

# InworldAI Unity SDK



On this page



# Unity

## Unity

## Download

You can download our **Unity Integration package here**. Before getting started, we will walk you through our compatibility requirements, assets, and API references. TLDR, you can check the following video tutorial.

**⚠ Note:** If your current Unity SDK version is below 2.1.X. We recommend you to delete the existing `Inworld.AI` folder in Unity and import the newest one.



## Unity Integration

The **Inworld AI Unity SDK** is a powerful cross-platform virtual character integration plugin for Unity. You can easily drag and drop virtual characters into your Unity scene and communicate with them.



For a more vivid experience, our plugin can be easily integrated with other features, such as,

1. **Realistic Eye Movements**
2. **Oculus Lipsync**



On this page



# compatibility

## Compatibility

### Update

We've recently added Windows IL2CPP support and Apple Silicon (M1) support after Ver. 2.1.2.

**⚠ Note:** If your current SDK version is below 2.1.0, please delete the `Inworld.AI` folder in your project and download the newest one [here](#). Additionally, please notice that the legacy package (Ver. 2.1.0 or lower) may overwrite **Oculus lipsync** package if both packages have been installed into your project. We recommend you also upgrade **Oculus lipsync** as well.

### Unity Version

The minimum supported Unity version is 2021.3.2f1. Any version below that may not be compatible. We have tested the package on **2021.3.2f1**, **2021.3.6f1**, and **2021.3.8f1**, **2021.3.11f1**, **2021.3.14f1**, **2022.1.20f1** for both .NET Standard 2.1 and .NET framework. Other versions of Unity will be added to the list after testing.

### Platform

Here are the detailed compatibility for each platforms, scripting backends, and .net levels.

Platform	MONO	IL2CPP	.NET
Windows	√	√	.NET Standard 2.1 or .NET 4.x+
Mac Intel 64-bit	Only in Editor Mode	Only in Editor Mode	.NET Standard 2.1 or .NET 4.x+
Android	✗	√	.NET 4.x+
Oculus	✗	√	.NET 4.x+

Platform	MONO	IL2CPP	.NET
iOS	N/A	✓	.NET 4.x+
Mac Apple Silicon (M1)	Only in Editor Mode	Only in Editor Mode	N/A
Linux	✗	✗	N/A
WebGL	✗	✗	N/A

## Rendering Pipeline

Currently, the default Inworld avatar (based on Ready Player Me) is only worked in **Build-in** pipeline. But all the Ready Player Me characters you created by <https://studio.inworld.ai/> support both **Build-in** and **URP** naturally. We'll add **HDRP** support in the future. Moreover, you can always work around by creating your own **URP/HDRP** avatars and attach `InworldCharacter` scripts.

## Input System

We've noticed that our `Inworld Player Controller` is not compatible with Unity's `Input System` by default. We'll add this support in future.

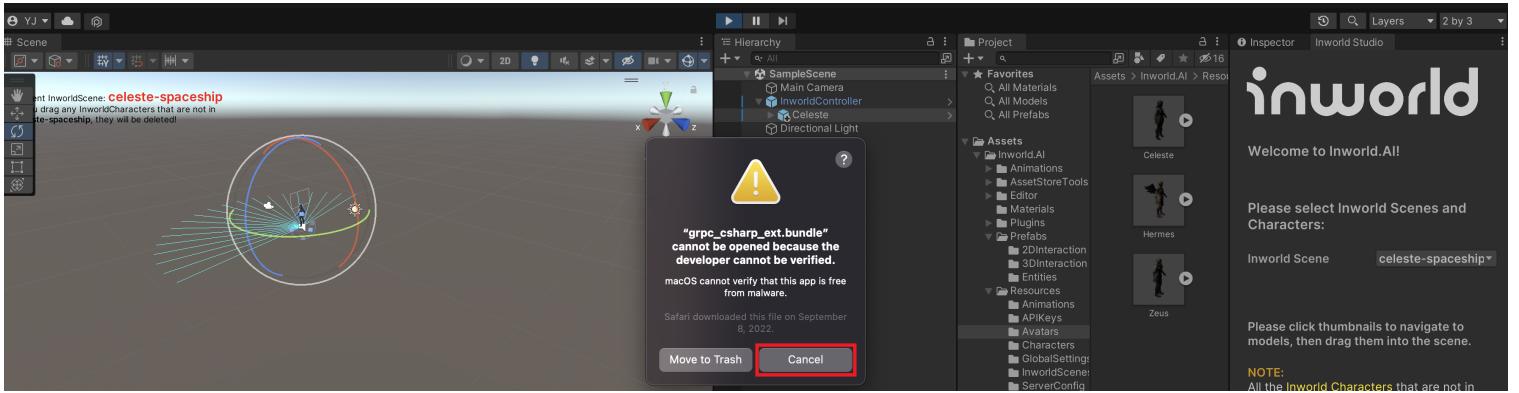
## Android

Unity cannot proceed Android build by 2021.3.6f1. It's a known bug for Unity. To solve this, you need to copy the whole **Tools** folder from the previous Unity version. Check this page for more details.

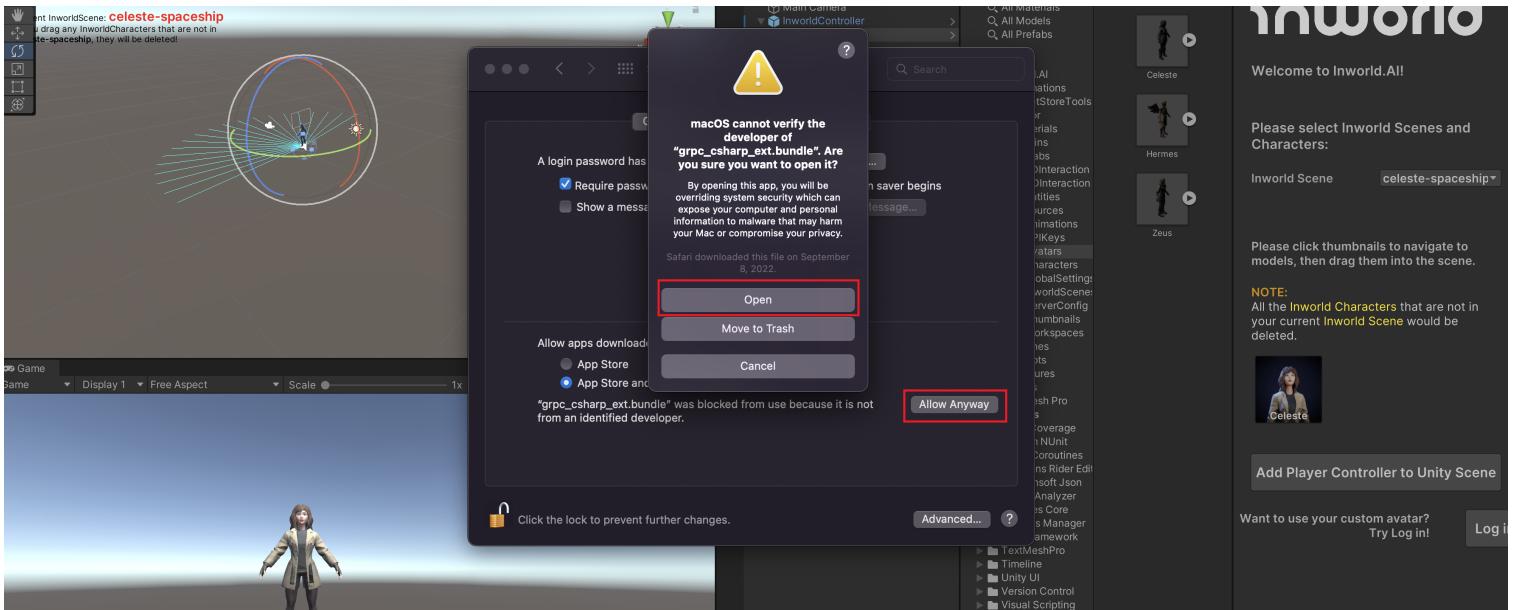
## MacOS

### 1. Allow Access for grpc bundle

MacOS will automatically delete the `grpc_csharp_ext.bundle` which is necessary to our package if privacy is not allowed. Once you click the `Play` button, the following message box will appear,



Navigate to **System Preferences > Security & Privacy** and click the **Lock** button. When the **grpc\_csharp\_ext.bundle** appears under the **Security & Privacy** panel, click **Allow anyway**.



Return back to Unity, and click **Cancel** in the message box. Do **NOT** click **Move to Trash**. The dialog may appear again. If it does, then click **Open** once you have allowed success.

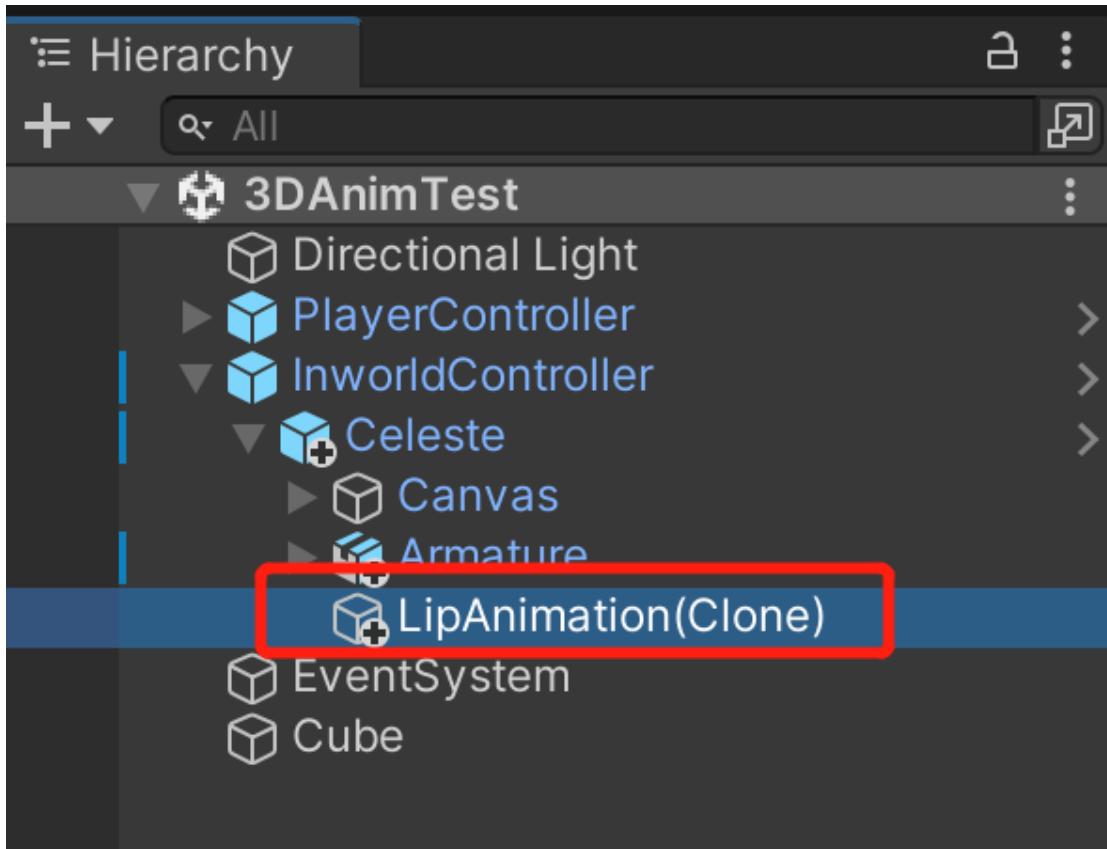
## 2. Compatibility with the MacOS build

This is a known bug for Unity, which will be fixed in version 2022.2.X.

## 3. Lipsync with M1

The current lip sync dll doesn't support M1. We'll upgrade our backend to support in future.

To work around this feature, you can just delete the **LipAnimation** below in your scenes.



## 4. Microphone Usage Description

If you are building an iOS application, then you need to add the `Microphone Usage Description` to your process build.

Configuration	
Scripting Backend	IL2CPP
Api Compatibility Level*	.NET Standard 2.1
C++ Compiler Configuration	Release
Use incremental GC	<input checked="" type="checkbox"/>
Assembly Version Validation (editor only)	<input checked="" type="checkbox"/>
Camera Usage Description*	
Microphone Usage Description*	ADD_YOUR_DESCRIPTION
Location Usage Description*	ADD_YOUR_DESCRIPTION
Use on-demand resources*	<input type="checkbox"/>
Accelerometer Frequency*	60 Hz
Mute Other Audio Sources*	<input type="checkbox"/>

On this page



# get-started

## Getting Started

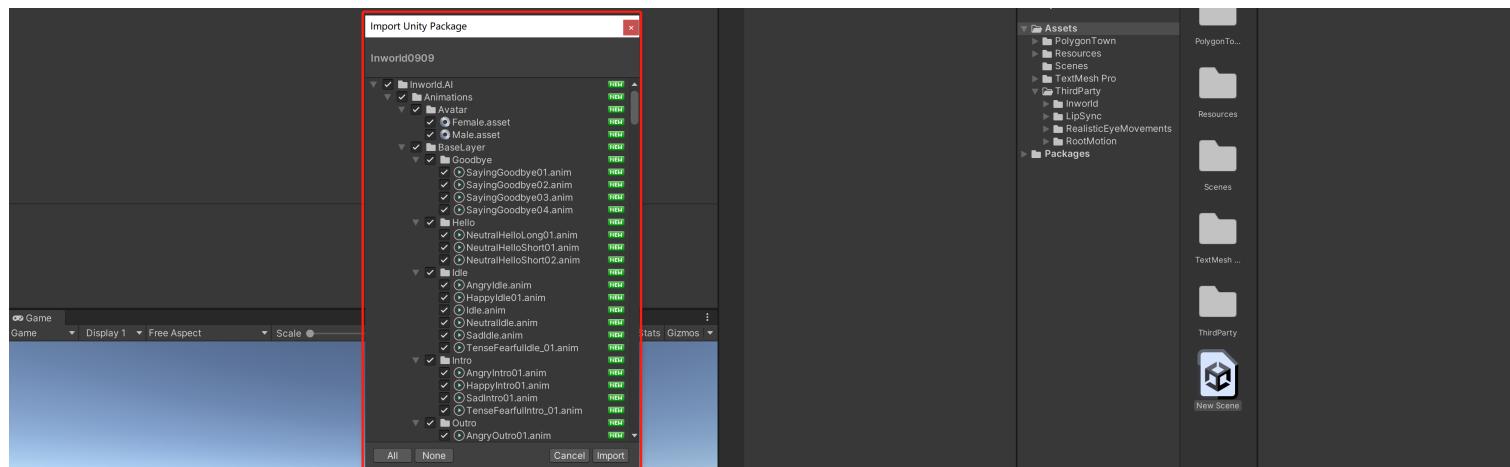
Our Unity Integration package comes with several sample scenes. Before integrating your own scene, we highly recommend you go through these samples first.

### 1. Importing the Package

You can choose any of the following to import the **Inworld AI Unity SDK** `unitypackage` into your scene.

- Open your project in Unity and import the Asset Package file at `Assets > Import Package > Custom Package`, then navigate to the `unitypackage` you downloaded to import.
- OR Drag the `unitypackage` you downloaded into Unity's `Project` panel.
- OR Simply double-click the downloaded `unitypackage` if you have only one Unity instance running.

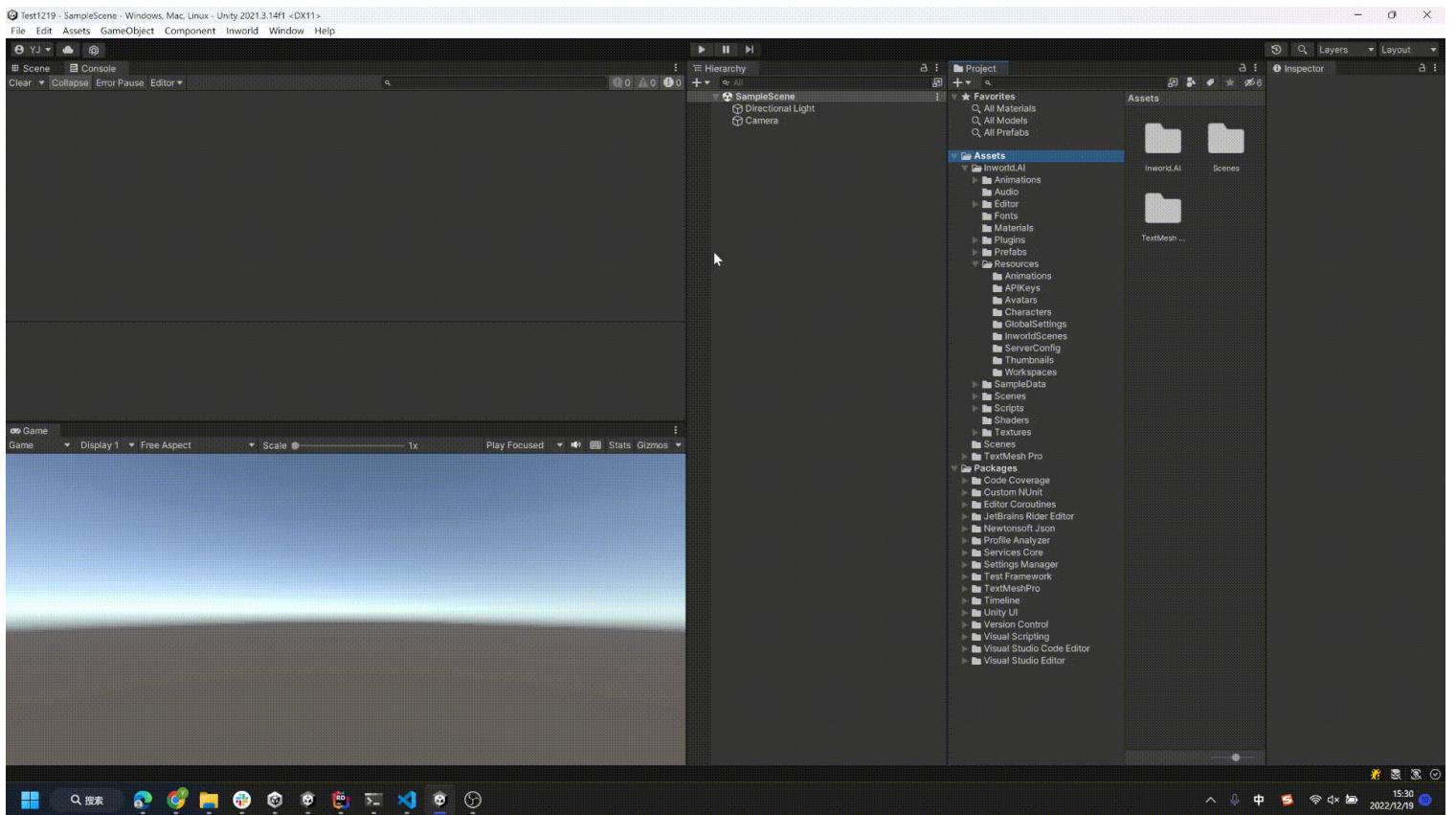
When the import dialog appears, select `Import`:



**⚠ Note:** If you've used an old version of **Inworld AI Unity SDK** before, it is recommended that you delete the folder `Inworld.AI` in your `Asset` folder before importing, as the new package may not be compatible.

### 2. Opening the basic sample scene

In `Project Panel`, click `Assets > Inworld.AI > Scenes > SampleBasic` to open the basic sample scene.



