Building AI-Driven NPCs in Minutes with the Player2 Unity SDK

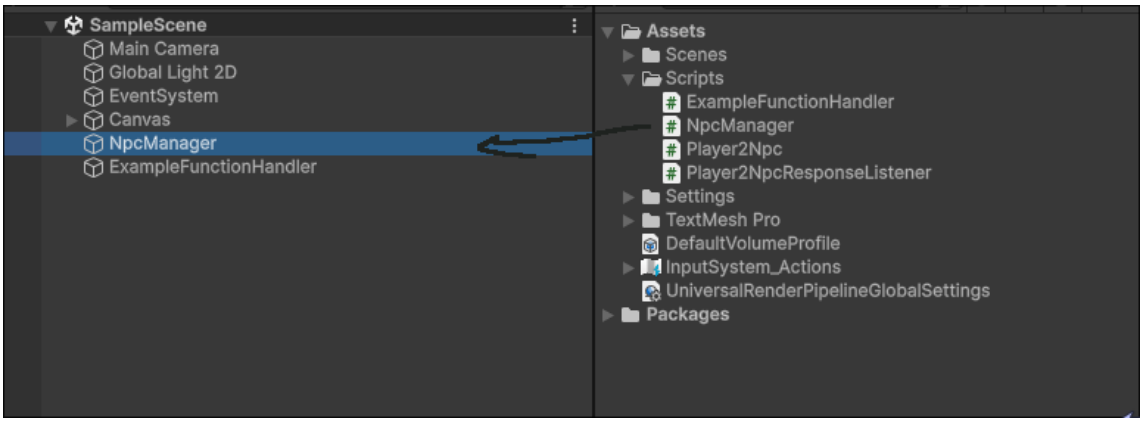
Table of contents

NpcManager

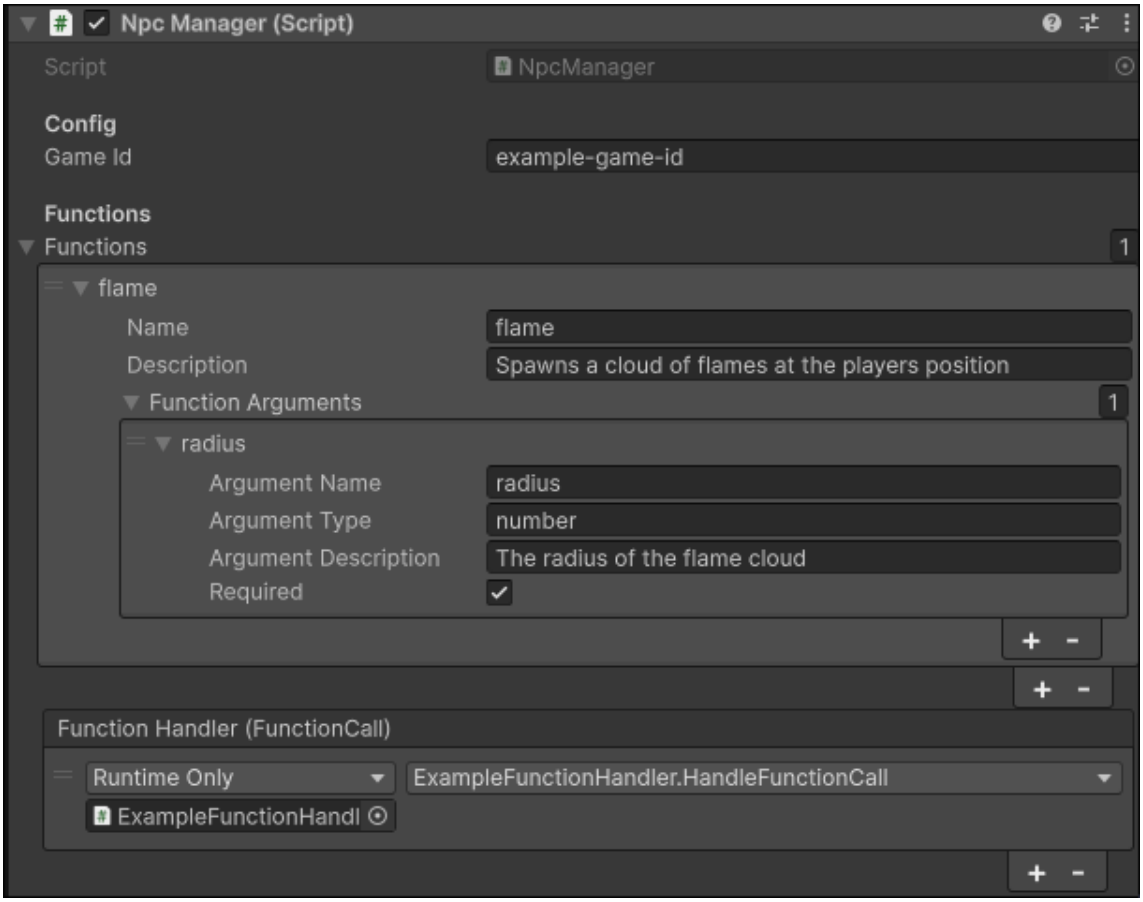
Introduction

The `NpcManager` component is the heart of the Player2 Unity SDK, allowing you to create AI-driven NPCs that can chat and perform actions in your game world.

