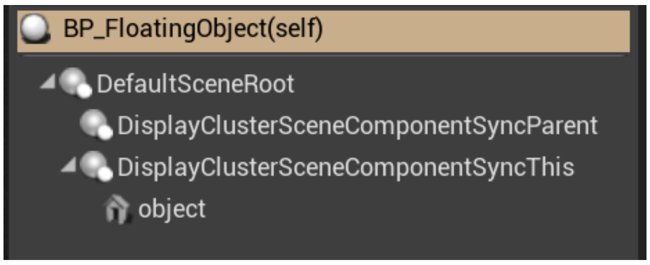
4 Testing your cluster application locally with nDisplay

In order to get the unreal project to work in the dome, we will be using the UE4 nDisplay plugin provided by Epic Games. Some reference material regarding nDisplay-specific setup can be found at:

https://docs.unrealengine.com/en-US/Engine/Rendering/nDisplay/index.html

nDisplay allows several computers linked together to run the same UE4 application in a synchronized manner. One instance of the application is started per cluster node and as such some consideration must be put into the setup of the application.

In order to get an Actor (which is what a scene object is referred to in UE) to deterministically replicate across a network of cluster nodes, we need to insert some nDisplay-specific components into it. An example could be found in the setup of the FloatingObject blueprint class.


BP_FloatingObject(self)
DefaultSceneRoot
DisplayClusterSceneComponentSyncParent
DisplayClusterSceneComponentSyncThis
object

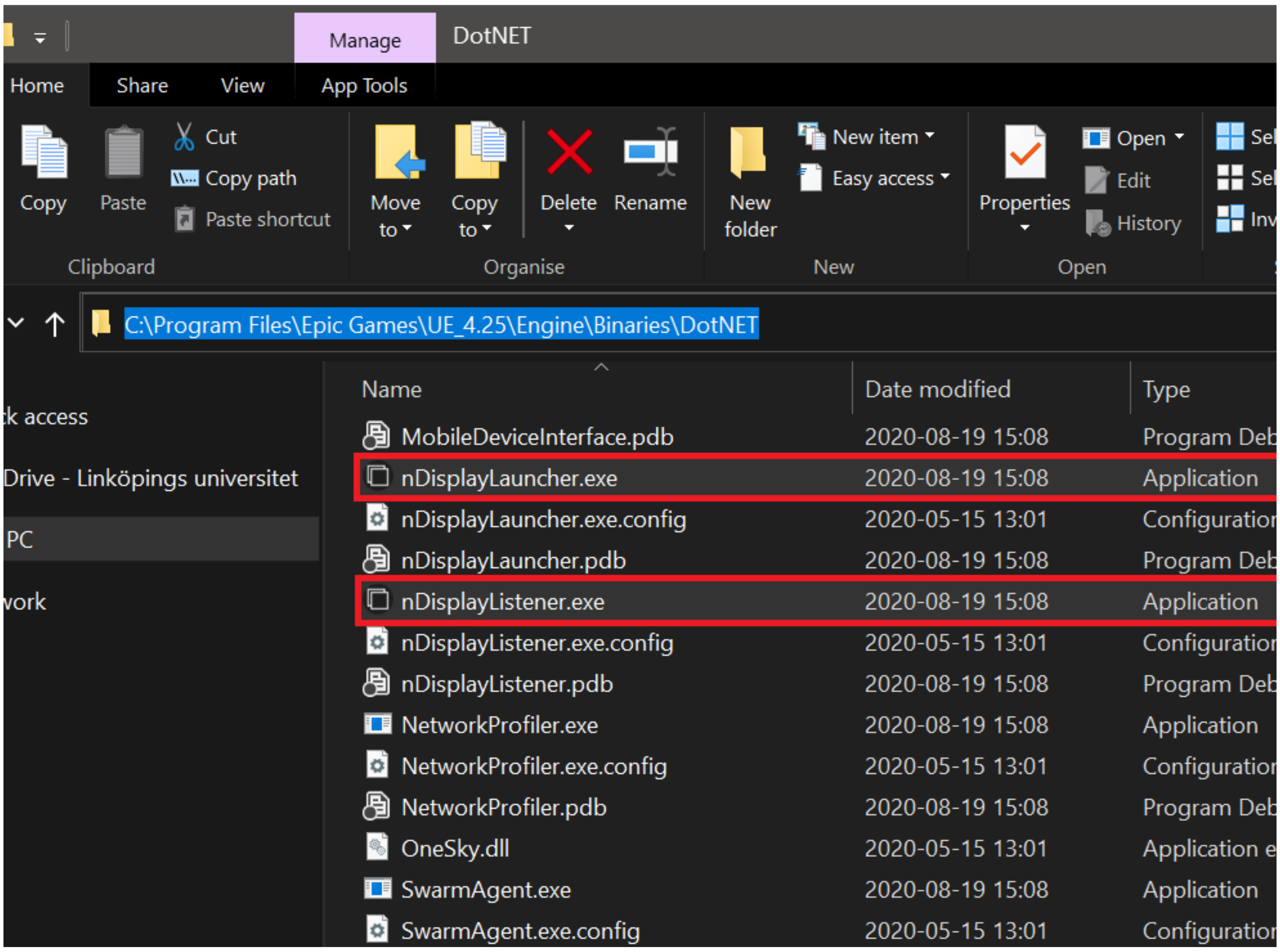
Here the DisplayClusterSceneComponentSyncParent/SyncThis components have been added in order to ensure that synchronization happens for this object. These components are more described on the documentation website but ensure that replication happens for nested components within this Actor.