**⚠ Note:** The **Inworld AI Unity SDK** requires **Text Mesh Pro**, which is inside Unity but not imported by default. If it is the first time you are opening the **Inworld Studio Panel**, please click **Import** if the dialog for the **Text Mesh Pro**

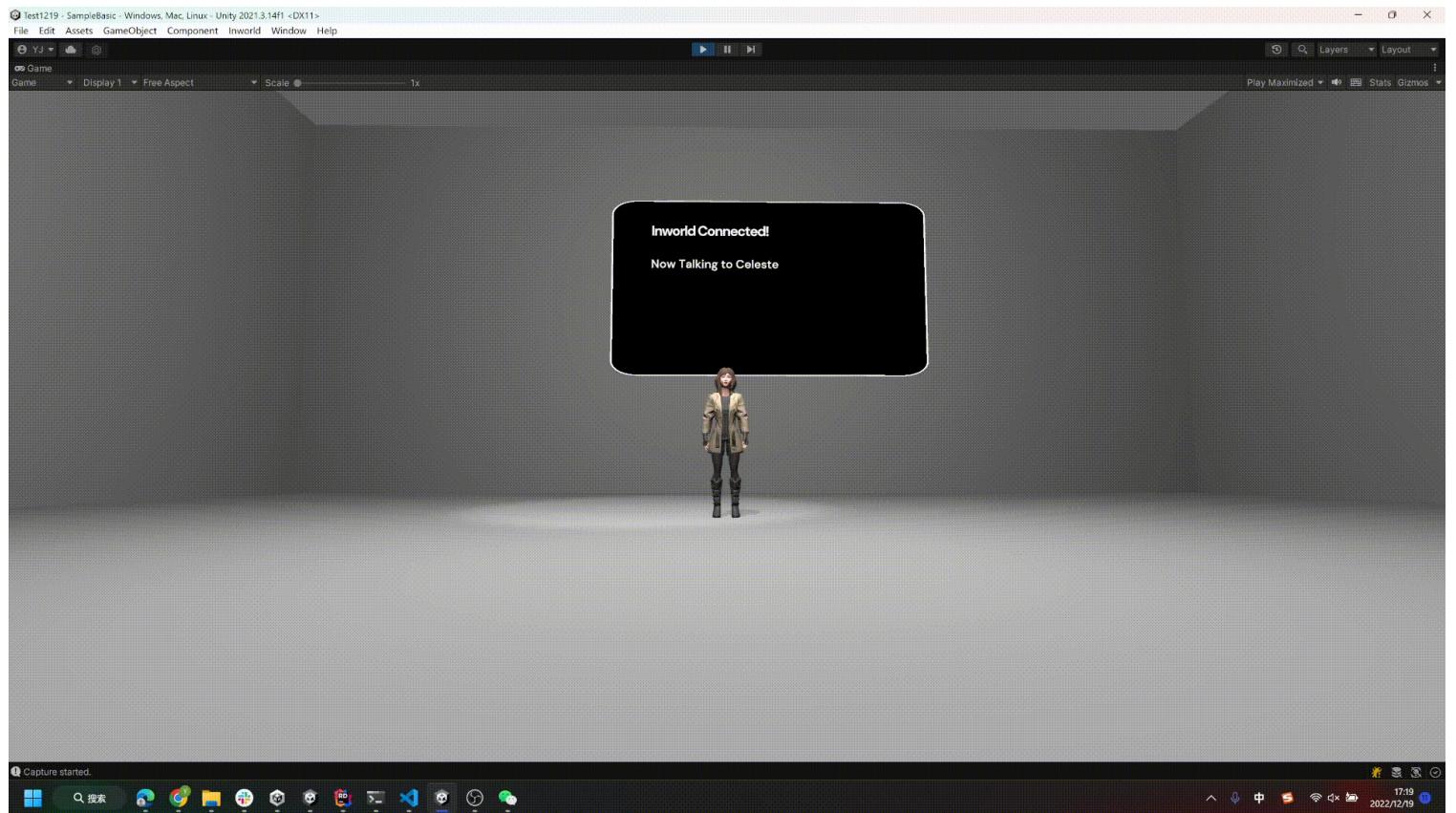
### 3. Talking to the character

After press **Play** button, Here's a monitor showcasing the connecting status to Inworld Server.

Here is a character **Celeste** in the room. Once server is connected, she will wave at you if you move closer. You can talk to her then.

You can use button **WASD** of your keyboard to move forward, back, left, and right.

You can directly speak to her with your microphone.

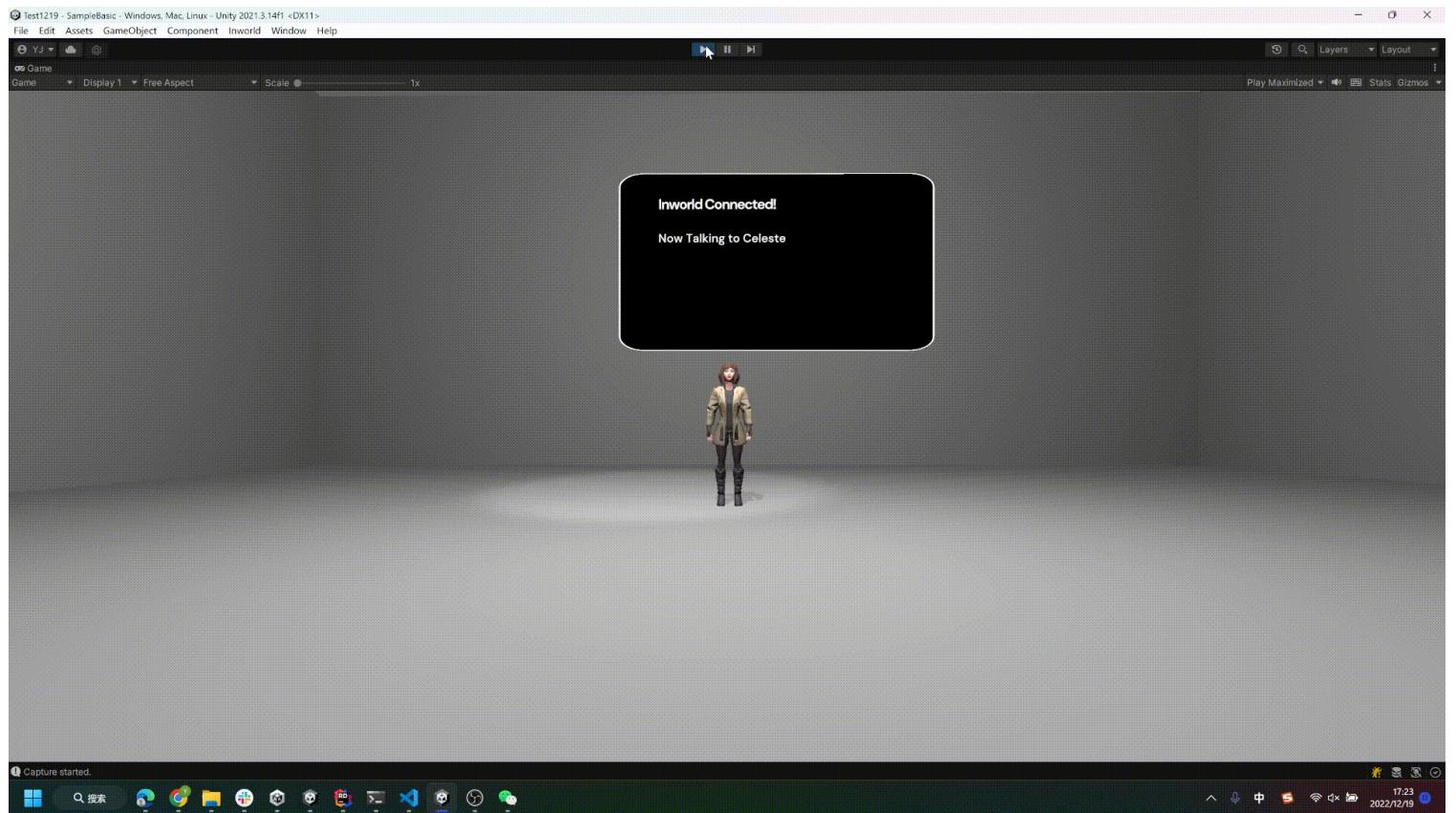


**⚠ Note:** If you change your audio input in the runtime, the new voice will not be recognized, please restart the app to take effect.

## 4. Typing to the character

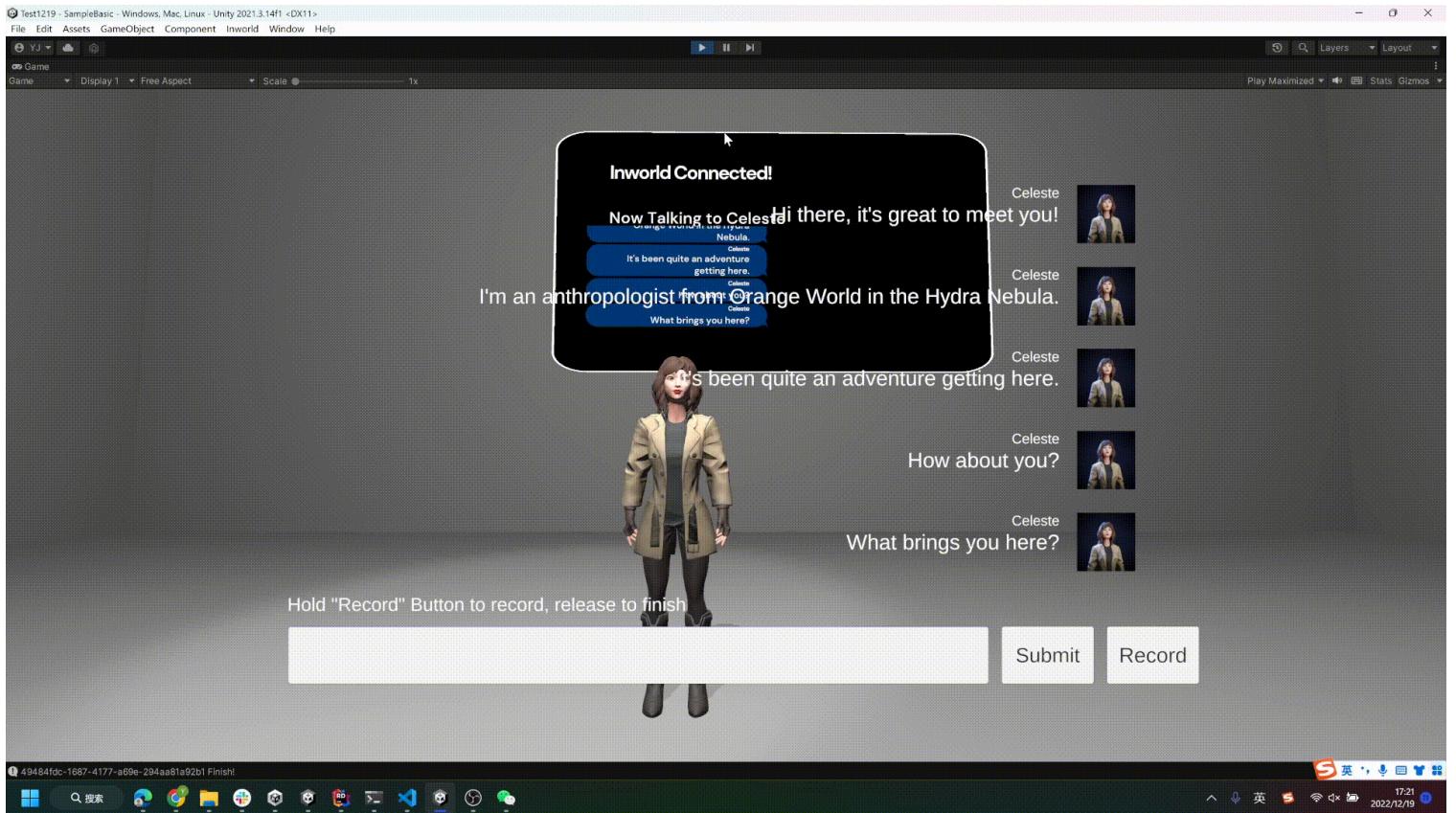
Press button  to open close the text input panel in run time.

You can type sentences to character while the text input panel is open.



## 5. Voices to text

While the text input panel is open, you could hold **Record** button to record your sentences with your microphone, and release to send.



On this page



# sample-manual

## Connection control demo

Usually, whenever you start a game with InworldController, we'll automatically connect to server and try to reconnect due to inactivity.

In this demo, we'll showcase how to let you have more controls about the connection. Such as when to connect to server, and when to disconnect. So that you can optimize your connection time by your decision.

### 1. Opening the sample scene

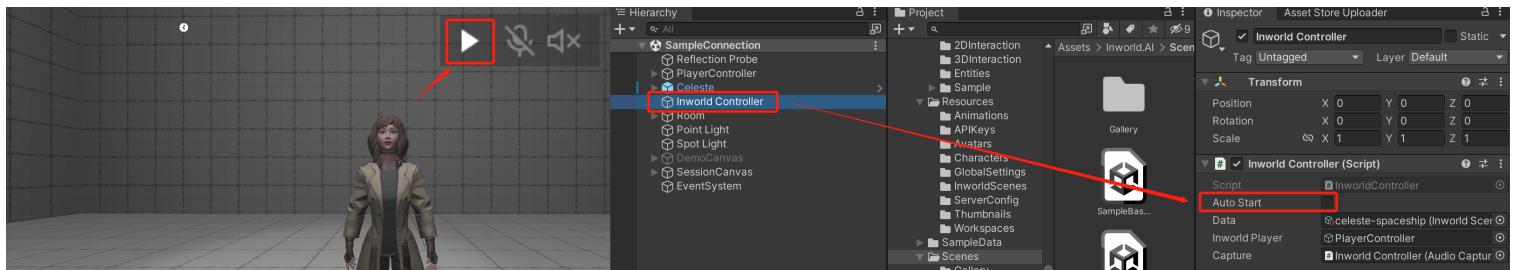
In Project Panel, click Assets > Inworld.AI > Scenes > SampleConnection to open the demo scene for connections.



### 2. Connect to server

In this demo, we won't automatically connect to server, and the character won't talk to you.

You'll need to manually click **Play** button to connect.



If it's the first time start the scene, by clicking **Play** button you'll call `InworldController::Init()`, which will perform some additional process for the first time such as acquiring session tokens, etc.

If you terminate the session and click **Play** again in the run time, you'll call `InworldController::Reconnect()`, which makes connection time much shorter.

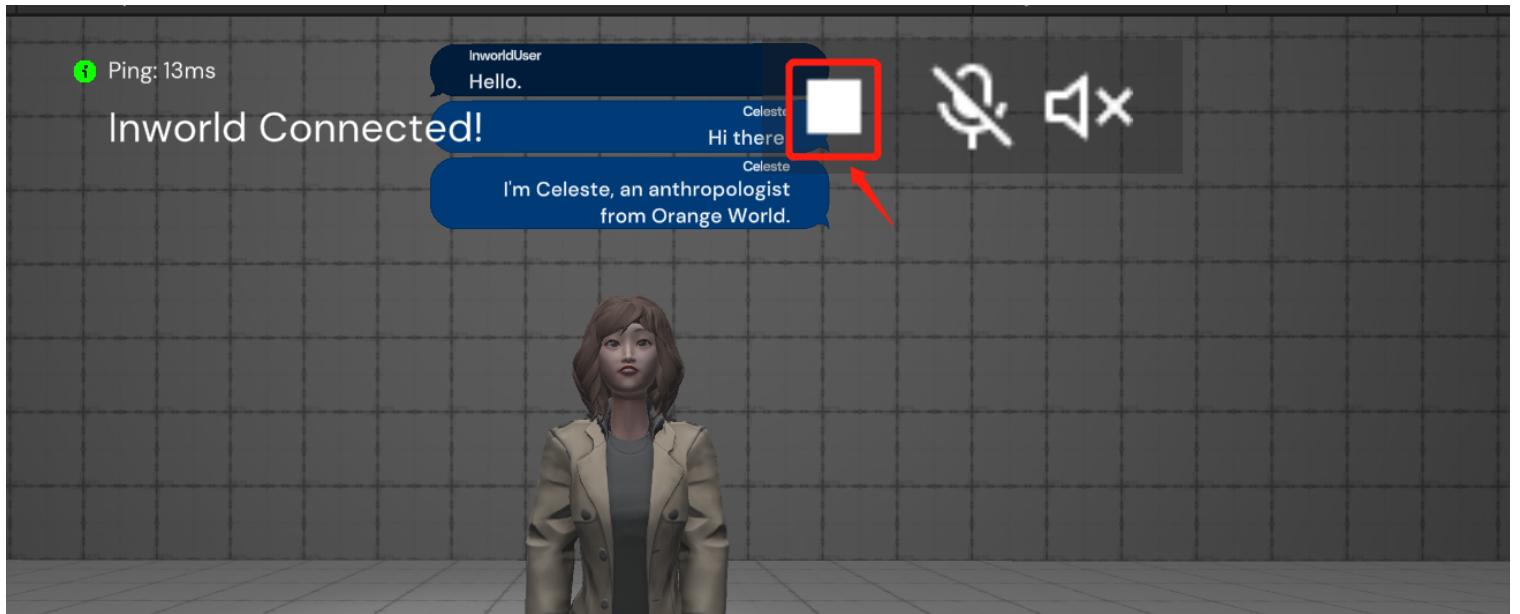
Please check our **API Reference for InworldController.cs** for more details.

```
public async void PlayPause()
{
    if (m_PlayPause.isOn)
    {
        if (!m_HasInit)
            InworldController.Instance.Init();
        else
            InworldController.Instance.Reconnect();
    }
    else
        await InworldController.Instance.Disconnect();
}
```

### 3. Disconnect

Once connected, the **Play** button would become **Stop** button, for you to manually terminate the communication session.

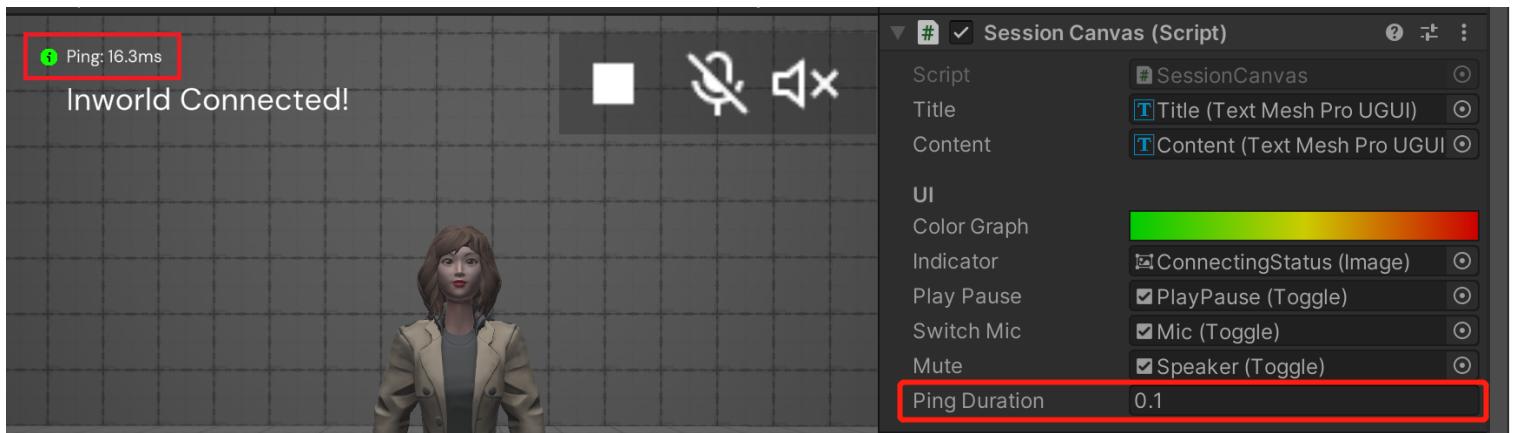
By clicking this button you'll call `InworldController::Disconnect()`. Please check our **API Reference for InworldController.cs** for more details.



## 4. Connection status

Once the client has connected Inworld server, we'll showcase the connection status. The ping latency would be displayer on the left.

You could make the **Ping Duration** in the **SessionCanvas** component larger to reduce the ping calls. By default it'll call every 0.1 seconds.



## 5. Microphone and Audio

Once connected, the switch buttons for microphone and audio would be interactable.

