

## INSTALLATION

**CREATE AN UNREAL ENGINE 4.26.0 GAME BLUEPRINTS ONLY NO C++**

**INSTALLATION**

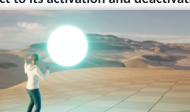
Advanced Simple Go to the next step

**Install Visual Studio**

- Search for Visual Studio Community [Download](#) Open Install.
- Select as components:
  - NET Framework Development
  - Universal Windows Platform Development
  - Desktop Development with C++
  - Game Development with C++
    - Make sure to manually select "Unreal Engine Installer" on the details panel here
  - Linux and embedded development with C++
- Install and do all updates that might be asked for
- Open Epic Games launcher
- Go to Unreal Engine 4.26.0 and Install latest Unreal Engine
- Get Epic from Marketplace and create Unreal Project from your library
- Install UnrealV5 (Engine Path/Engine/Extra/UnrealV5)
- Go to Unreal
- Go to Tools/Create Visual Studio Solution
- Close Unity, open the newly created solution from Unity install directory
- Right to you find the "Solution Explorer", right click on "Solution"
  - UnrealEngine4 -> Home
  - Clean Solution -> Look what some seconds and details all binaries
  - Rebuild Solution -> takes a couple of minutes while recompiling
- Start Unity via the green arrow "Local Windows Debugger" (or F5)

# ABILITIES USAGE

**Abilities 01: Subscribe to a Gameplay Ability and react to its activation and deactivation**



We wanted to have an actor subscribing to the gameplay tasks of another class actors and able to react to these tasks.

The advantage of using the Jira Gameplay Ability system is especially the much threaded implementation of G-Subscriptions like they will not run in the main gameplay thread and avoid unnecessary polls for updates.

We implemented an emote ability on the box to the left - now we want to tie to an independent actor - a glowing sphere - react to the usage of the ability.

- Get us some dialog information - along at the game ability task every tick
- Add a **dialog** routine to Character BP
- Implement the Gameplay event of our characters to add dialog above round conversationally
- Implement a material that exposes an emote color
- Get us an actor blueprint - add the emote color to above material
- Implement a gameplay event that subscribes to defined gameplay task AD and REDUCE event - reacting by switching the sphere color

**Abilities 02: Create an Gameplay Ability Template**

[illegible]

- has dozens of tag modifiers to fine tune the ability logic. This should be preferred to implement game logic when possible
- The `GameplayCue` triggered points to a `gameplay tag` named `"GameplayCue.xxx"`. This is also referred by a `GameplayCueAuthority BP`
- The `GameplayCueAuthority BP` points to all aspects of a `gameplay cue`:
  - a force feedback effect
  - a Niagara effect
  - a sound name
  - a camera shake
- another `GameplayEffect` is defined in the `Class Defaults` to define the `CoolDown GameplayEffect`. This will block the ability temporarily