4.1 nDisplayLauncher/nDisplayListener

In order to test-run a cluster application you can use the Unreal Engine-provided nDisplayLauncher tool. On a Windows PC the path to these programs is:

```
C:\Program Files\Epic Games\UE_4.25\Engine\Binaries\DotNET
```

Here you will find the nDisplayListener.exe and nDisplayLauncher.exe programs, which usually is run in a cluster, but can be used for a single PC as well.



C:\Program Files\Epic Games\UE_4.25\Engine\Binaries\DotNET

Name	Date modified	Type
MobileDeviceInterface.pdb	2020-08-19 15:08	Program Deb
nDisplayLauncher.exe	2020-08-19 15:08	Application
nDisplayLauncher.exe.config	2020-05-15 13:01	Configuratio
nDisplayLauncher.pdb	2020-08-19 15:08	Program Deb
nDisplayListener.exe	2020-08-19 15:08	Application
nDisplayListener.exe.config	2020-05-15 13:01	Configuratio
nDisplayListener.pdb	2020-08-19 15:08	Program Deb
NetworkProfiler.exe	2020-08-19 15:08	Application
NetworkProfiler.exe.config	2020-05-15 13:01	Configuratio
NetworkProfiler.pdb	2020-08-19 15:08	Program Deb
OneSky.dll	2020-05-15 13:01	Application e
SwarmAgent.exe	2020-08-19 15:08	Application
SwarmAgent.exe.config	2020-05-15 13:01	Configuratio

The nDisplayListener usually acts as a per-node listener, which listens for commands sent by the nDisplay-Launcher. It is necessary to use this in order for the nDisplayLauncher to work, as the listener is responsible for starting the application in the cluster.

The nDisplayLauncher, on the other hand, controls which application is started on the cluster, and which configuration file to use. The config file is an Unreal Engine-specific file which describes the physical projector planes, ip adresses and names of the cluster nodes. This file decides which computers the launcher will send commands to.