You could click **Microphone** button to switch on/off the audio capturing, and click **Audio** button to switch on/off of the audio receiving.

By clicking **Microphone** button, you will change the boolean property of **IsCapturing** of **InworldController**.

By clicking **Audio** button, you will change the boolean property of **IsMute** on **AudioInteraction** of that **InworldCharacter**.

Please check **API Reference** for more details.

```
public void MicrophoneControl()
{
    InworldController.IsCapturing = !m_SwitchMic.isOn;
}

public void SwitchVolume()
{
    InworldController.Instance.CurrentCharacter.Audio.IsMute = m_Mute.isOn;
}
```

On this page

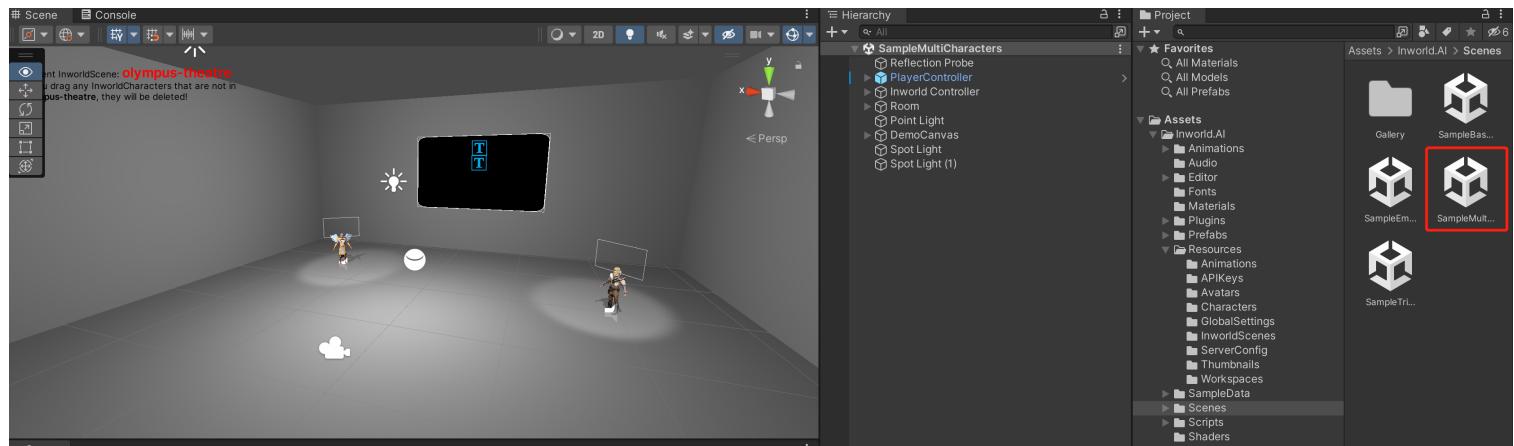


# sample-multi-characters

## Multiple characters demo

### 1. Opening the sample scene

In [Project Panel](#), click [Assets > Inworld.AI > Scenes > SampleMultiCharacters](#) to open the demo scene for supporting multiple characters.

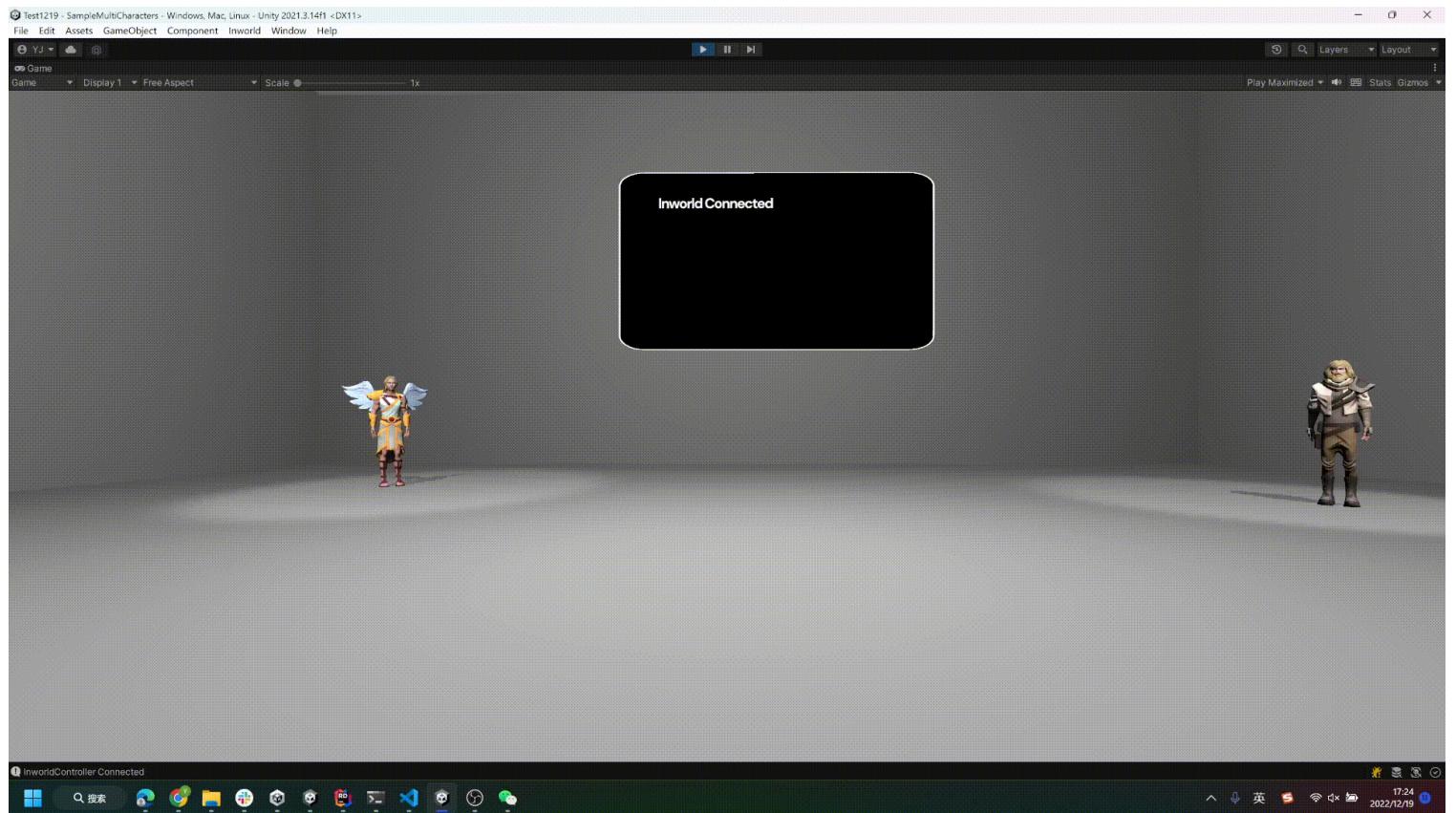


### 2. Talking to the characters

Here are 2 characters **Hermes** and **Zeus** in the room. Once connected, you can talk to them.

Use button **WASD** of your keyboard to move forward, back, left, and right.

You can directly speak to them with your microphone.



### 3. Selecting character

Although multiple characters can be loaded in a single scene, you are still allowed to talk to one character at a time.

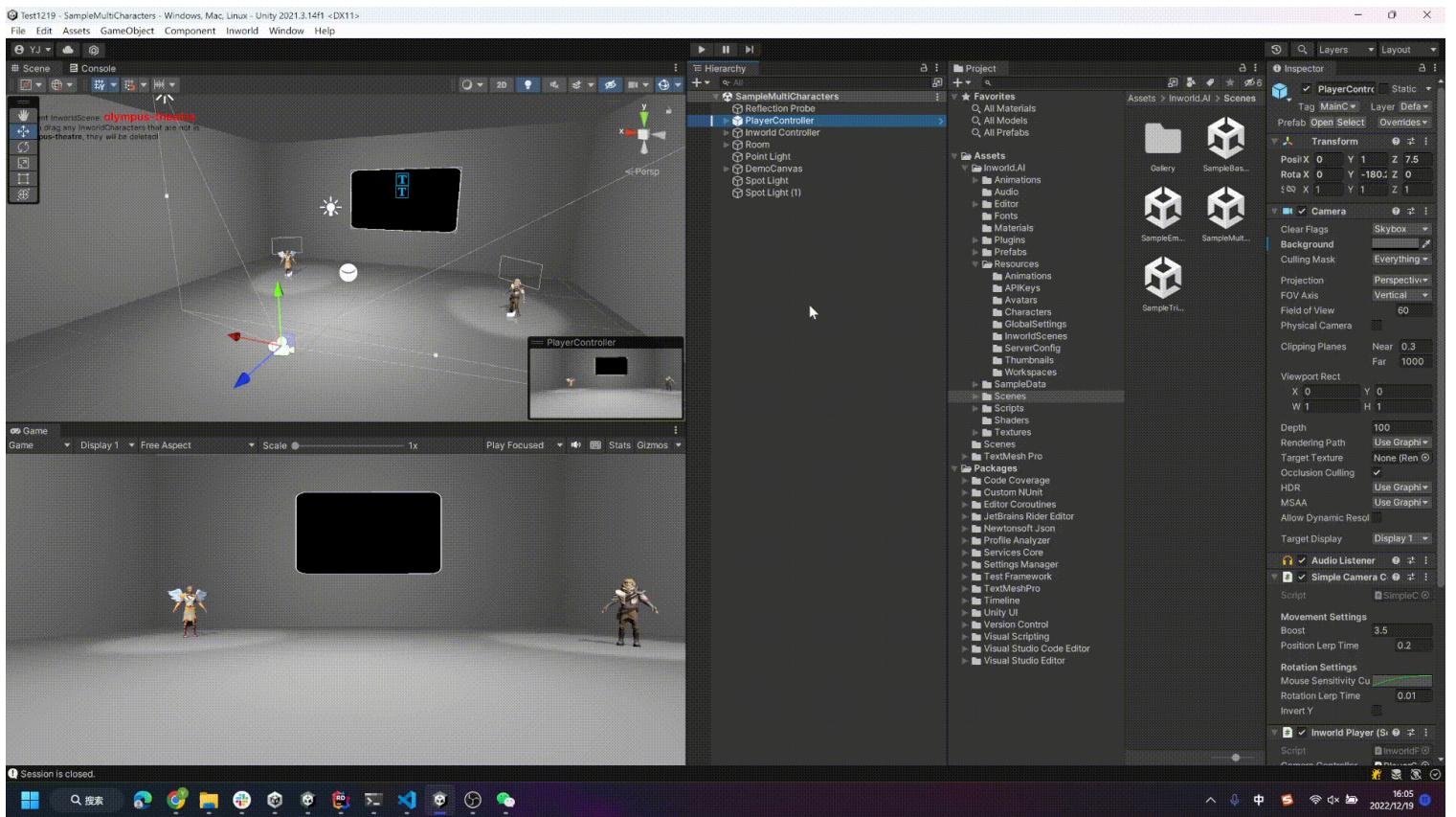
**⚠ Note:** Additionally, We used **turn-based** strategy to reduce server calls. Which means, when any character is talking, you are not able to switch characters and talk to any of them before they finish.

Which character is currently selected is defined by their sight and who you are looking at.

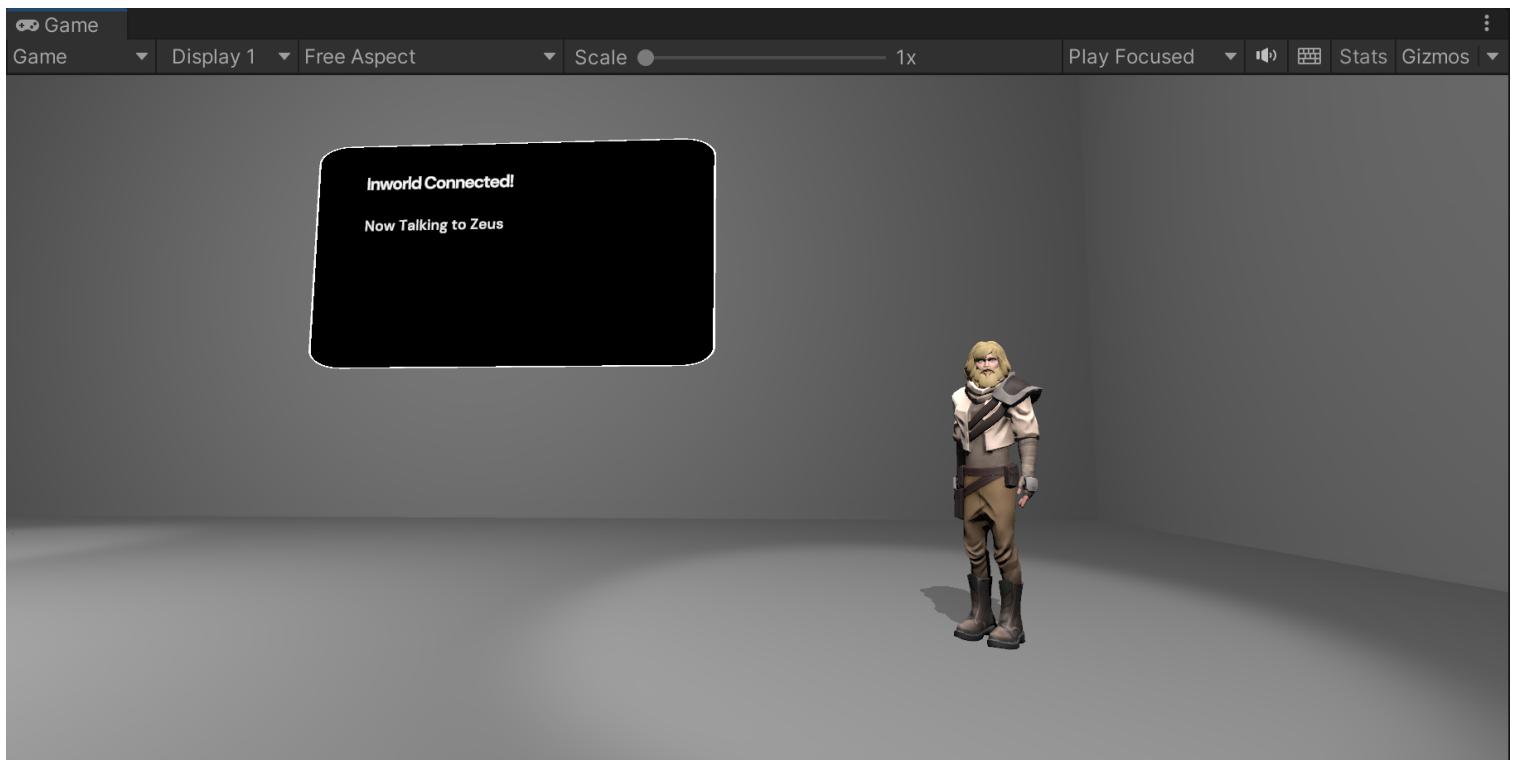
Their sight are also adjustable in **Editor Mode**. You can adjust their sight angle and distance.

The blue lines are showcasing if they can see you (the **Player Controller** needs to be in these areas)

The red lines are calculating the angle between where you look at and where the character looks at. Which means, if you are in multiple character's zone, the narrower the red angle is, the character will be selected.



In runtime, the monitor will display current which character you are talking to.

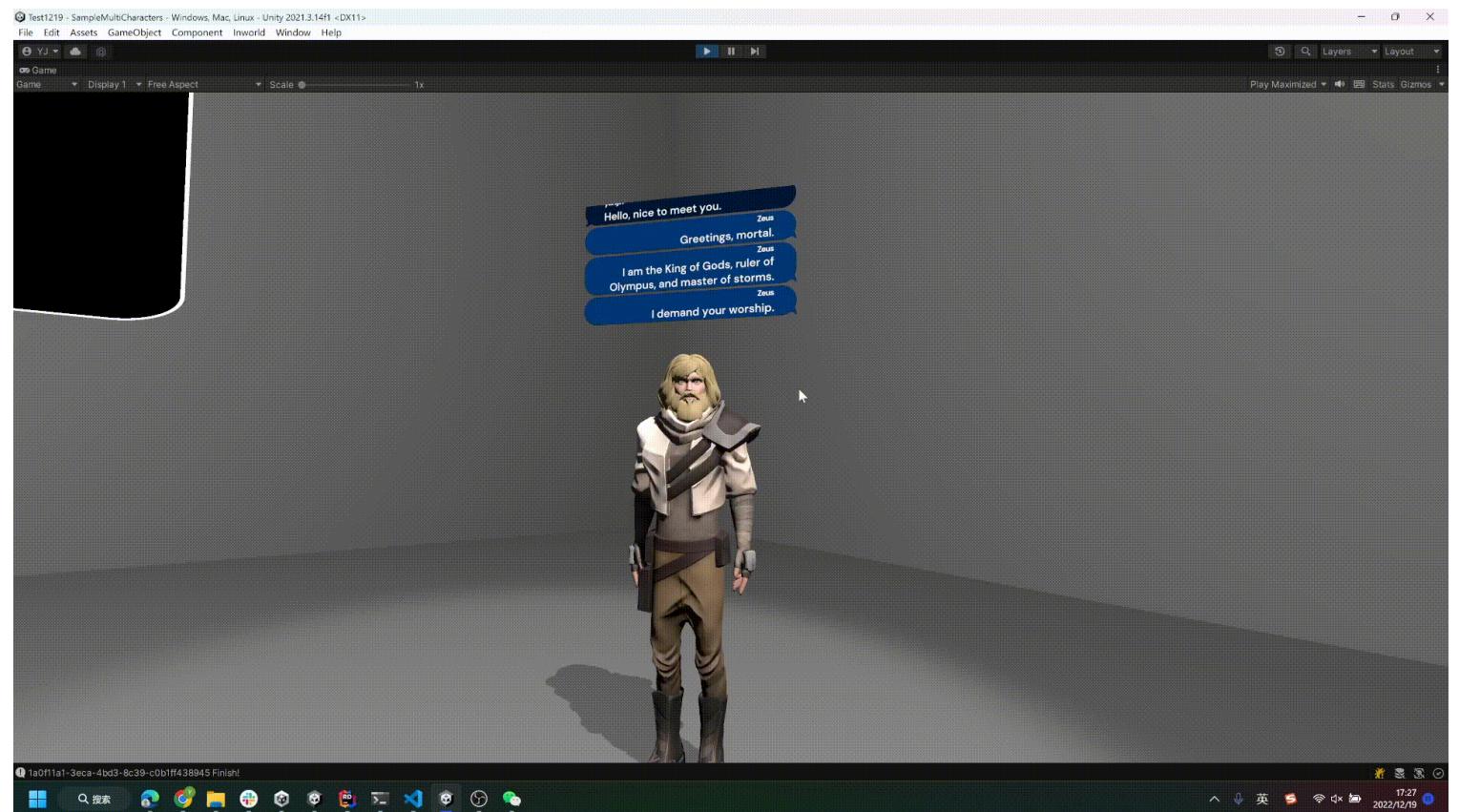


## 4. Typing and Recording text to the characters

You can also press button ~ to open close the text input panel in run time.

You can check the history logs for all the characters in the scene.

You can type or record sentences to character while the text input panel is open, but only the current selected character you can communicate with.



