Directorate Subsystem

Guide and Documentation

Hephaistos Engineering Systems

1.0

Summary

1 Introduction

This middleware is designed to simplify the use of the game instance by breaking down the code into objects, the Machines, and managing the transitions between them and loading screens automatically.
It presents the following features:

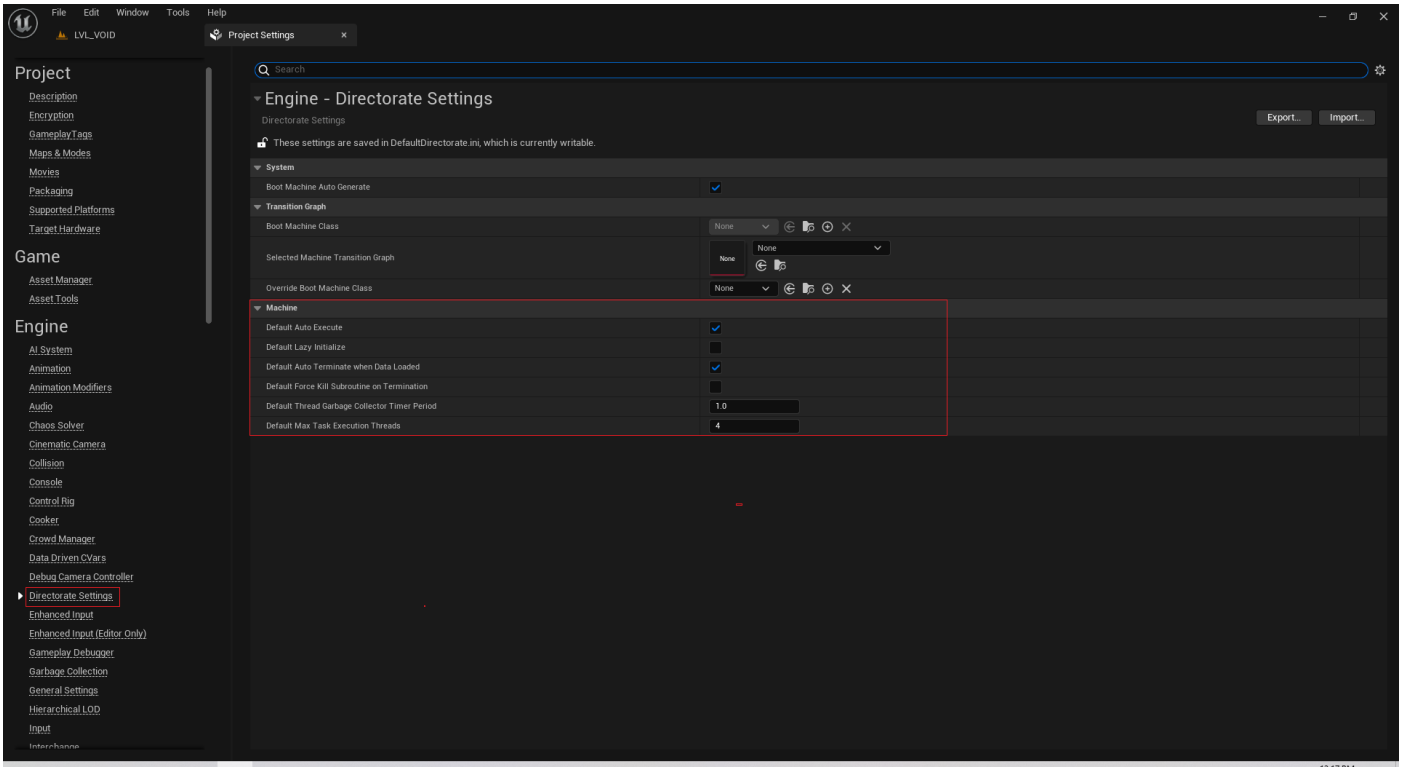
- Machines, which are fully programmable in Blueprint, can execute synchronous or asynchronous code.
- Transition Graphs, visual structures in which machines can be connected to determine their transitions and Transition Behaviors, special machines where it is easy to insert a loading screen and load the required data.

2 Getting Started

Here are described step-by-step the same operations performed for the sample project.
Check section **Directorate Objects** for a complete description of the various assets.

1. **Design the Graph**
   Define the number of machines, what they do, how and when they are connected. Investing time in a good starting design will spare much time in testing and debugging phases.