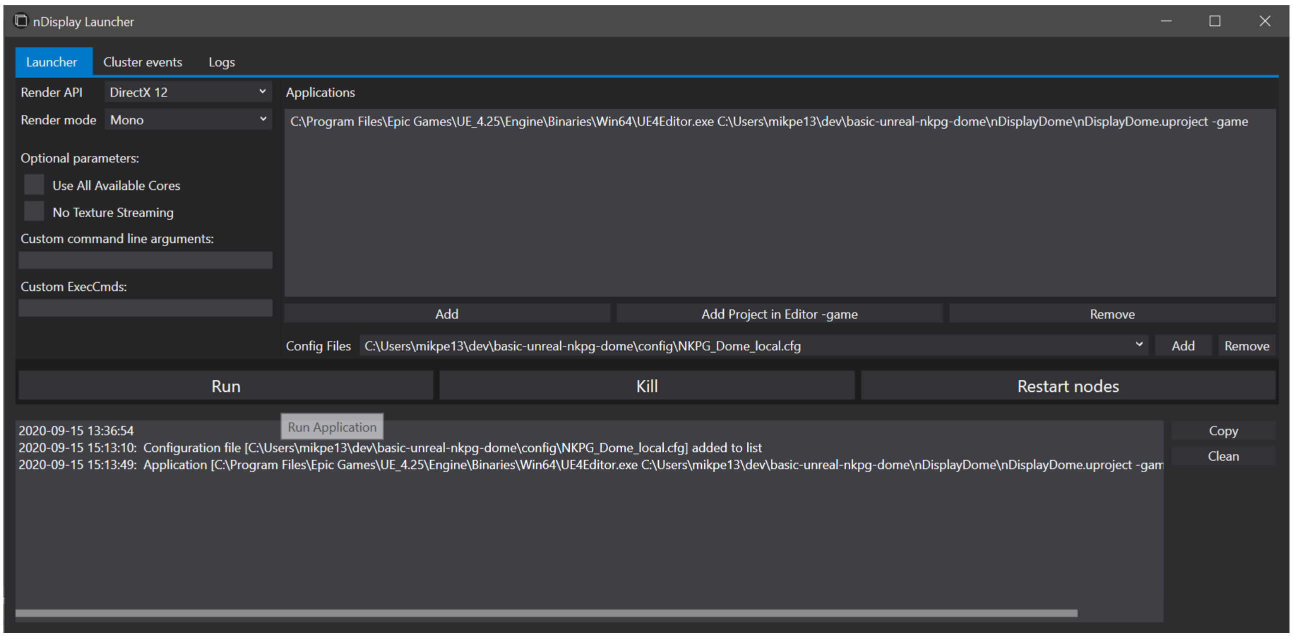
For this lab, it is recommended to use the provided `Content/DomeConfig/NKPG_Dome_local.cfg` config file.

In order to test your application on a locally simulated cluster, click "Add Project in Editor -game" and first select your UE4 editor executable located at

```
C:\Program Files\Epic Games\UE_4.25\Engine\Binaries\Win64\UE4Editor.exe
```

After the editor has been selected, select your nDisplayDome.uproject file.

Your nDisplay launcher should look similar to this:



Render API: DirectX 12
Render mode: Mono

Optional parameters:
Use All Available Cores
No Texture Streaming
Custom command line arguments:
Custom ExecCmds:

Applications
C:\Program Files\Epic Games\UE_4.25\Engine\Binaries\Win64\UE4Editor.exe C:\Users\mikpe13\dev\basic-unreal-nkpg-dome\nDisplayDome\nDisplayDome.uproject -game

Config Files: C:\Users\mikpe13\dev\basic-unreal-nkpg-dome\config\NKPG_Dome_local.cfg

```
2020-09-15 13:36:54
2020-09-15 15:13:10:  Configuration file [C:\Users\mikpe13\dev\basic-unreal-nkpg-dome\config\NKPG_Dome_local.cfg] added to list
2020-09-15 15:13:49:  Application [C:\Program Files\Epic Games\UE_4.25\Engine\Binaries\Win64\UE4Editor.exe C:\Users\mikpe13\dev\basic-unreal-nkpg-dome\nDisplayDome\nDisplayDome.uproject -gam
```

For the single-PC setup we will be using DirectX12 and Mono render mode.

Once this has been setup and the nDisplayListener is running, you can try running with the “Run”-button and later kill the cluster with the “Kill”-button.

5 Blueprints and Player Controller

One crucial concept in this lab are *Blueprints*. The Blueprints Visual Scripting system in Unreal Engine is a complete gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor. As with many common scripting languages, it is used to define object-oriented (OO) classes or objects in the engine. `https://docs.unrealengine.com/en-US/Engine/Blueprints/GettingStarted/index.html`

The most common Blueprint types you will be working with are Level Blueprints and Blueprint Classes. Each level has its own Level Blueprint, and this can reference and manipulate Actors within the level. The Level Blueprint can also interact with Blueprint Classes placed in the level, such as reading/setting any variables or triggering custom events they might contain. Blueprint Classes are then ideal for making interactive assets on Actors themselves.

Another concept in Unreal that is critical for this lab, is PlayerController. A PlayerController is the interface between the Pawn (a controlled actor) and the human player controlling it. The PlayerController essentially represents the human player’s will. `https://docs.unrealengine.com/en-US/Gameplay/Framework/Controller/PlayerController/index.html`

In the Content Browser, in `Content - Blueprints` you can find two blueprint classes, named `DomeGameMode` and `DomePlayerController`. If you double-click on `DomeGameMode` you will see that the parent class of this blueprint is `Game Mode Base`, which indicates that this is our game mode settings. And in the section regarding PlayerController, our `DomePlayerController` has been marked as the one used currently.