On this page

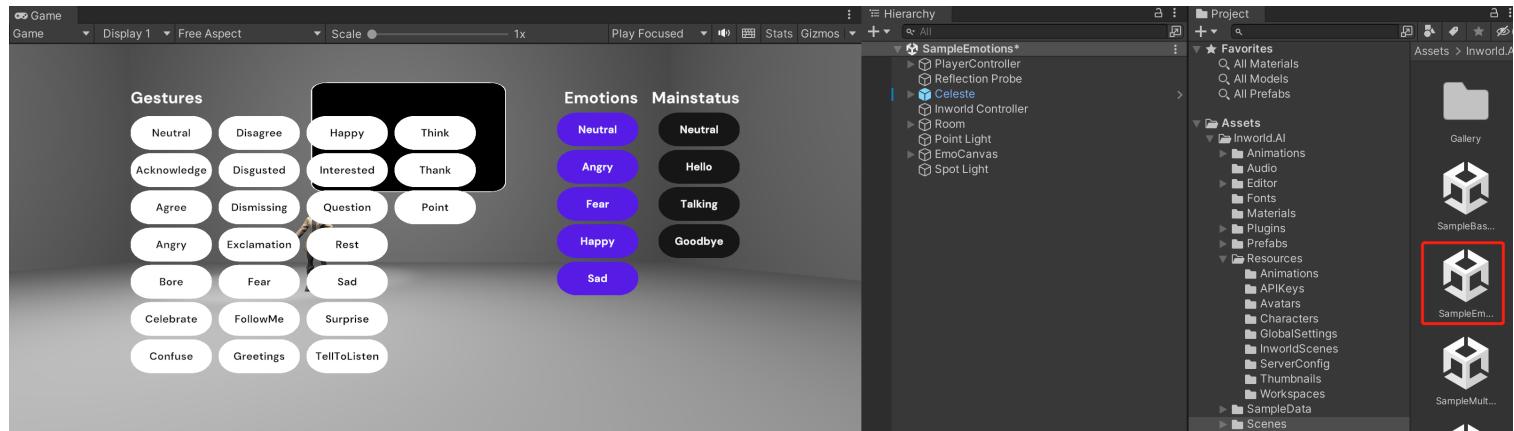


# sample-emotion

## Emotion and animation demo

### 1. Opening the sample scene

In **Project Panel**, click `Assets > Inworld.AI > Scenes > SampleEmotions` to open the demo scene for showcasing emotions and animations.

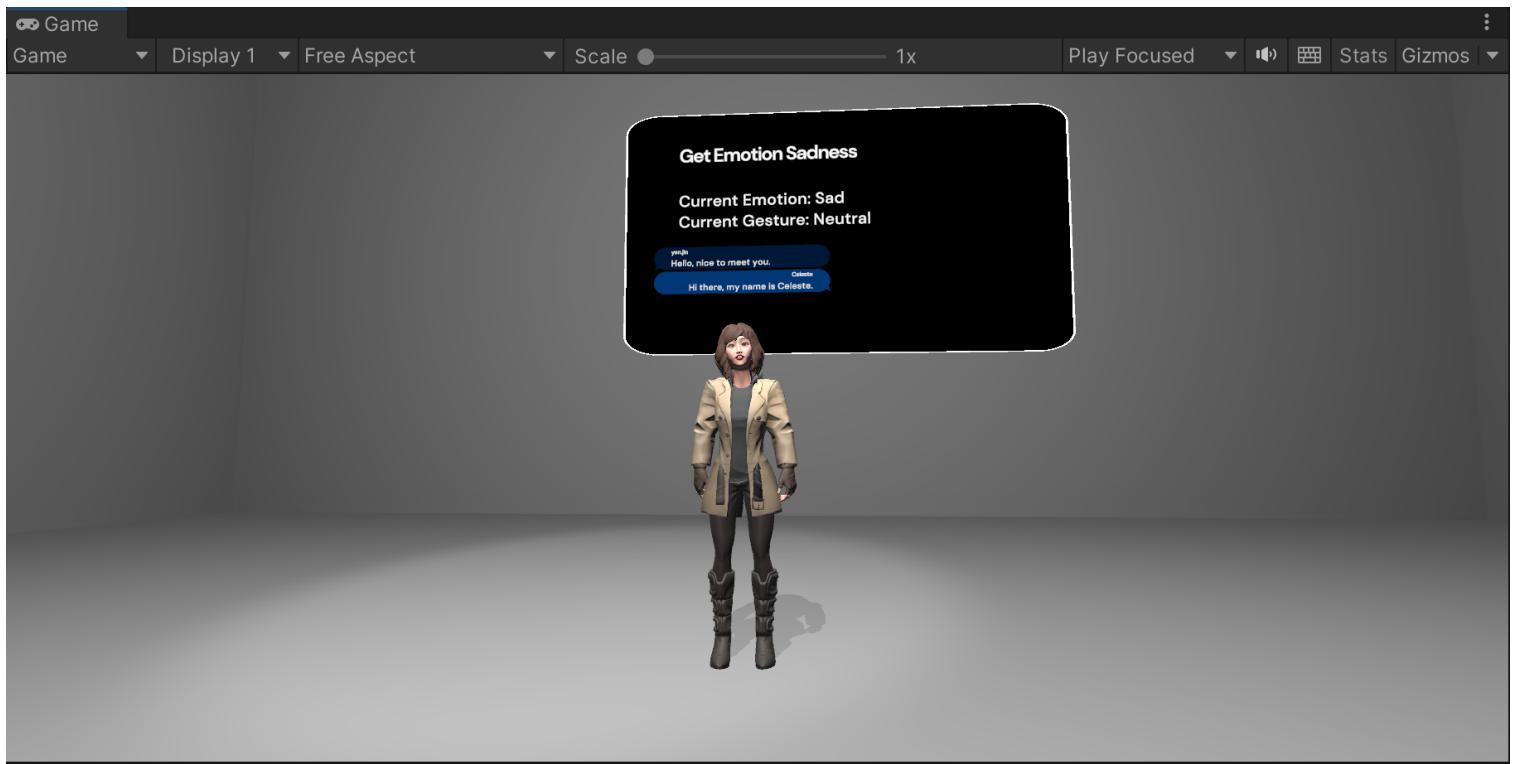


### 2. Talking to the characters

Here is a characters **Celeste** in the room. Once connected, you can talk to them.

On the monitor, it will display the current character's emotions and gestures returned from server.

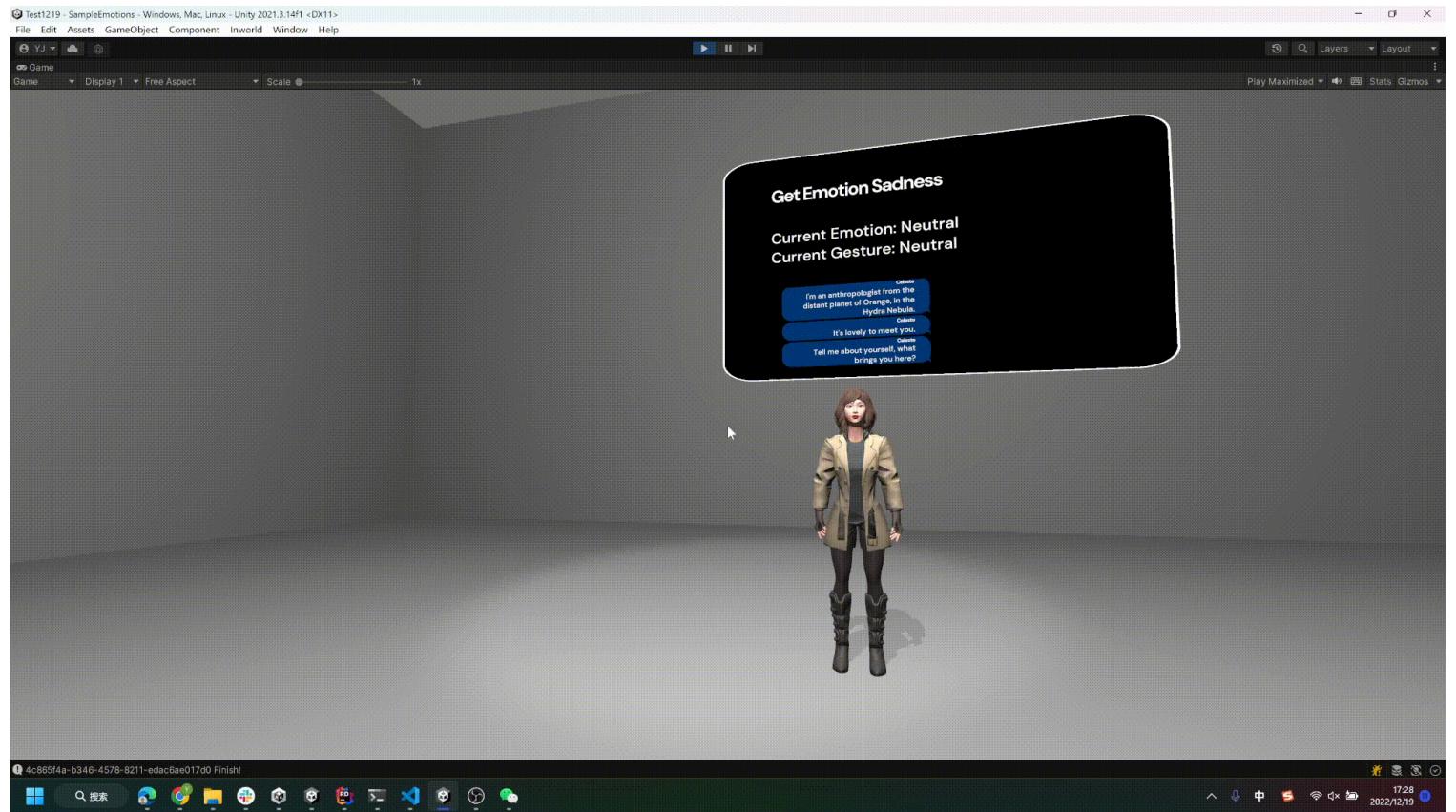
The character will perform the correspondent emotions and animations.



### 3. Testing Animations

In this scene, press button  to open close the animation controller in run time.

You can click buttons to test the character's animations.



## 4. References

For the detailed animation structure, please visit [Animation page](#)

On this page

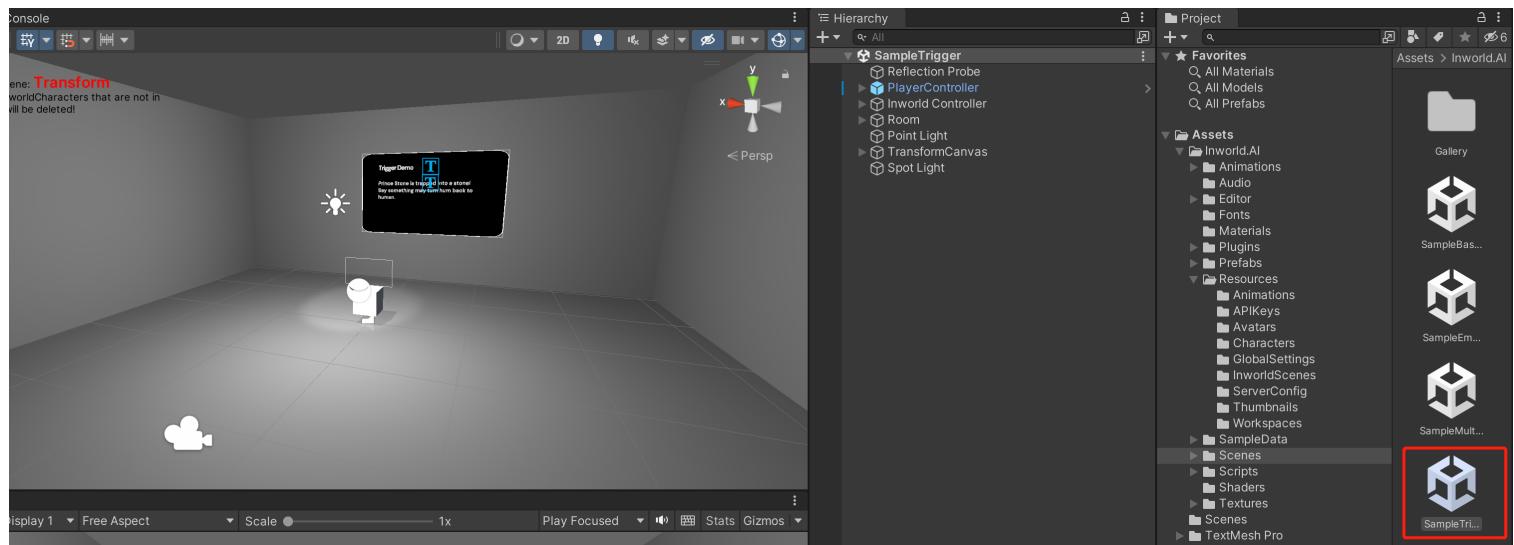


# sample-trigger

## Trigger Demo

### 1. Opening the sample scene

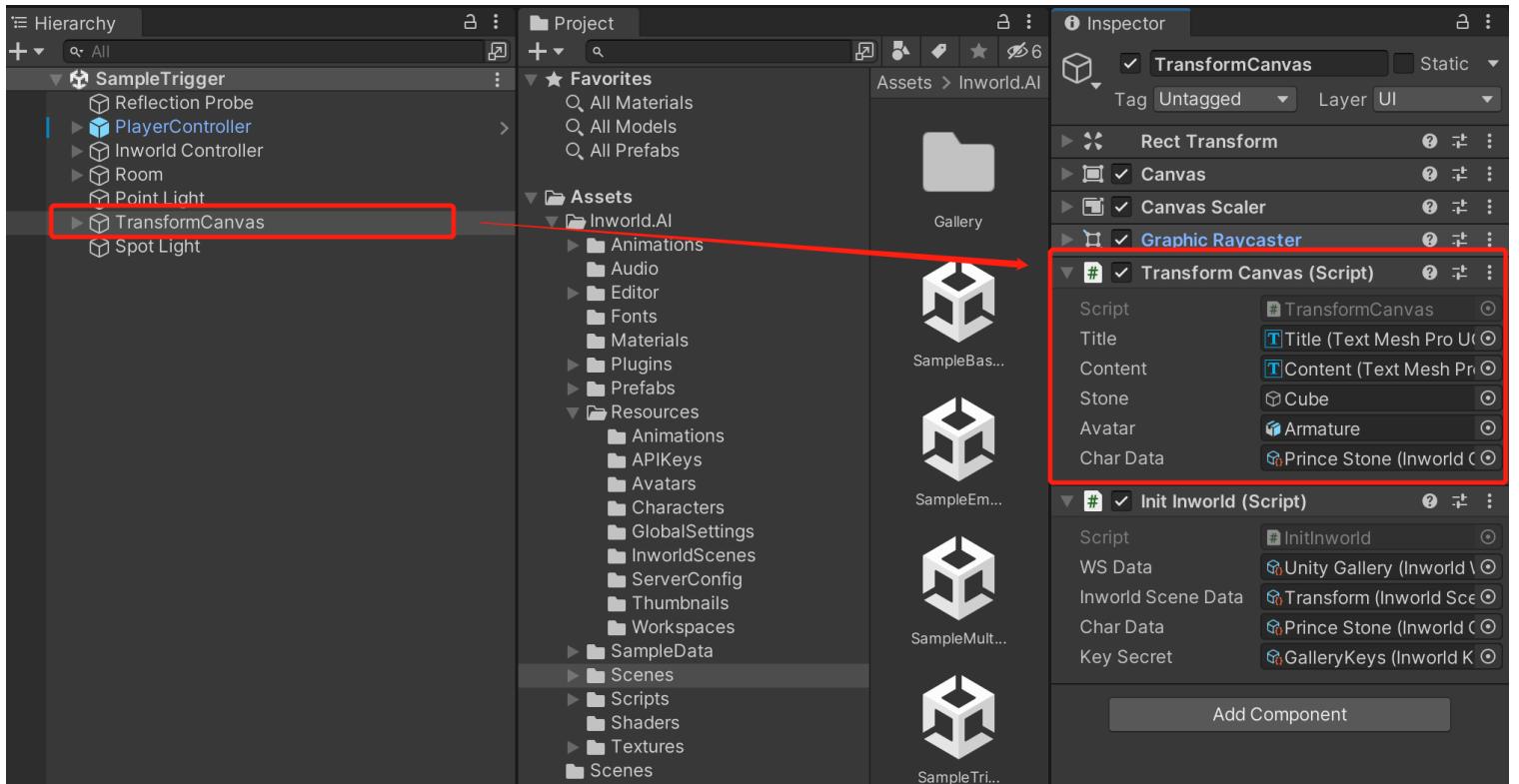
In Project Panel, click Assets > Inworld.AI > Scenes > SampleTrigger to open the demo scene of sending triggers.



### 2. Scene description

In this scene, there is a character **Prince Stone** in the room. But he is a stone (cube specifically) at the beginning.

In the script `TransfromCanvas` that attached in this scene, whenever the character has registered in the server in runtime, we'll immediately send its first trigger `Init`.



```

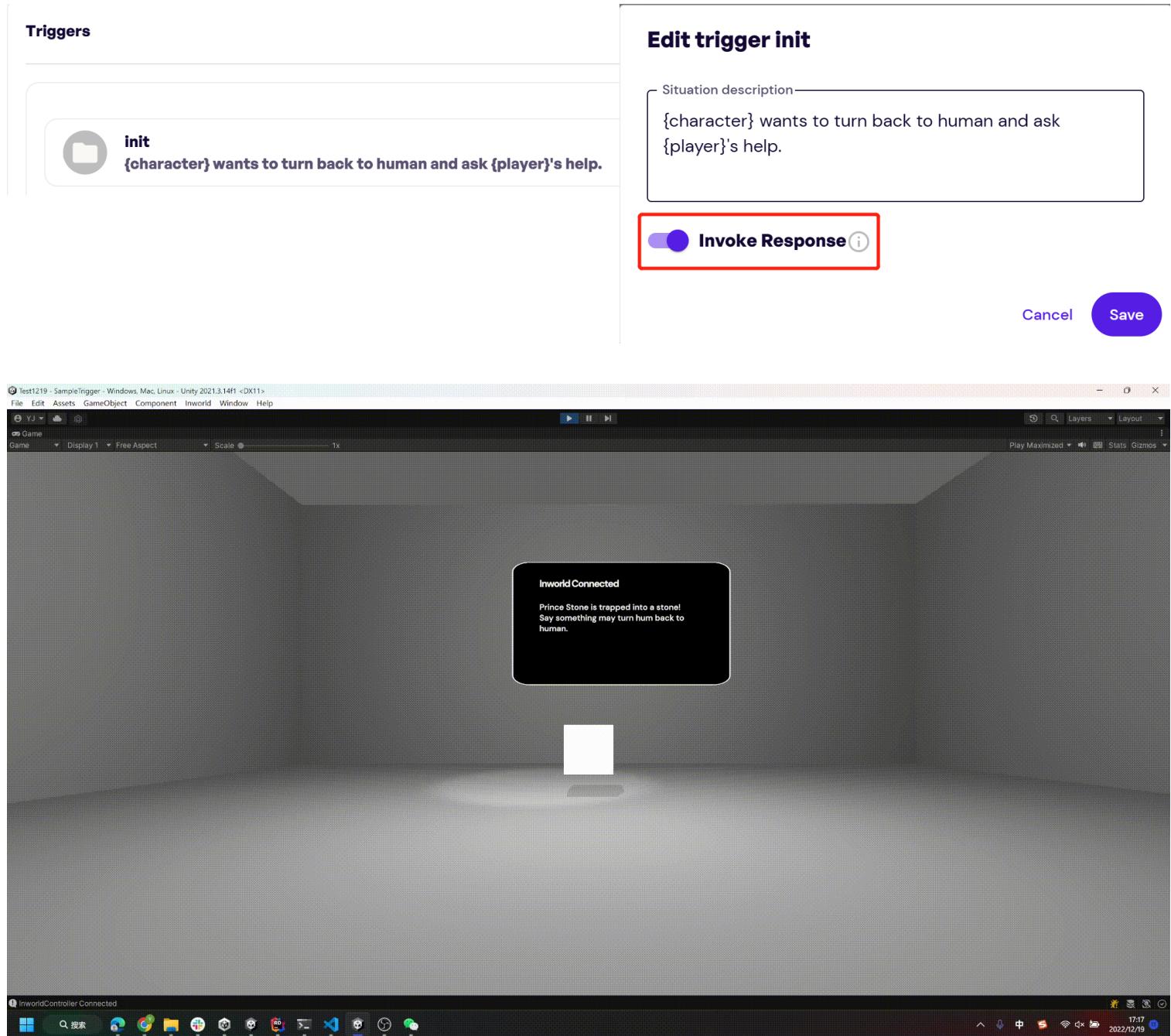
void Start()
{
    InworldController.Instance.OnStateChanged += OnStatusChanged;
    InworldController.Instance.OnCharacterChanged += OnCharacterChanged;
    InworldController.Instance.OnPacketReceived += OnPacketEvents;
}

protected override void OnCharacterChanged(InworldCharacter oldCharacter, InworldCharacter newCharacter)
{
    if (!newCharacter && oldCharacter)
        m_Title.text = $"{oldCharacter.transform.name} Disconnected!";
    else
    {
        m_Title.text = $"{newCharacter.transform.name} connected!";
        m_CurrentCharacter = newCharacter;
        if (m_CurrentCharacter.Data.characterName == m_CharData.characterName)
        {
            m_CurrentCharacter.SendTrigger(m_CurrentCharacter.Data.triggers[0]);
        }
    }
}

```

This is a **Invoke Response** trigger, which means when the trigger is sent to server, the character will respond immediately.