To start integrating the player2-sdk into your project; Add `NpcManager` to your scene root, never use more than one NpcManager. It stores your *Game ID* and the list of functions the LLM can invoke.



Example setup of `NpcManager`



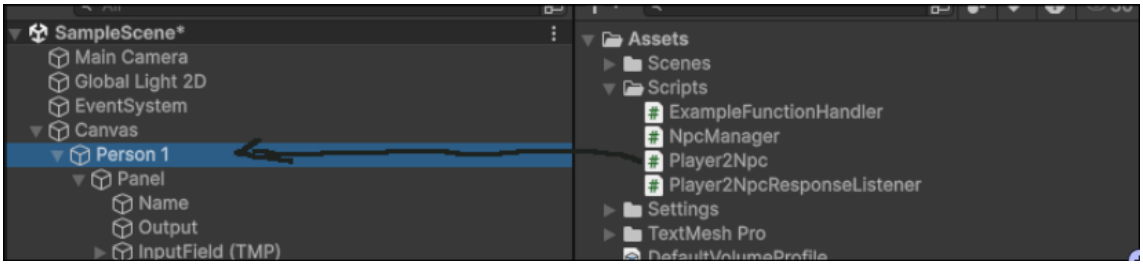
- **Game ID** – the name of the game/mod that you are making.
- **Functions** → **+** – one element per action.
    - *Name* – code & prompt identifier.
    - *Description* – natural-language hint for the model.
    - *Arguments* – nested rows for each typed parameter (e.g. `radius:number`).
        - Each argument can be specified if it is *required* (i.e. is not allowed to be null)

Example above exposes `flame(radius:number)` which spawns a fiery VFX cloud.

NPC Setup

Npc Initialisation

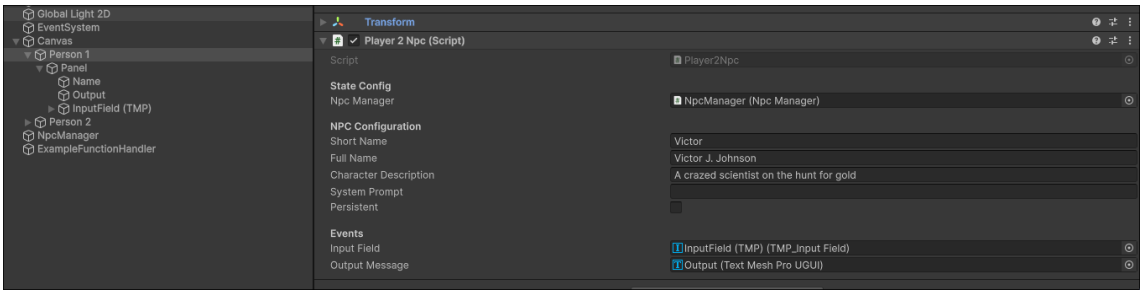
Select the GameObject that represents your NPC (`Person 1` in the image below) and add **Player2Npc.cs**.



Configure the NPC component

1. **Npc Manager** – drag the scene's NpcManager.
2. **Short / Full Name** – UI labels.
3. **Character Description** – persona sent at spawn.
4. **Input Field / Output Message** – TextMesh Pro components that your npc will listen to and output to.
5. Tick **Persistent** if the NPC should survive restarts of the Player2 client.

That's it—hit **Play** and chat away.



Adding rich NPC functions (Optional)

If you want to allow for a higher level of AI interactivity,

1. Add a script like the sample below to the Scene Root.
2. In **NpcManager** → **Function Handler**, press **+**, drag the object, then pick **ExampleFunctionHandler** → **HandleFunctionCall**.

```
using UnityEngine;

public class ExampleFunctionHandler : MonoBehaviour
{
    public void HandleFunctionCall(FunctionCall call)
    {
        if (call.name == "flame")
        {
            float radius = call.ArgumentAsFloat("radius", defaultValue: 3f);
            SpawnFlameCloud(radius);
        }
    }
```

```
    void SpawnFlameCloud(float r)
    {
        // Your VFX / gameplay code here
    }
}
```

You never respond manually; the back-end keeps streaming text while your Unity logic happens in parallel. Now, whenever the model decides the NPC should *act*, `HandleFunctionCall` fires on the main thread.