2. **Set custom defaults for new machines**
   Go to Project Settings / Engine / Directorate Settings.

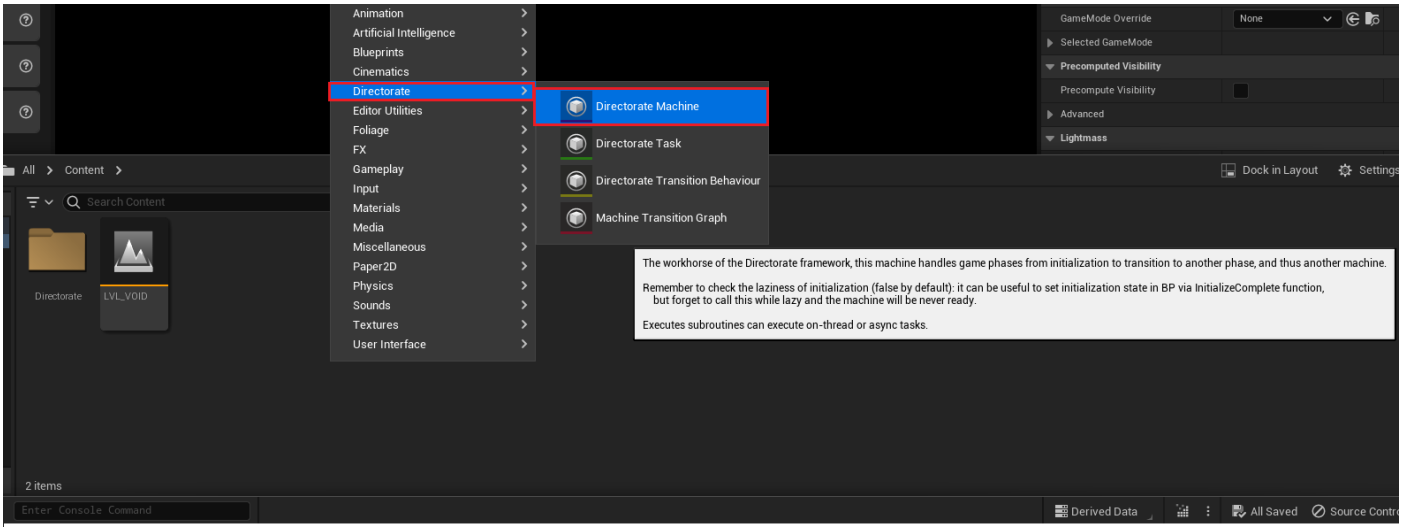


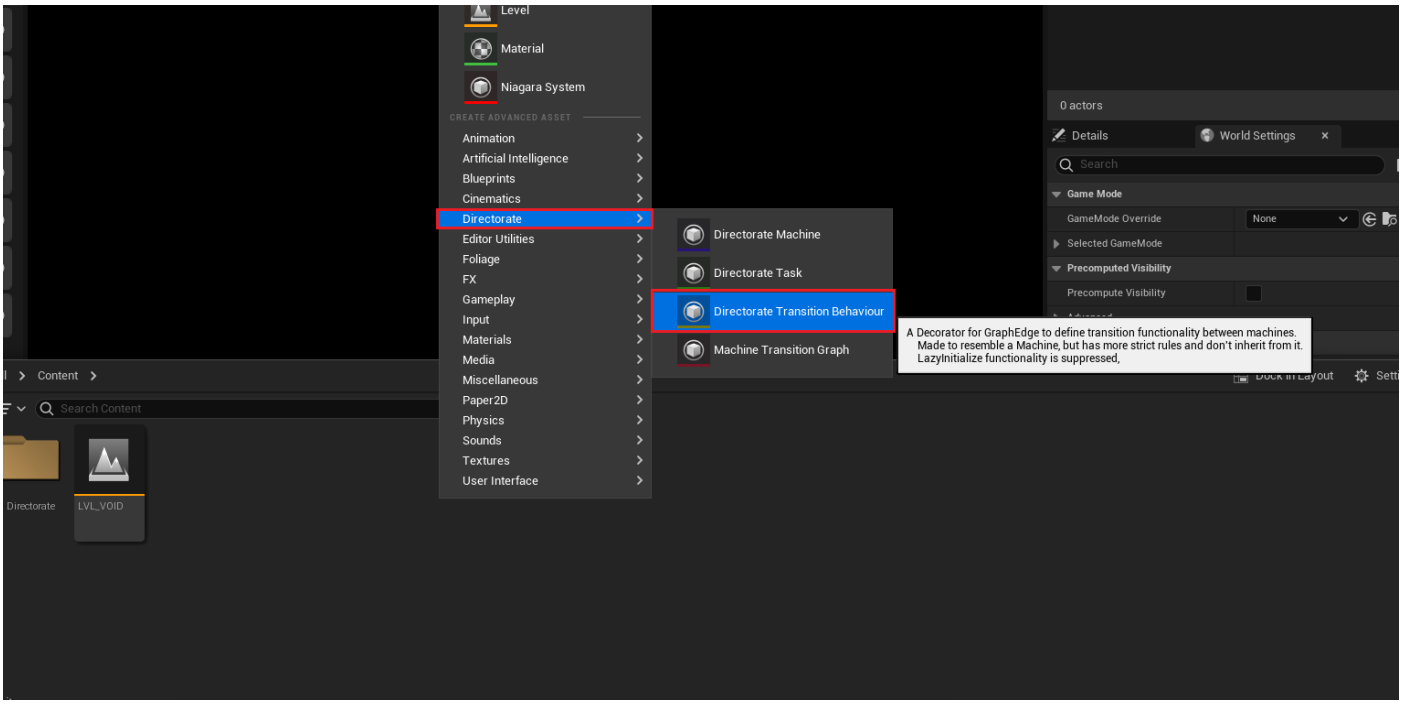
   Customize the settings under System and Machine categories for now, newly created machines will use those default values.