So instead of you talk to the character first, in this scene, this character will talk to you first.



### 3. Sending Triggers

In this scene, the actual answer to cure that curse is `www`. So if you speak `www` (or related) to him, he will set back to human with sending trigger `www` to server. You can see, the character would response differently and thank for your help if `www` is triggered.

```
void OnPacketEvents(InworldPacket packet)
{
    if (!InworldController.Instance.CurrentCharacter)
        return;
```

```

string charID = InworldController.Instance.CurrentCharacter.ID;
if (packet.Routing.Target.Id != charID)
    return;
if (packet is TextEvent textEvent)
    _HandleTextEvent(textEvent);
}

void _HandleTextEvent(TextEvent textEvent)
{
    int nWCount = 0, nstartIndex = -1, nEndIndex = -1;
    for (int i = 0; i < textEvent.Text.Length; i++)
    {
        if (textEvent.Text[i] != 'W' && textEvent.Text[i] != 'w')
            continue;
        nWCount++;
        if (nstartIndex == -1)
            nstartIndex = i;
        else
            nEndIndex = i;
    }
    // YAN: Have some margin as the answer "www" is not recognized well.
    if (nWCount >= 3 && nEndIndex - nstartIndex < 5 && nEndIndex - nstartIndex > 0)
    {
        m_CurrentCharacter.SendTrigger(m_CurrentCharacter.Data.triggers[1]);
        m_Stone.SetActive(false);
        m_Avatar.SetActive(true);
    }
}

```

## Triggers



**init**

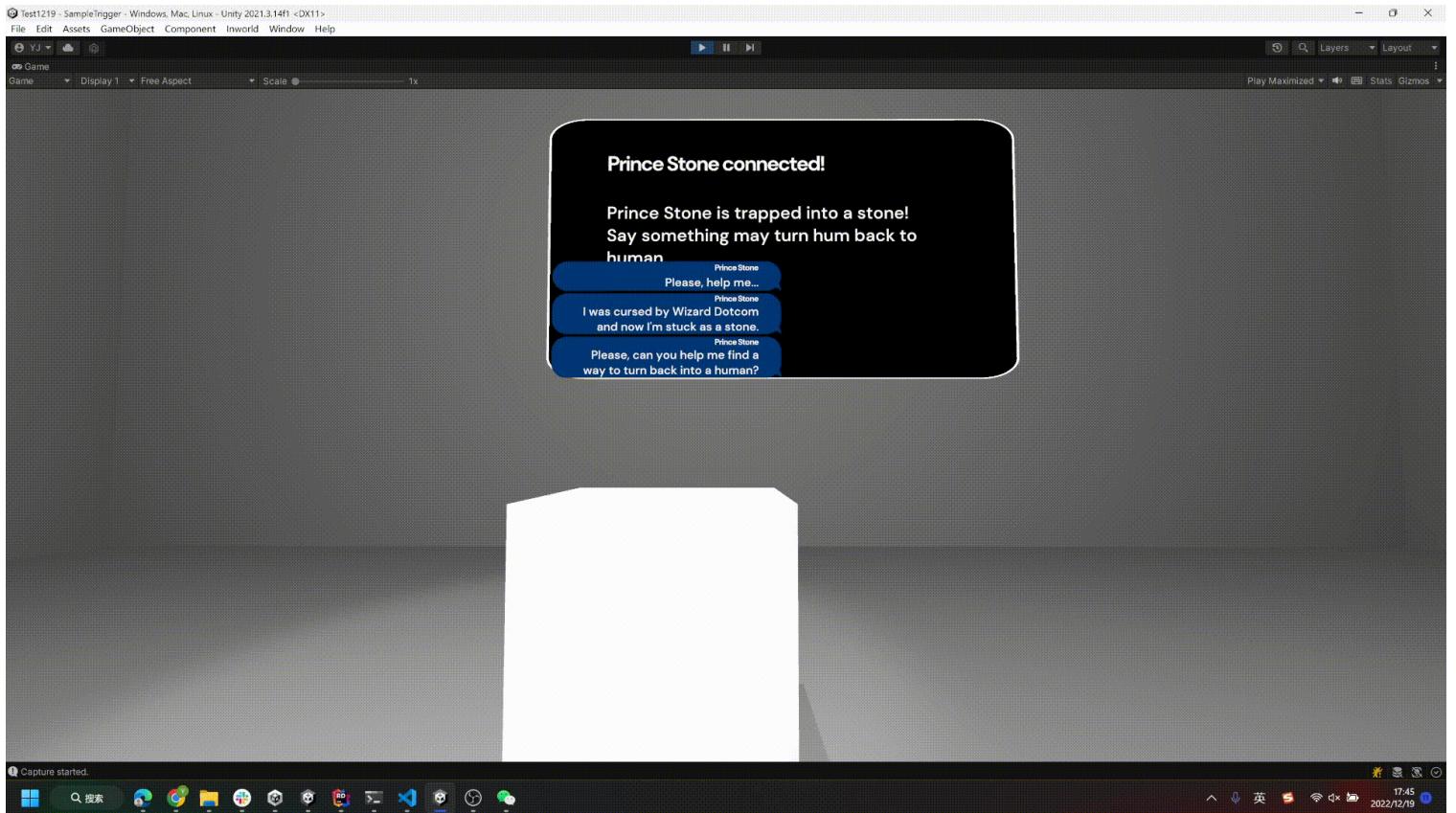
{character} wants to turn back to human and ask {player}'s help.



**www**

{character} is cured by this spell. {character} is now turned into a man! {character} is surprised that he finally turns back to human, {character} thanks {player} a lot!





On this page



# integrate-to-your-scene

## Integrate Inworld to your scene

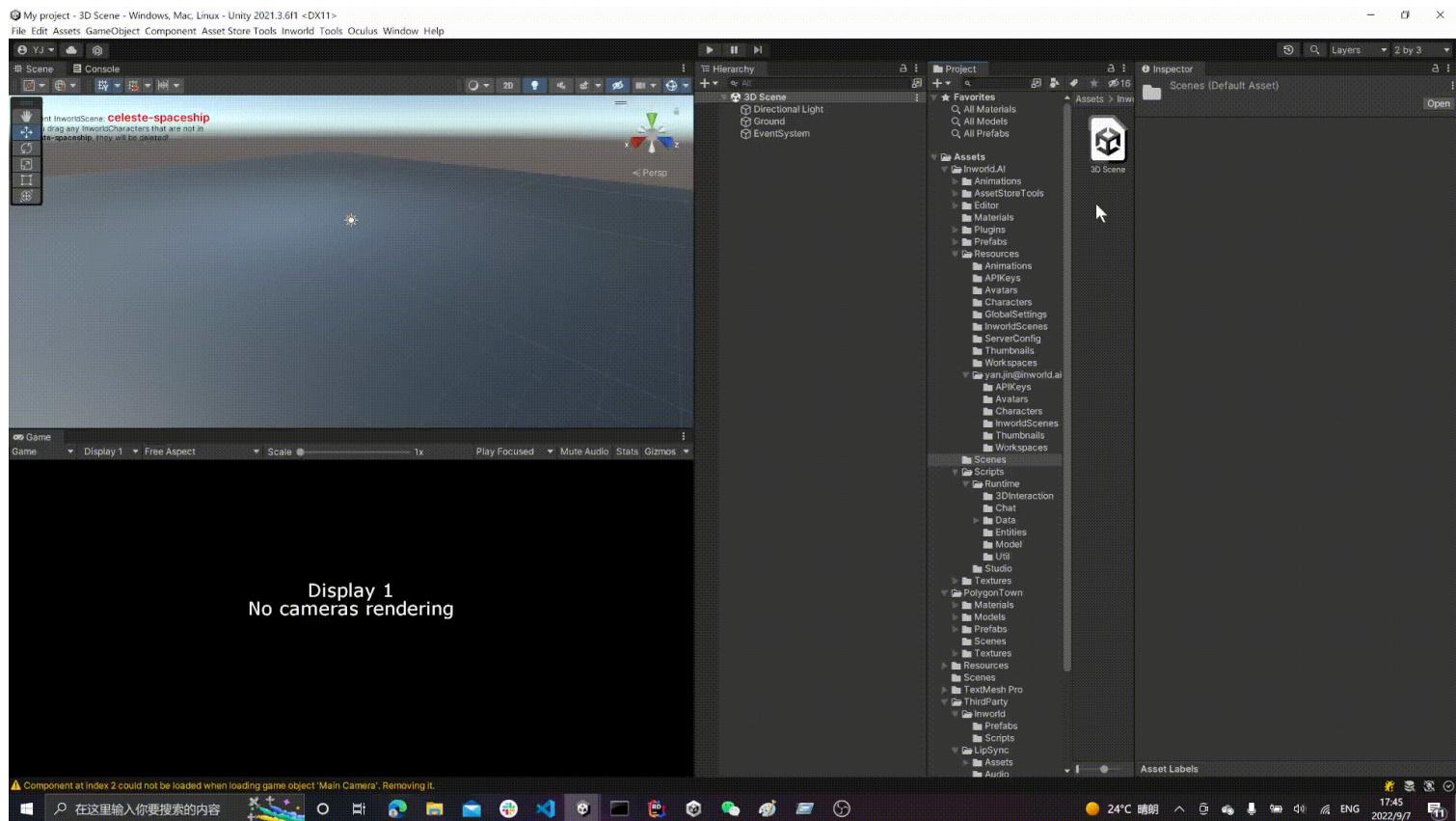
Our Unity Integration package comes with two example **InworldScenes** and three example **InworldCharacters** that you can interact with out of the box (without authentication)

The InworldScene **celeste-spaceship** contains the character **Celeste**, and the InworldScene **olympus-theatre** contains the characters **Zeus** and **Hermes**. In this section, let's set up the environment step by step.

### 1. Opening the Inworld Studio Panel

You can do either of the following to open the **Inworld Studio Panel**.

- At the top menu, click **Inworld** > **Studio Panel**.
- Right click anywhere inside the **Assets** folder, then click **Inworld Studio Panel**.



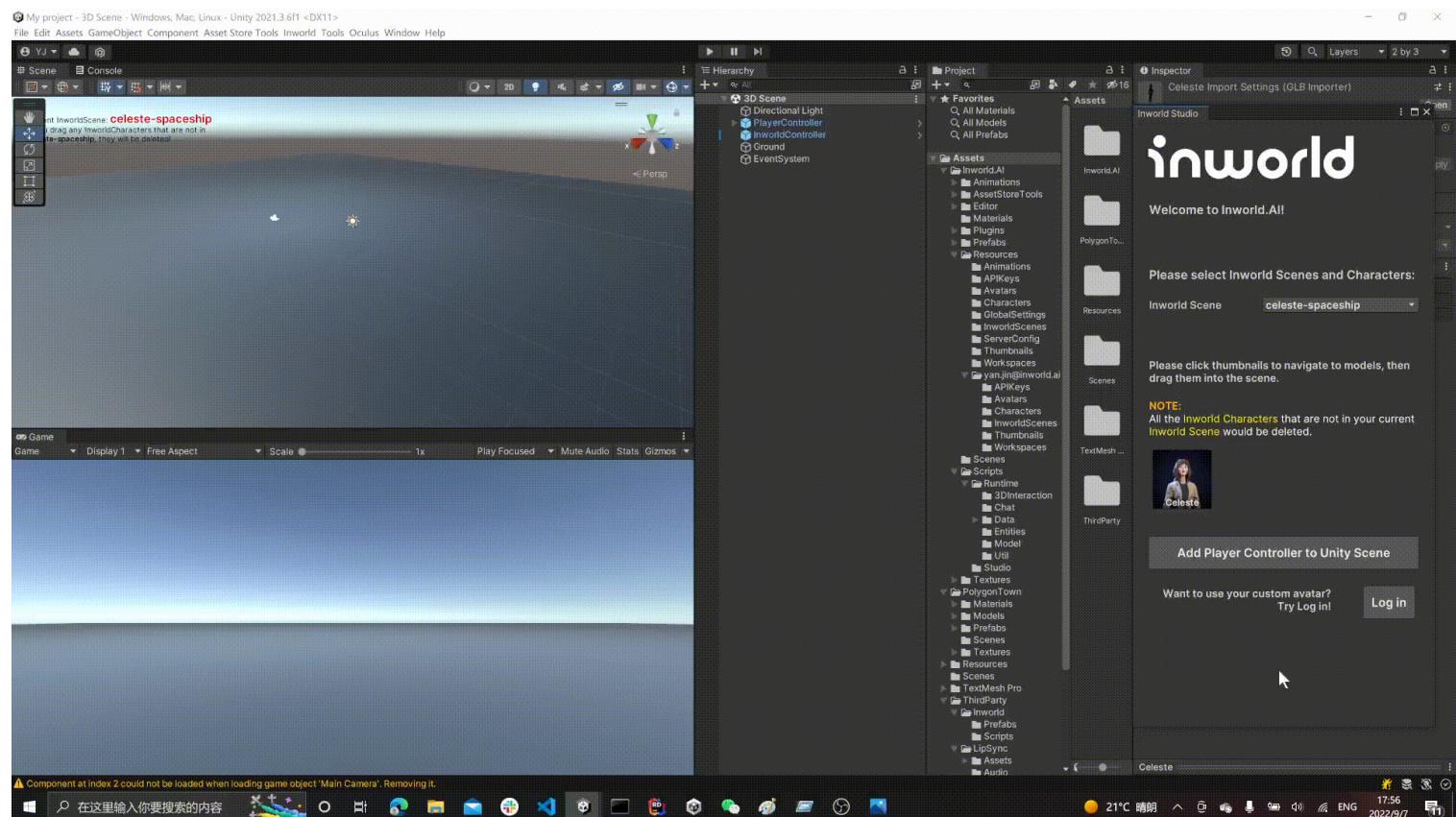
**Note:** The Inworld AI Unity SDK requires **Text Mesh Pro**, which is inside Unity but not imported by default. If it is the first time you are opening the **Inworld Studio Panel**, please click **Import** if the dialog for the **Text Mesh Pro**

## 2. Choosing Your Scene and Character

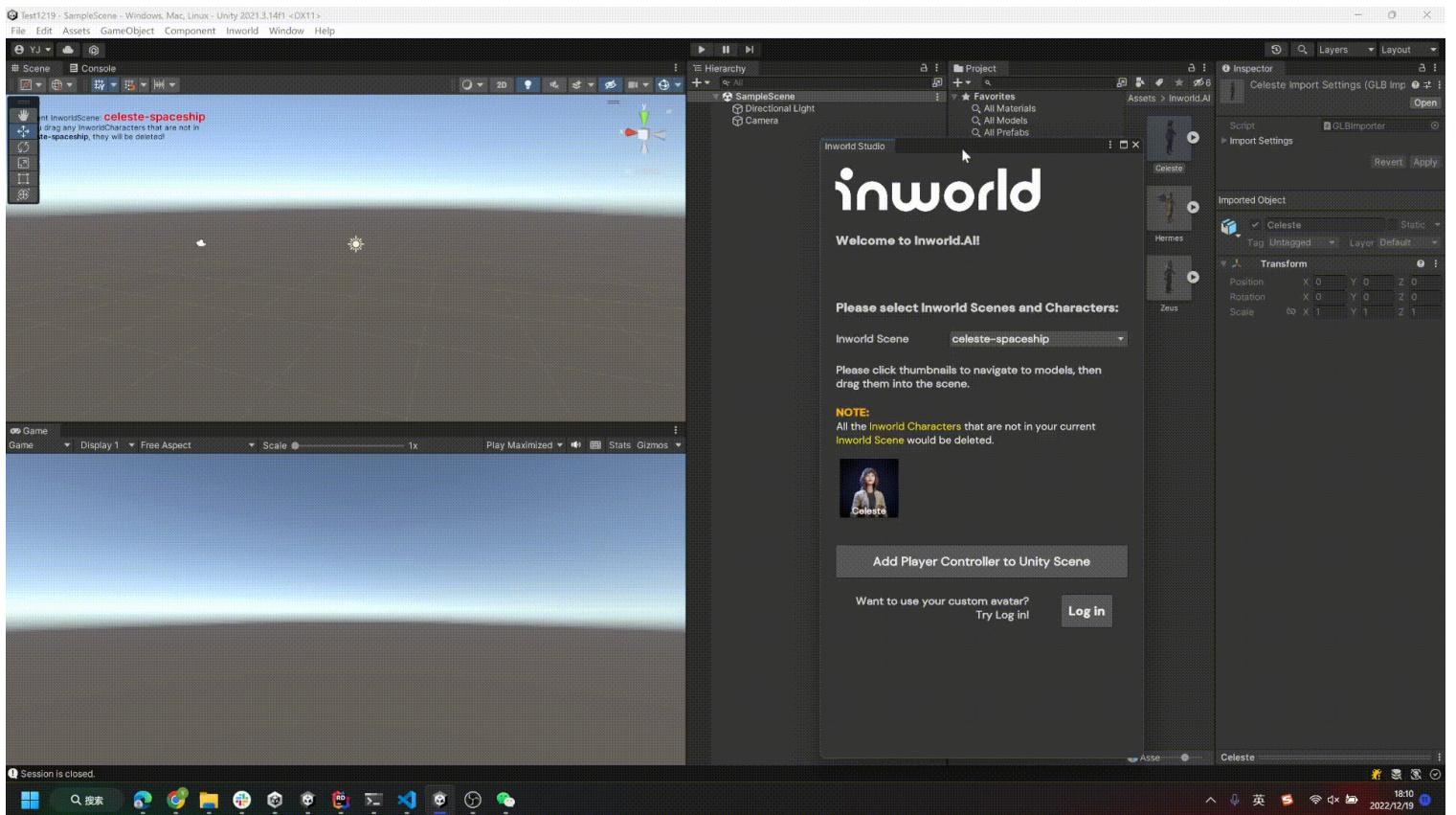
There is a drop down field for you to switch **InworldScenes**. When switching InworldScenes, all characters will also be replaced.

By clicking the thumbnail of the **InworldCharacter**, Unity will navigate to the model you've chosen.

You can drag the model into the Unity Scene. Data and components will be installed to that avatar when you release dragging.

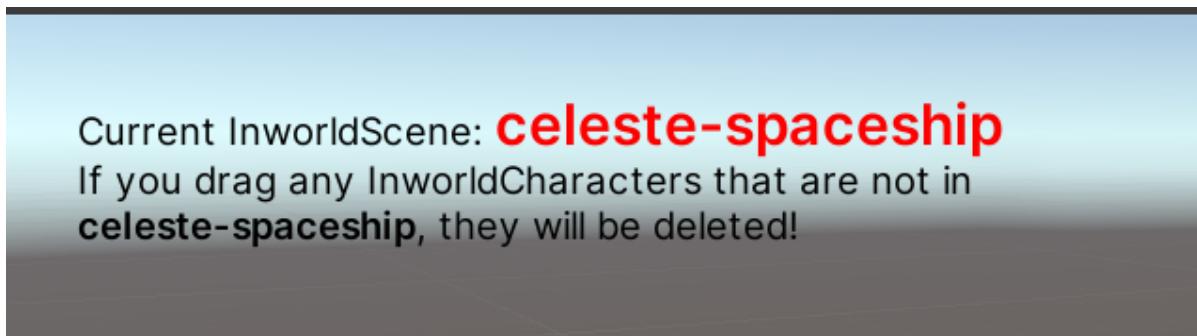


Additionally, after version 2.0.3, you can also drag the model into **Hierarchy Panel**, it'll perform the same action.



**⚠ Note:** Whenever you drag an **InworldCharacter** into a Unity Scene, all the **InworldCharacters** that do not belong to the current **InworldScene** will be deleted! If the **InworldCharacter** that you are dragging does not belong to the current scene, then it will be deleted.

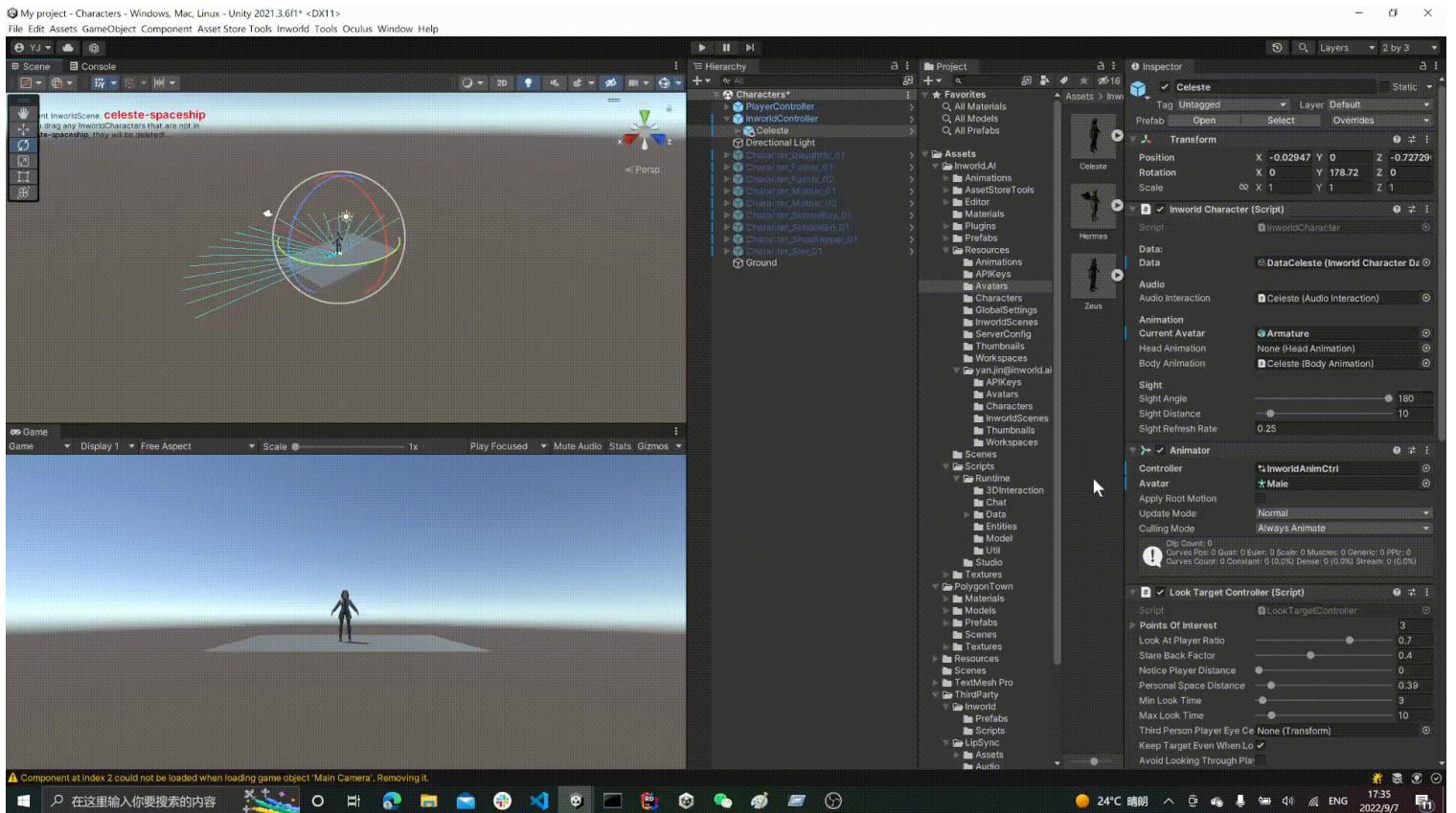
By default, the **Current Scene** will be displayed as a gizmos in **Scene View**.



### 3. Configure Characters in the Scene

Once the **InworldCharacter** `gameObject` is in the scene, you can modify the character. Besides Unity's normal features such as Translating, Rotating, you can also adjust the character's field of view, and/or sight distance.

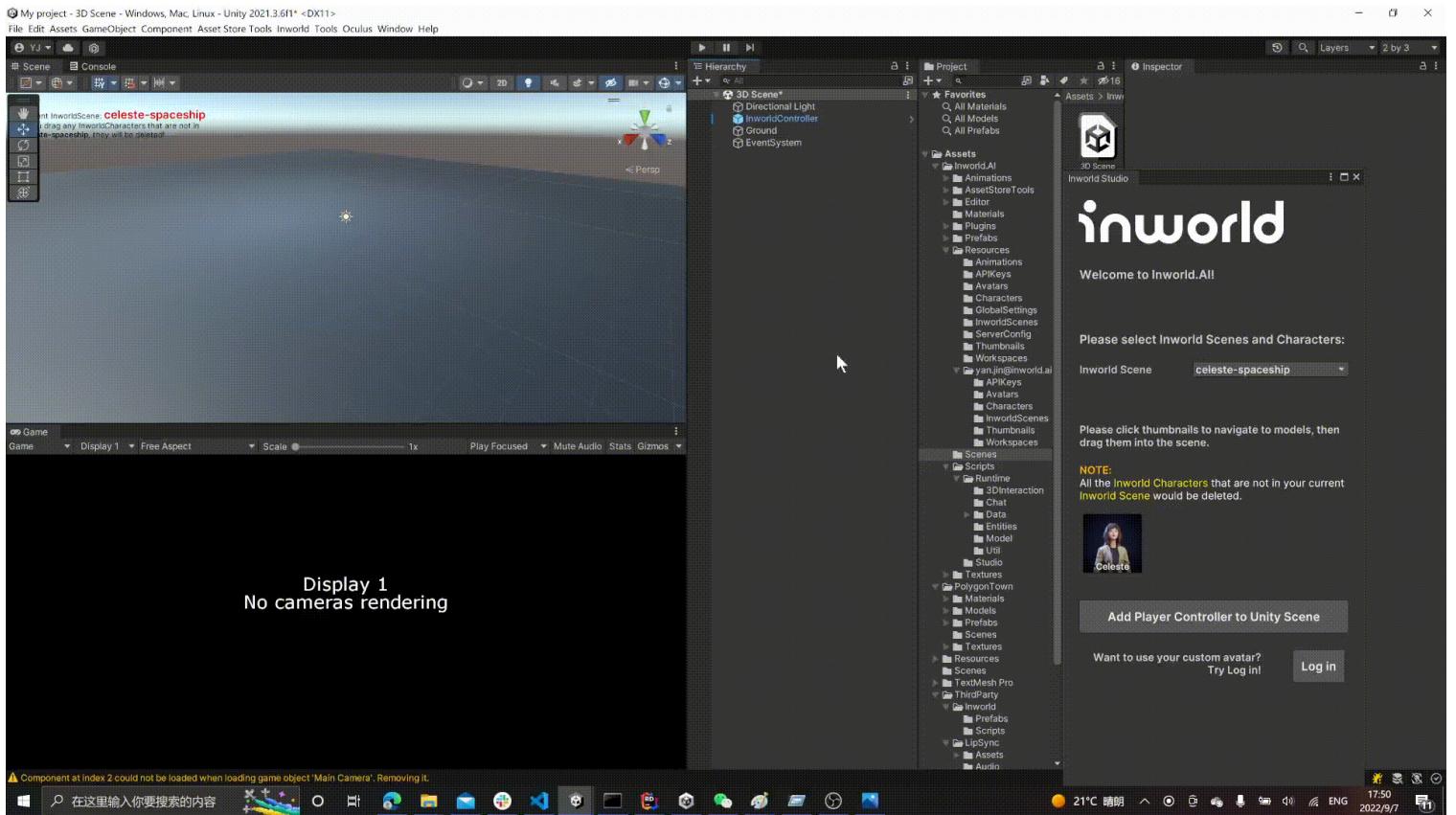
**⚠ Note:** After you drag the avatar into the Unity Scene, you need to click the object in the **Hierarchy** to proceed with the operations above.



## 4. Add a Player Controller

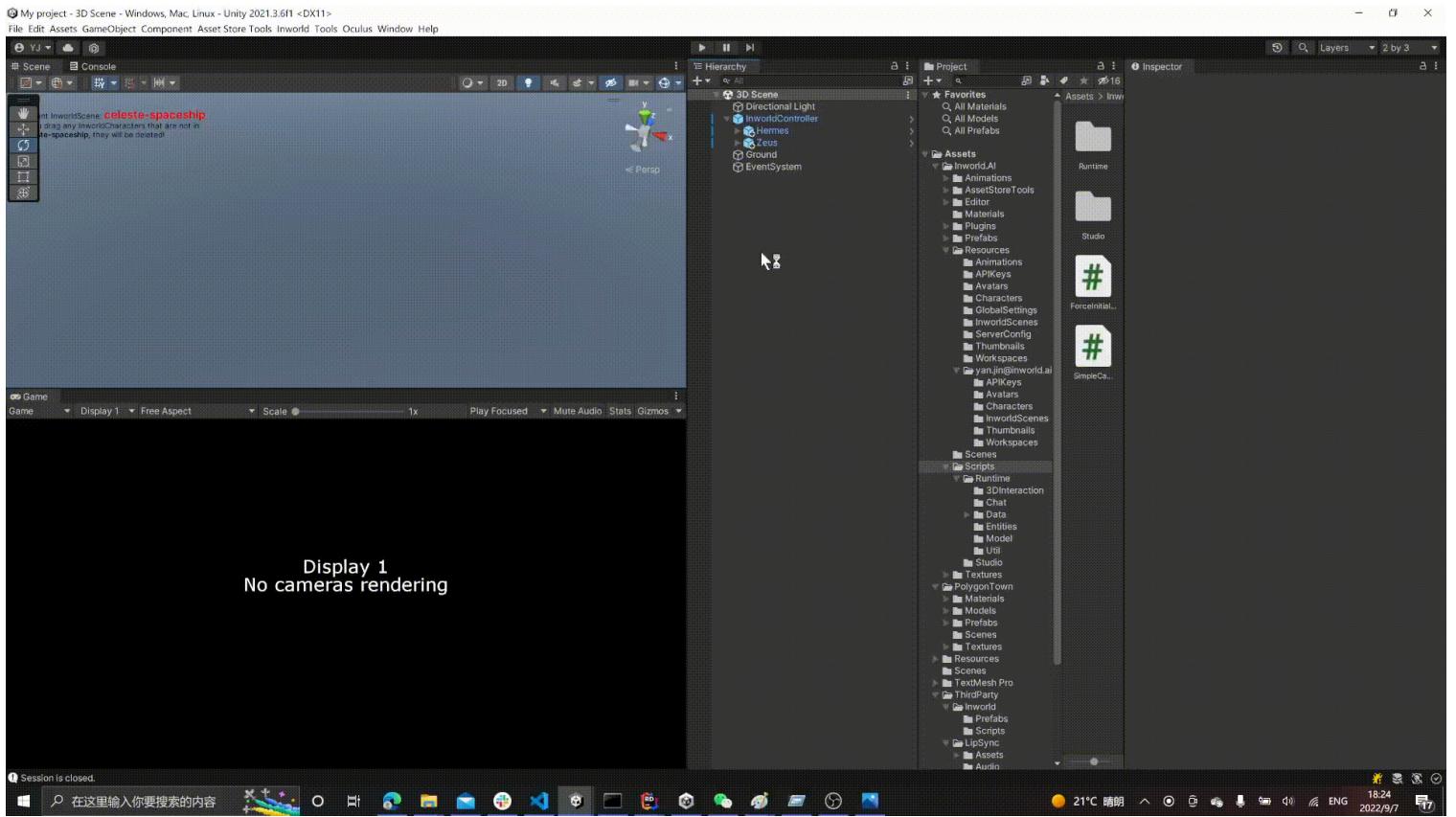
**InworldCharacters** can only be communicated with by **InworldController's** **Inworld Player** **gameObject**. It can be set to any kind of **gameObjects**. If you did not implement your own **Player Controller**, You can simply click **Add Player Controller into Scene** to generate one.

**Note:** The **PlayerController** prefab made by Inworld AI contains a **main camera**. By adding a **PlayerController**, it will delete the current **main camera** in your scene. It also contains an **EventSystem**. Please make sure to keep only one **EventSystem** if you have already created one.

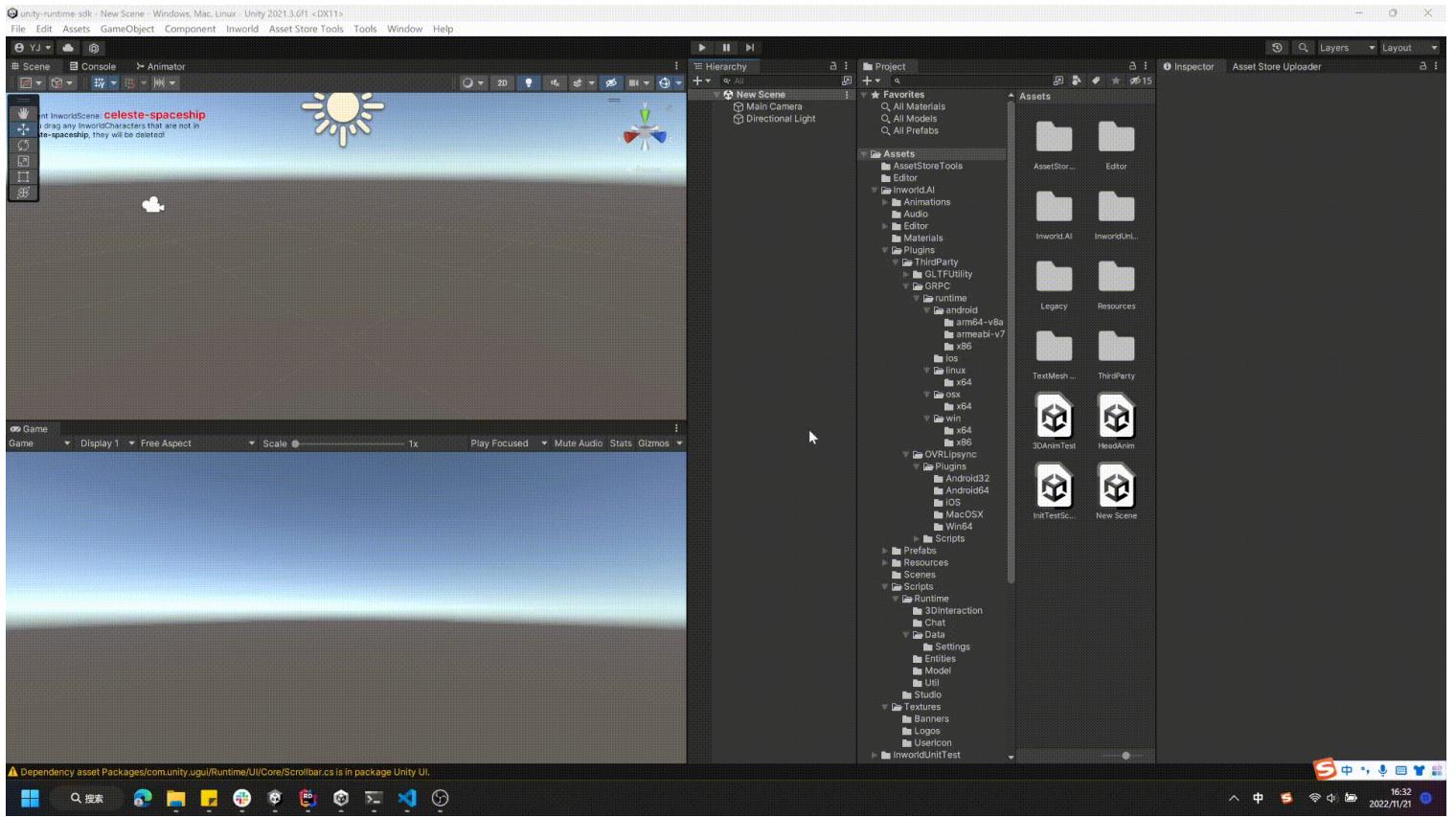


If you have implemented your own logic for player control, then you can drag that `gameObject` into `InworldController's InworldPlayer` field.

An `InworldController` will be generated whenever any `InworldCharacter` is dragged into the scene. It is a `Singleton` `gameObject`, so you should ensure that there is only one copy in your scene.



**⚠ Note:** After Inworle Unity SDK ver. 2.0.3, you can also add **Inworld Player** without opening **Inworld Studio Panel**. To do so, you can simply right click anywhere in the **Hierarchy** view, then click **Inworld > Add > Inworld Player**. It'll automatically add an **Inworld Player** AND an **Inworld Controller** if not existed in the current scene.



## 5. Runtime

Congratulations, you're all set! Now, let's click the Play button to see the behavior:



Please note the following if you are using our default **PlayerController**:

1. You can use the **W**, **A**, **S**, and **D** keys on your keyboard to move around
2. You can use the **Q** and **E** keys on your keyboard to move up and down
3. By single clicking the **Left Mouse Button**, you can look around
4. You can talk to the character that you are looking at using **microphone**
5. You can click the  key to open a text panel. This allows you to type text or hold the record button to send verbal responses

On this page



# create-your-own

## Create Your Experience

In this section, let's setup the environment by the character you created.

### 1. Check your assets.

To integrate your own characters into your Unity project, you will need to ensure that you can access:

1. At least one **Workspace**.
2. At least one **API Key** in your workspace.
3. At least one **Character** in your workspace.
4. At least one **Inworld Scene** in your workspace.
5. Ensure that your **Inworld Scene** contains all the Characters you need to interact with.

If you do not know how to create them, please check our [Prerequisites page](#).