3. **Create new Machines**
   Go to your Content tab/ right click for contextual menu / Directorate / Directorate Machine.
   Rename new asset. Repeat this step until a desired number of machines are generated. Wait for step XX. for the machine programming, as the transition proxies have no pins right now.

4. **Create new Transition Behaviours**
   Go to your Content tab/ right click for contextual menu / Directorate / Directorate Transition Behaviour.

   Rename new asset and program your new behaviour. Repeat this step until a desired number of behaviours are generated.





3 Directorate Objects

3.1 Directorate Subsystem

Extension of Game Instance and the hub of the Directorate workflow. It holds the current Machine and all the functions to control it.

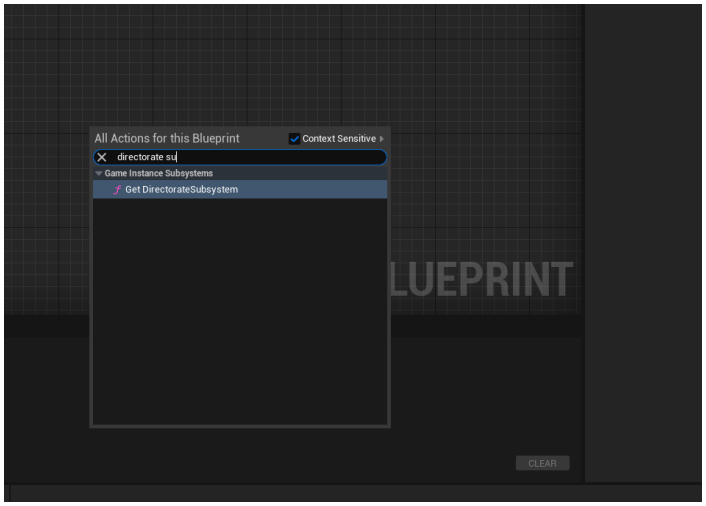
API

```
void OnSpawn()
```

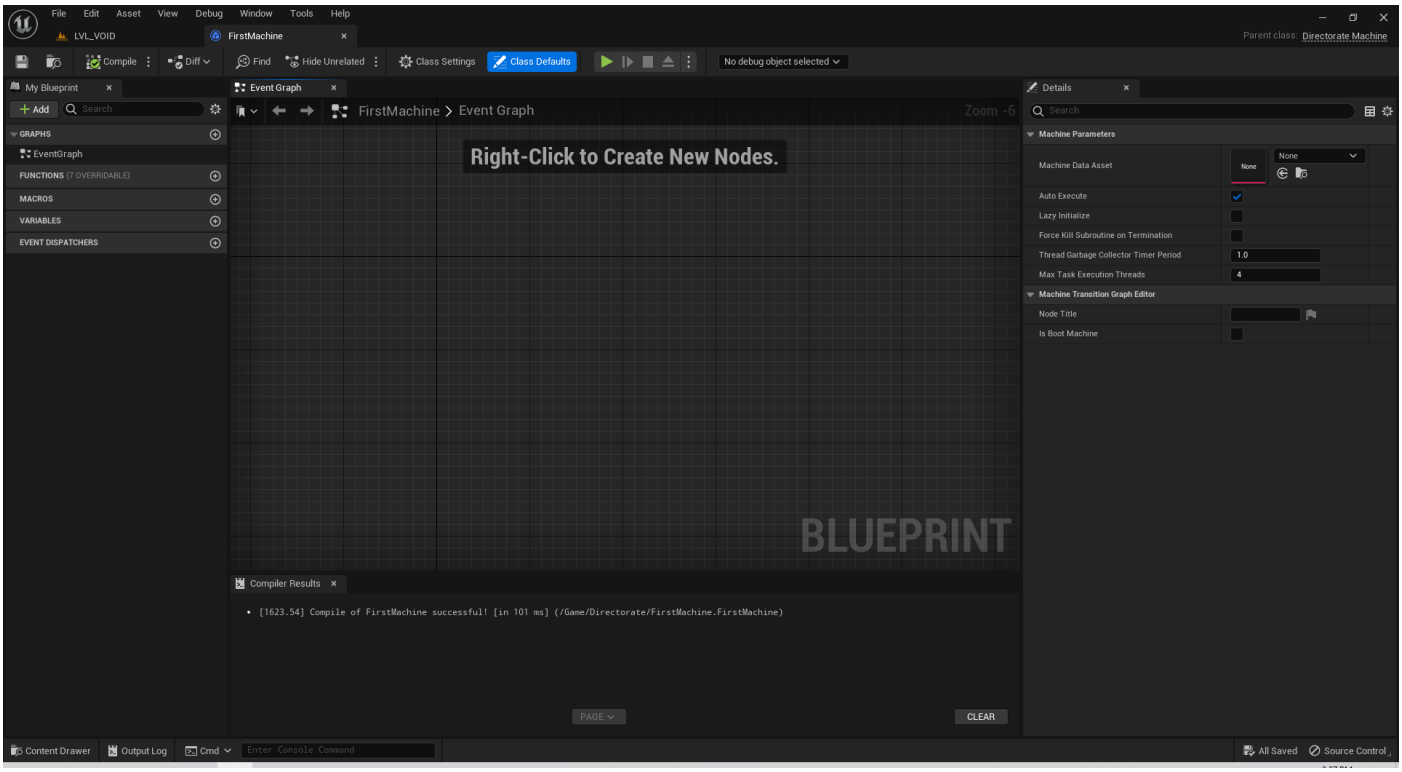
3.2 Directorate Machine

The workhorse of the Directorate framework, the Machine handles custom blueprint and native code. This asset has the structure of a common blueprint, so events, functions, macros, variables are allowed.

Lazy Initialization When a Machine has its Class Default LazyInitialize setting set to true, Initialize will not be called and Machine Status is stalled to **Uninitialized**. Can be useful if some operation are needed before the Machine can be executed You can see in Class Default 2 specific booleans that During its lifecycle, the Machine will call events accordingly to its current status, customizable via function override:



As any Game Instance Subsystem, can be called with ease on every blueprint ambients.




```
void OnSpawn()
```

Called when the Machine is created, before its Initialization. Machine Status is **Uninitialized**.

```
void OnInitialize()
```

The Machine is initialized and ready to accept an Execution() call. Machine Status is **Ready**.

```
void OnInitialize()
```

The Machine is initialized and ready to accept an Execution() call. Machine Status is **Ready**.

API

! *to be done, useful for advanced users with intention of virtual overrides.*

Can be manually Terminated, but to fully exploit Directorate workflow, * it's best to use transition proxies for a transition call to a new machine (will automatically call Terminate and StopExecution) and the subsystem Shutdown for the Quit Game And Last Machine * Termination. * * Remember to check the laziness of

initialization (false by default). It can be useful to manually set initialization state in BP via InitializeComplete function, * but forget to call this function on a lazy machine and will never be ready. * * Async tasks on Execute.

Directorate Machine Data Asset

Transition Behaviour

Task

Machine Transition Graph

Machine Transition Proxy

4 Troubleshooting

- All nodes in the Transition Graph must have at least a path for all other nodes. Although this is an intended behaviour, saving while failing this condition will cause a deadlock on Unreal Editor.