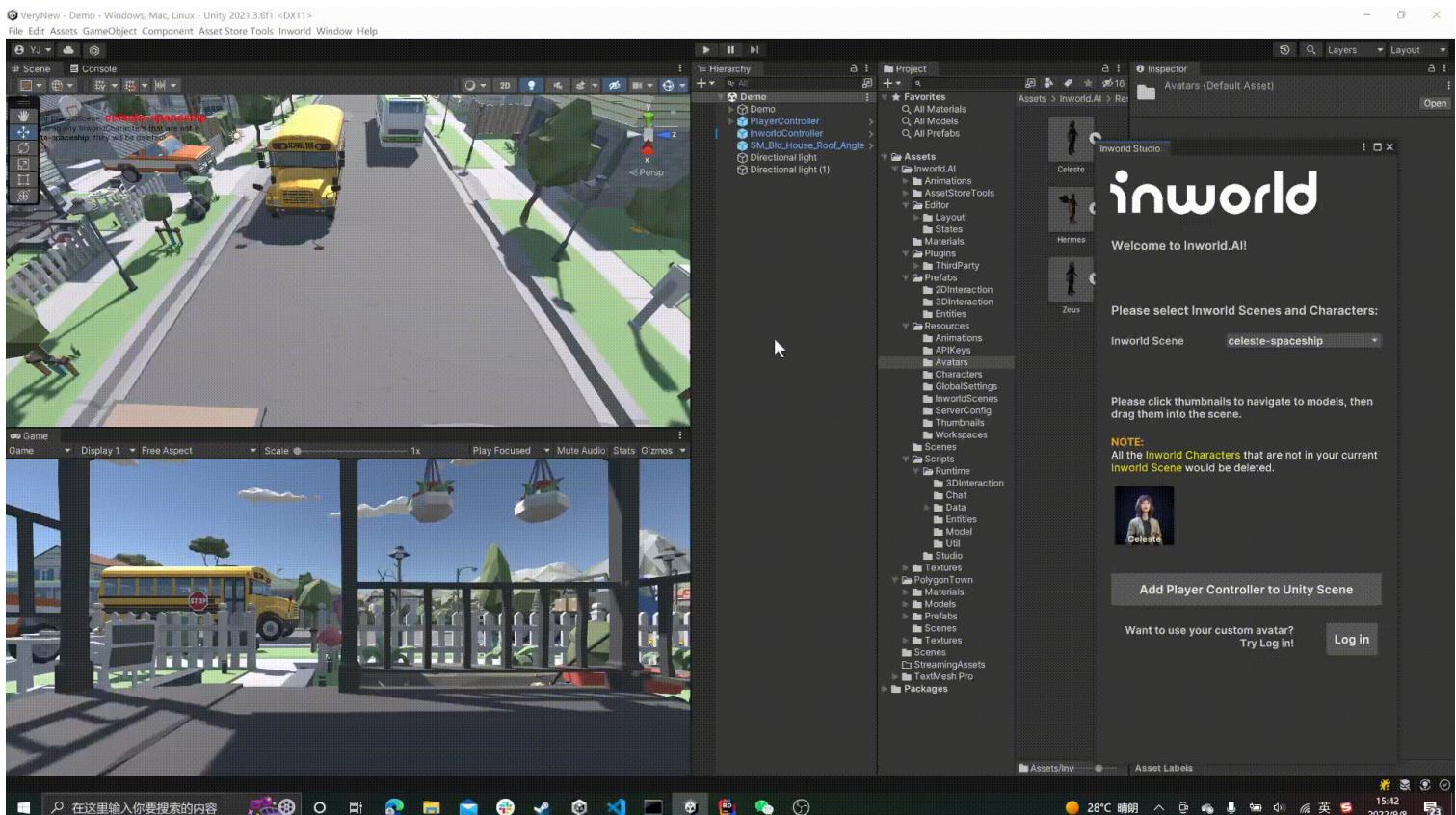
For more information, please check [Studio Basics Tutorial Series](#).

### 2. Open Inworld Studio Panel in Unity.

If you do not know to open **Inworld Studio Panel**, please check the [Integrate to your Scene](#) page.

### 3. Login

1. Click [Login](#).
2. At <https://studio.inworld.ai>, click [Integration](#).
3. At **Studio Access Token**, click [Generate Studio Access Token](#).
4. Click the [Copy](#) button to copy it.
5. Back to Unity, paste the token you copied to **Inworld Studio Panel**.
6. Click [Login](#).



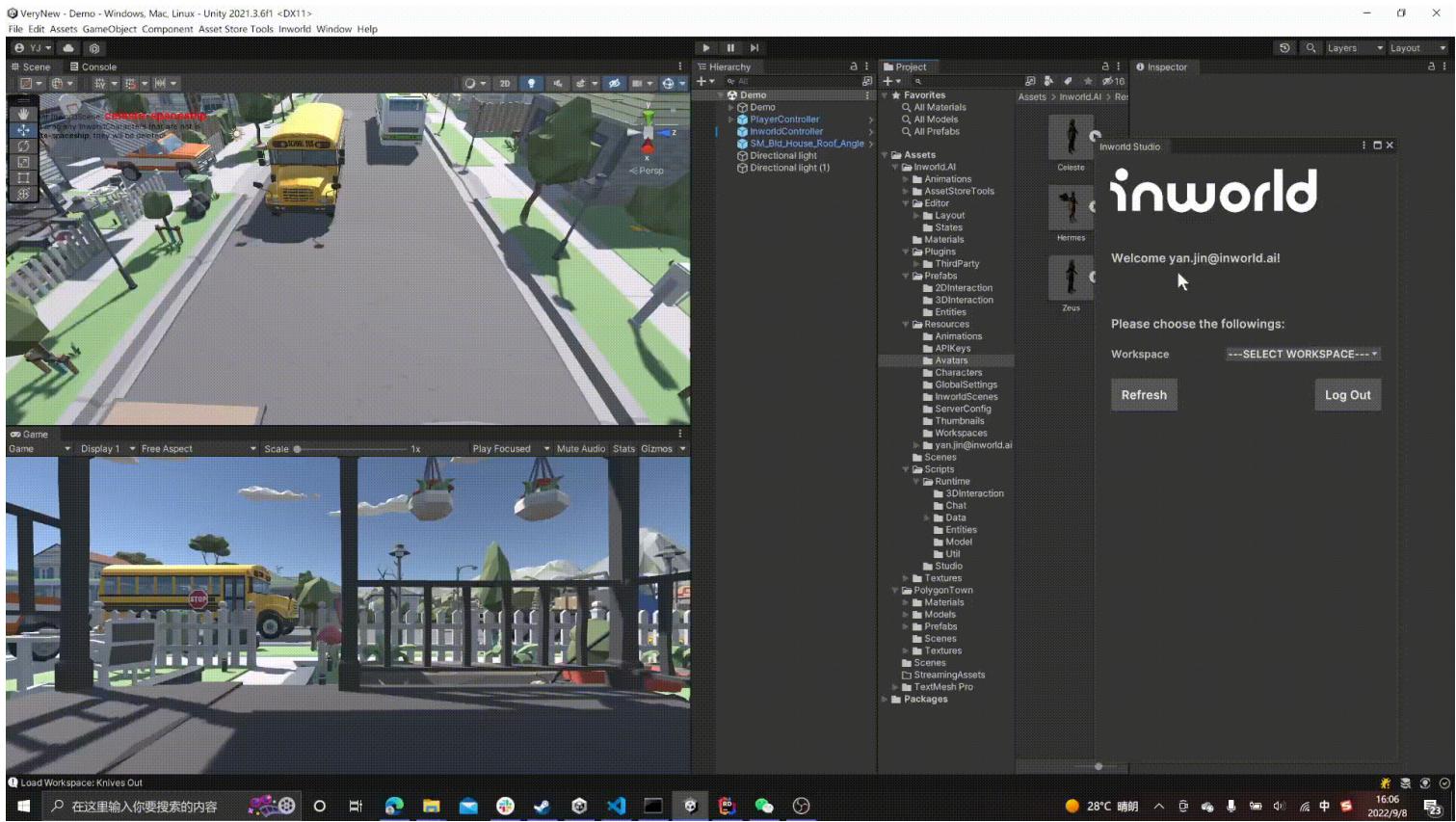
### **⚠ Note:**

- The token copied this way has an expiration time for 1 hour.
- Once user logged in, This SDK will try reconnect if expired.
- If sometimes the reconnection is not working, try clicking the **Refresh** button located at the bottom of the **Inworld Studio Panel**.
- If the problem persists, please check your network connection and copy-paste the login token again.

## 4. (Optional) Change your user name.

After connecting, the **Inworld AI Unity SDK** will fetch your Unity **UserName** (most likely your email address) as the default name used in the SDK.

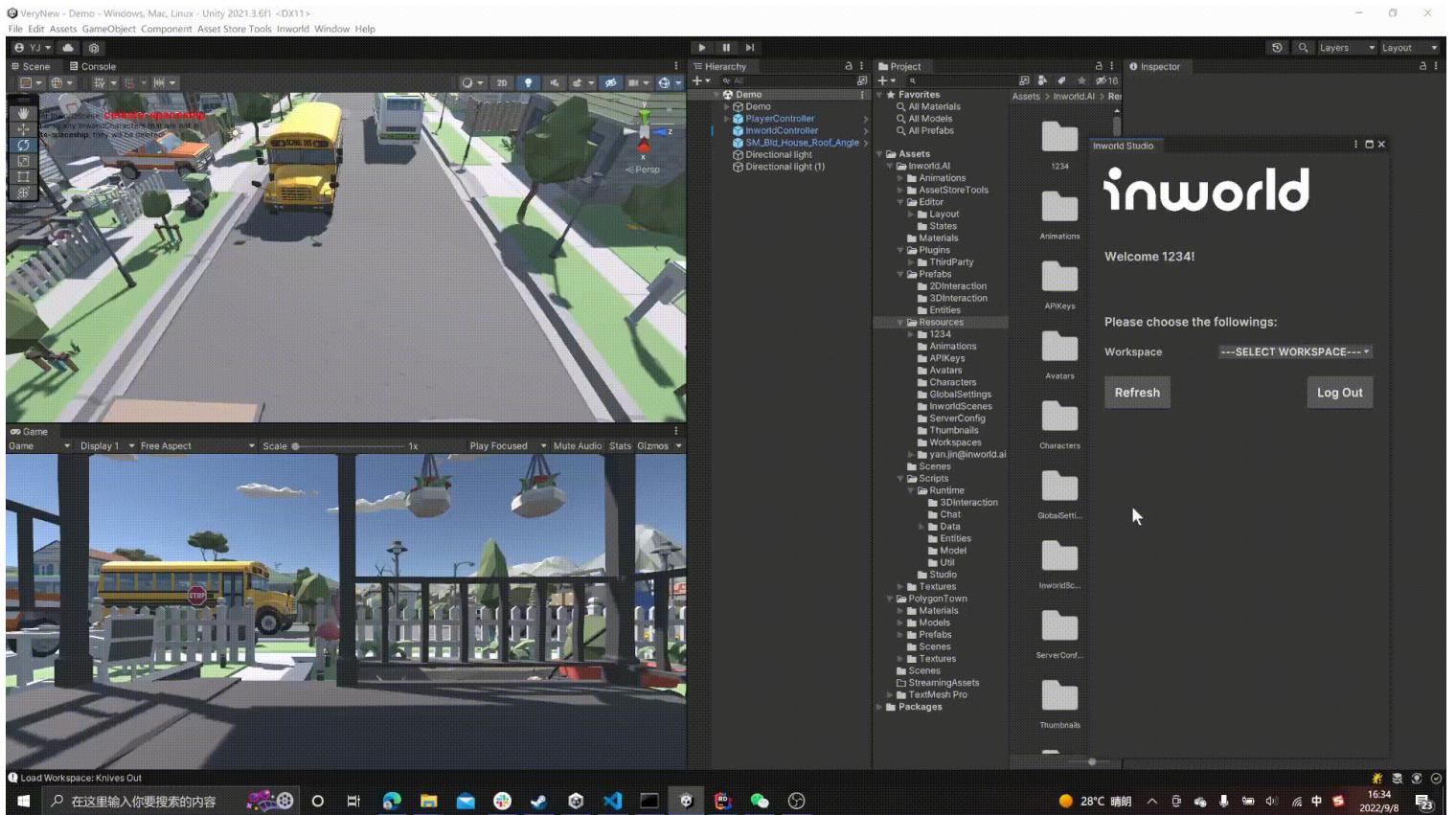
You can change this name at **Inworld Setting Panel > User Settings** or **Edit > Project Settings > Inworld.AI**.



**⚠ Note:** Inworld Setting Panel (ScriptableObject of InworldAI) is displayed in Inspector, and may sometimes be covered by other panel. You can manually click Inspector to bring the panel to the foreground

## 5. Import your own Character into a Unity Scene.

1. Once you're connected to Inworld Studio Server, you can choose your **workspaces**. The panel will download the data (key/scene/character references, etc) of that **workspace**.
2. Once your **workspace** has been downloaded, you can select the relevant **API Keys** and **Inworld Scenes** to download the data (character thumbnails/models, etc) of that Scene.
3. After that **Inworld Scene** have been downloaded, you can click on the thumbnail to navigate to the model and drag it into the Unity Scene.



### ⚠ Note:

- All the asset downloaded would be saved at `Inworld.AI/Resources/{YOUR_USER_NAME}/`. So if you changed your name, data would be downloaded to a new folder.
- All the **InworldCharacters** that are not belonged to the current **InworldScene** would be deleted. For more information, please check [getting-started page](#).

## 6. Next steps.

Once you've loaded the **InworldCharacter** into the scene, you can continue from step 3 of the [Integrate to your Scene](#) page.

On this page



# Prerequisites

## Prerequisites

This section will cover how to set up the demo scene. Please review the prerequisites that follow.

## Studio Requirements

To integrate **Workspaces**, **Characters** and **Scenes** into your Unity project, you will need to access:

### 1. At least one workspace.

You can determine your current workspace here:

The screenshot shows the 'inworld' platform interface. On the left, a sidebar lists 'Workspace' (Inworld Sandbox), 'Characters', 'Scenes', 'Common Knowledge', and 'Integrations'. Below this are links for 'Discord', 'Documentation', and 'Contact support', along with 'Terms of Use | Privacy policy'. The main area is titled 'Characters' and displays four character cards: 'Morgana' (a woman with white wings and a white and gold outfit), 'Zeus' (a man in a blue and brown outfit), 'Hermes' (a man with white wings and a white and gold outfit), and a fourth character partially visible below them. A purple button at the top right says '+ Create new character'. The top right corner of the screen has a circular profile picture with the letter 'J'.

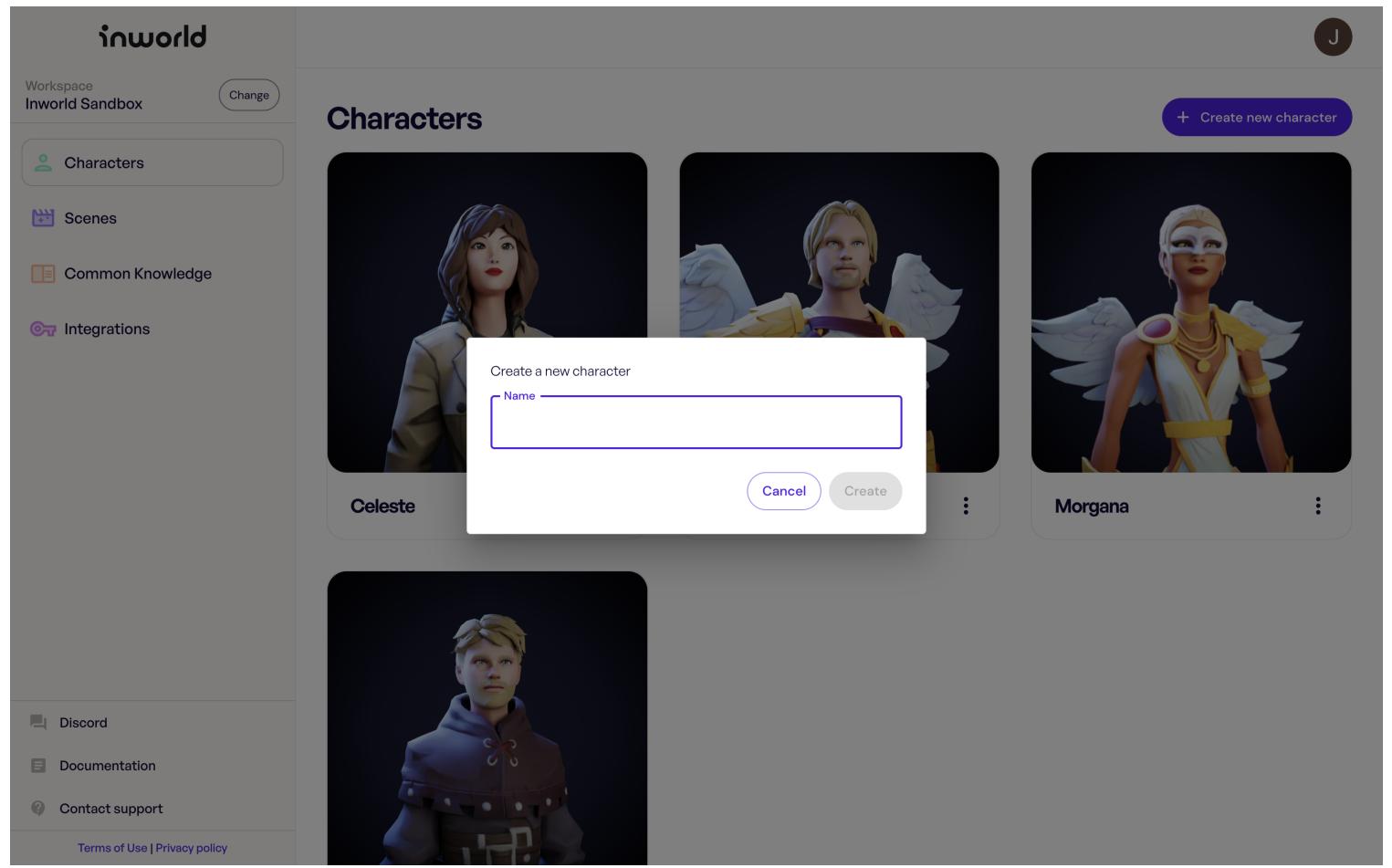
## 2. At least one API Key in your workspace.

You can create an API key via [Integrations > API Keys > Generate new key](#).

The screenshot shows the inworld workspace interface. On the left, there's a sidebar with links for Characters, Scenes, Common Knowledge, Integrations (which is selected), Discord, Documentation, and Contact support. The main area has a header with the inworld logo and a "Change" button. Below the header, it says "For inquiries about custom projects or other questions, please reach out to us at [support@inworld.ai](mailto:support@inworld.ai)". There are three cards for Unity, Unreal Engine, and Node.js. The "API Keys" section is highlighted with a purple border. It contains a table with columns "Key" and "Secret". Under "Key", there's a copy icon and a trash icon. Under "Secret", there's a copy icon, a "Suspend" button, and a trash icon. A blue "+ Generate new key" button is at the top right of this section. Below the API Keys section, there are sections for "Unity Studio" (with a "Generate unity studio login token" button) and "Oculus Integration" (with "Link Oculus account" and "Request access" buttons).

## 3. At least one Character in your workspace.

You can create a character here:



#### 4. At least one scene in your workspace.

You can create a scene here:

The screenshot shows the inworld platform interface. On the left, there's a sidebar with options like Workspace (Inworld Sandbox), Characters, Scenes (which is selected and highlighted in blue), Common Knowledge, Integrations, Discord, Documentation, and Contact support. At the bottom of the sidebar are links for Terms of Use and Privacy policy. The main area is titled "Scenes" and shows a placeholder image of a character in a speech bubble. Below it says "No scenes". There's a blue button at the bottom right labeled "+ Create new scene". In the top right corner, there's a dark circular icon with a white letter "J".

## 5. Ensure that the scene you have created contains at least one character.

You can check this here:

The screenshot shows a list of characters in a scene. It includes a thumbnail of a character named "Zeus", a count of "1", and a "Characters in scene" button with an info icon. Below the character is a dashed blue box with a plus sign, indicating where to add more characters. At the bottom, there are three dots and a small "Zeus" name.

If you want to know more about how to create your **workspace**, **character** and **scene**, then you can check out our **Studio Basics Tutorial Series**.

On this page



# own-character

## Use your own avatar

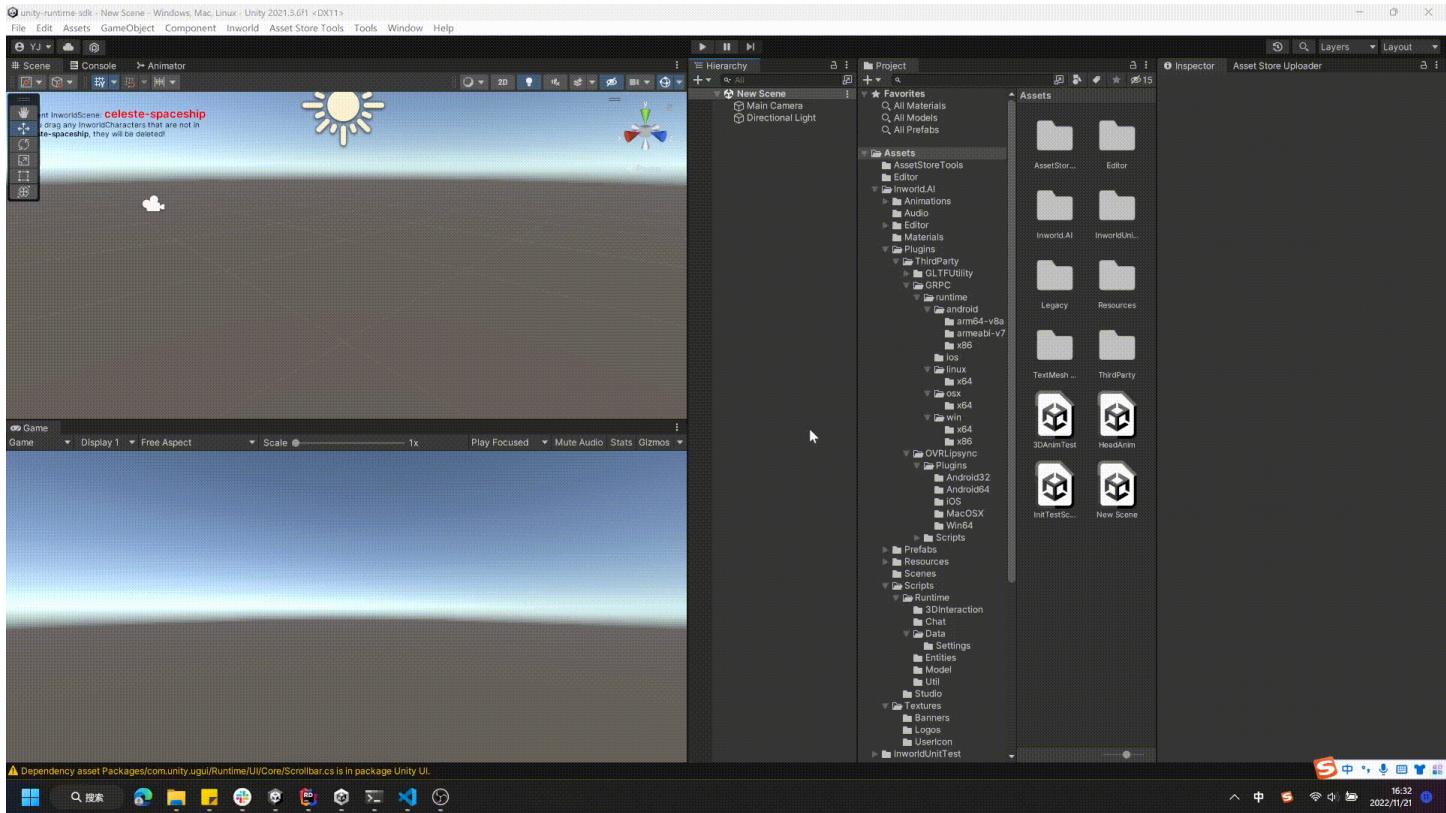
In Inworld Unity SDK 2.0.3 or later, we made it much easier to support your own avatars integrating with **InworldCharacter**. Let's take a look.

## 1. Quickly establish your scenes

Once you've already finished the settings for the workspace, scene, key, and character in the **Inworld Studio Panel**, you can quickly create any scene without open it any more.

You can right click anywhere at the **Hierarchy** view, click **Inworld > Add > Inworld Player**, we'll automatically add an **Inworld Player** AND an **Inworld Controller** for you if they are not existed in the current Unity scene.

**⚠ Note:** The **InworldController** would be automatically created if not exist, and will contain the current **InworldScene** you've set in **Inworld Studio Panel**.

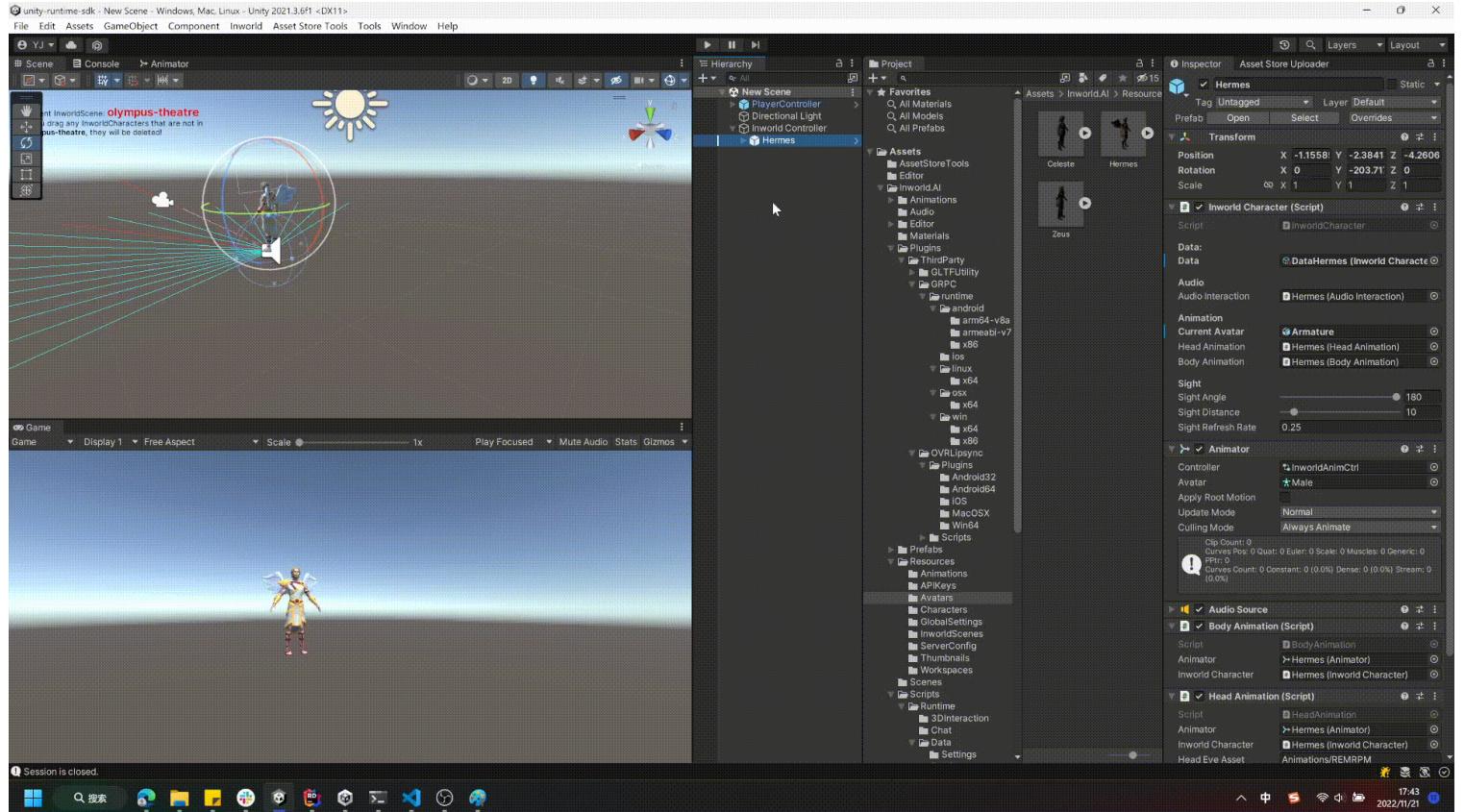


## 2. Quickly change your InworldCharacter

As long as the **InworldCharacter** is in the same **InworldScene**, you can just drag your related **InworldCharacterData** to replace the **Data** in **InworldCharacter** script, without opening **Inworld Studio Panel**.

For example, this following gif showcases how to let **Hermes**'s body have **Zeus**'s soul.

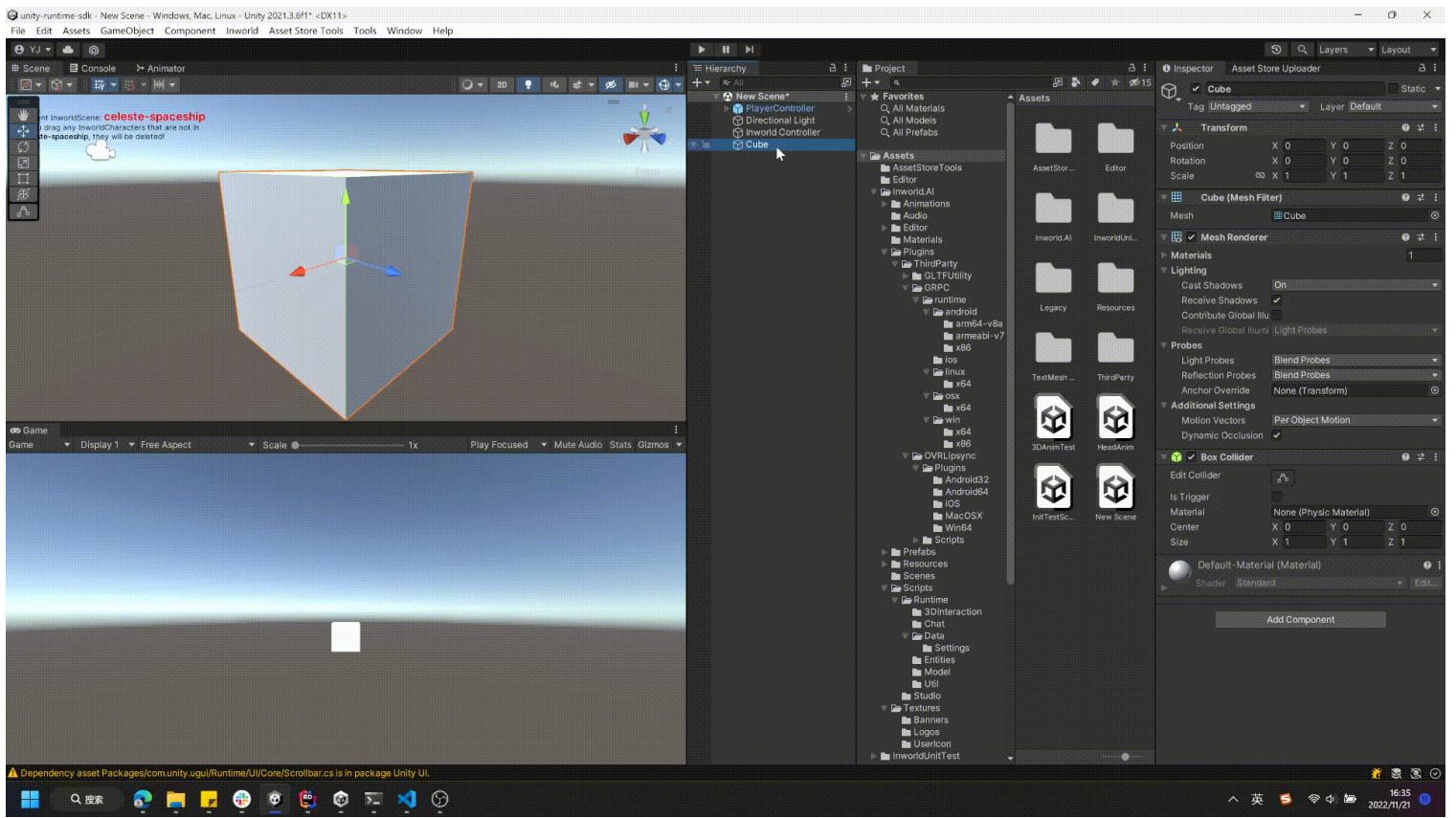
**⚠ NOTE:** If your **InworldCharacterData** is not belonging to the same **InworldScene**, it won't work!



## 3. Convert any object to InworldCharacter

If you want to use your own avatar, with your own implementation of animations, you could easily make it by right clicking it, click **Inworld > Add > Inworld Player**, we'll automatically add a set of **Inworld Character** scripts with the default character (most likely the first one) in your current **InworldScene**.

Take this following Cube for example.

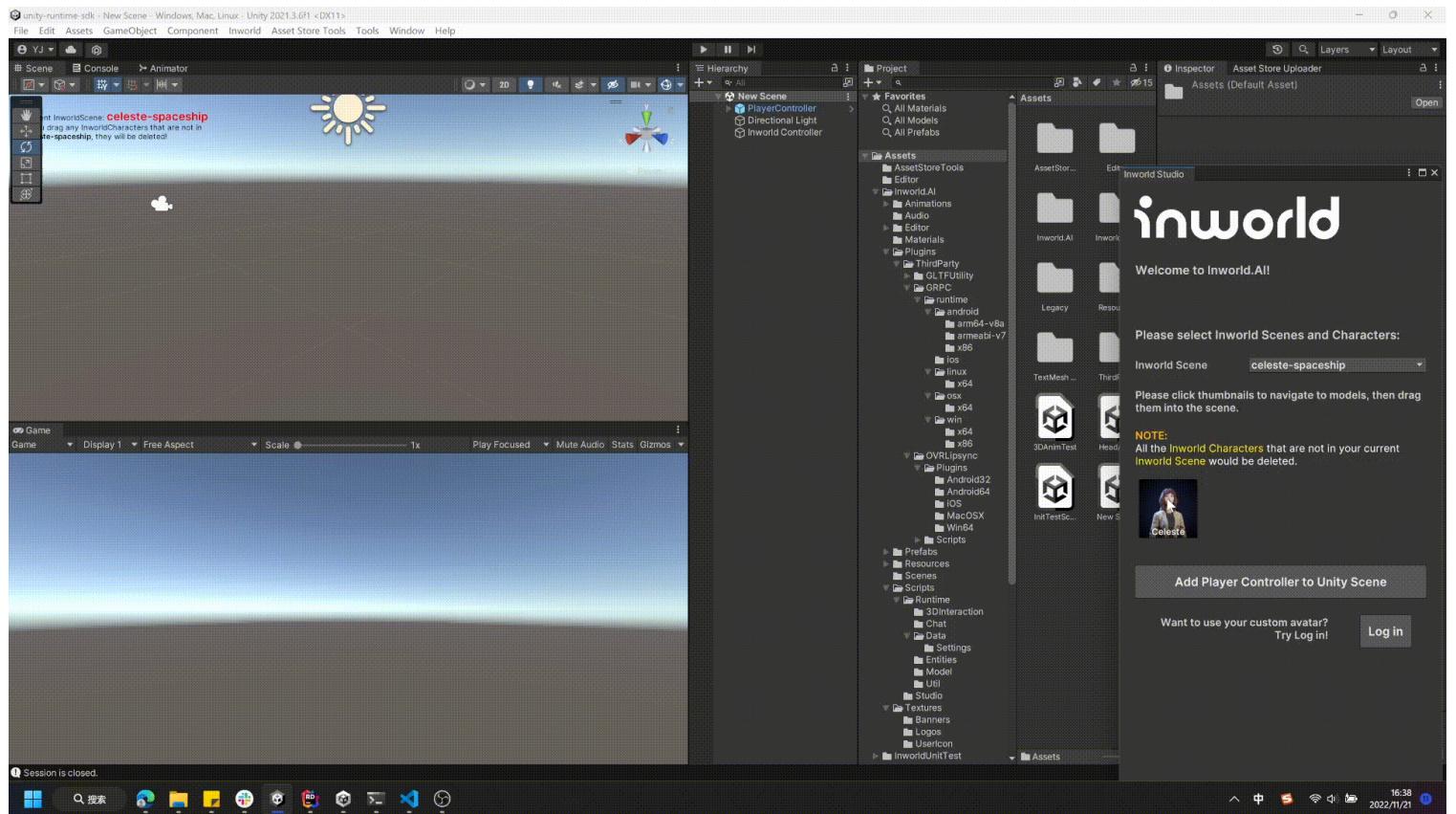


## 4. Use your own GLB humanoid avatar

If you want to use your own animation, the step above is enough. If you have a glb humanoid avatar, you can let it integrate with Inworld Animations and lip syncing.

### 1. Create an existing Inworld Avatar.

If you don't know how to do it, you could check out [this page](#).



## 2. Set position.

Drag your own avatar right below the Inworld Avatar you created.

## 3. Replace Armature.

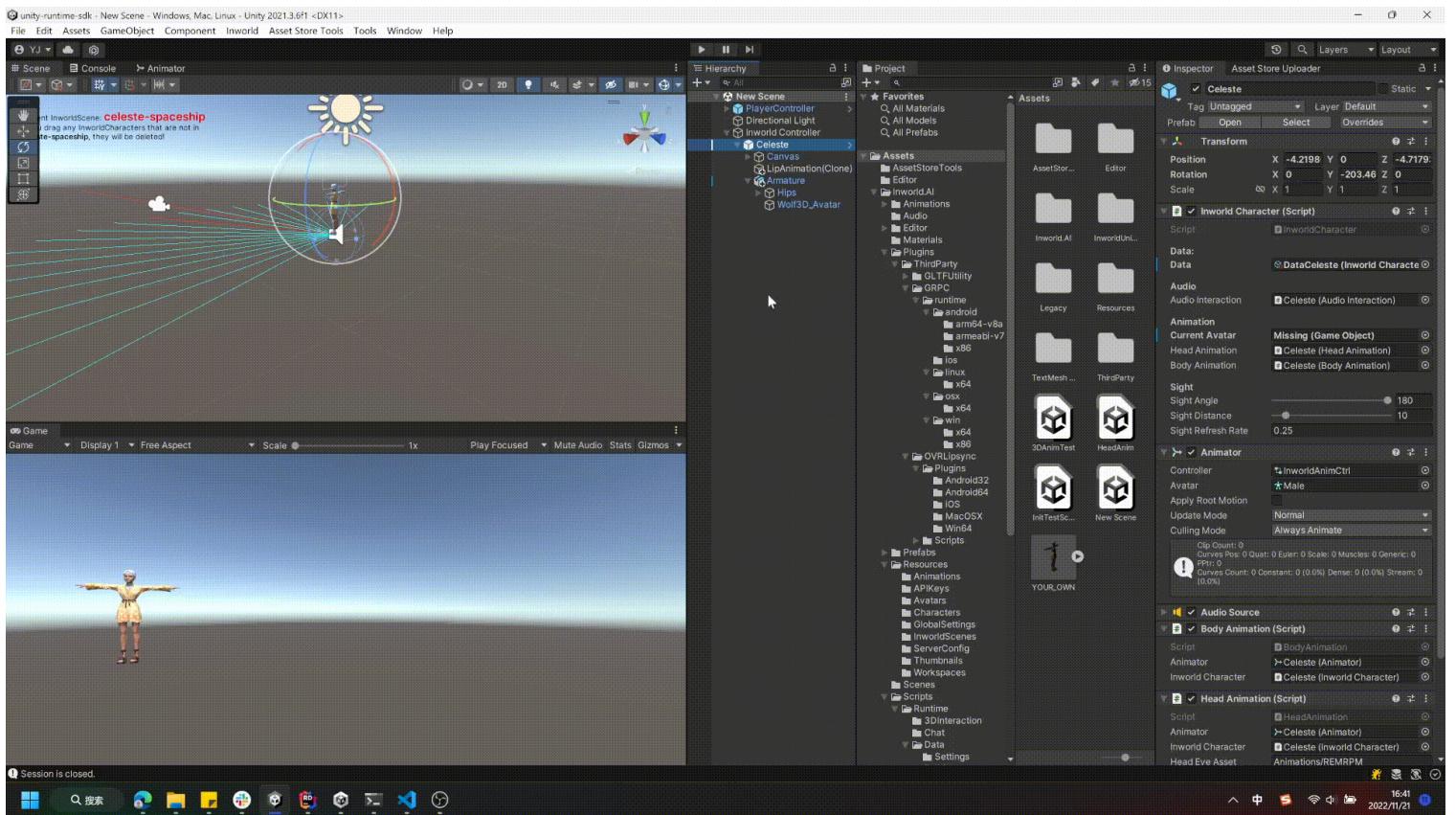
Rename your avatar to `Armature`, and delete the existing `Armature`.

## 4. Set Current Avatar.

Under `InworldCharacter` script's `Animation > Current Avatar`, drag your new `Armature` onto it.

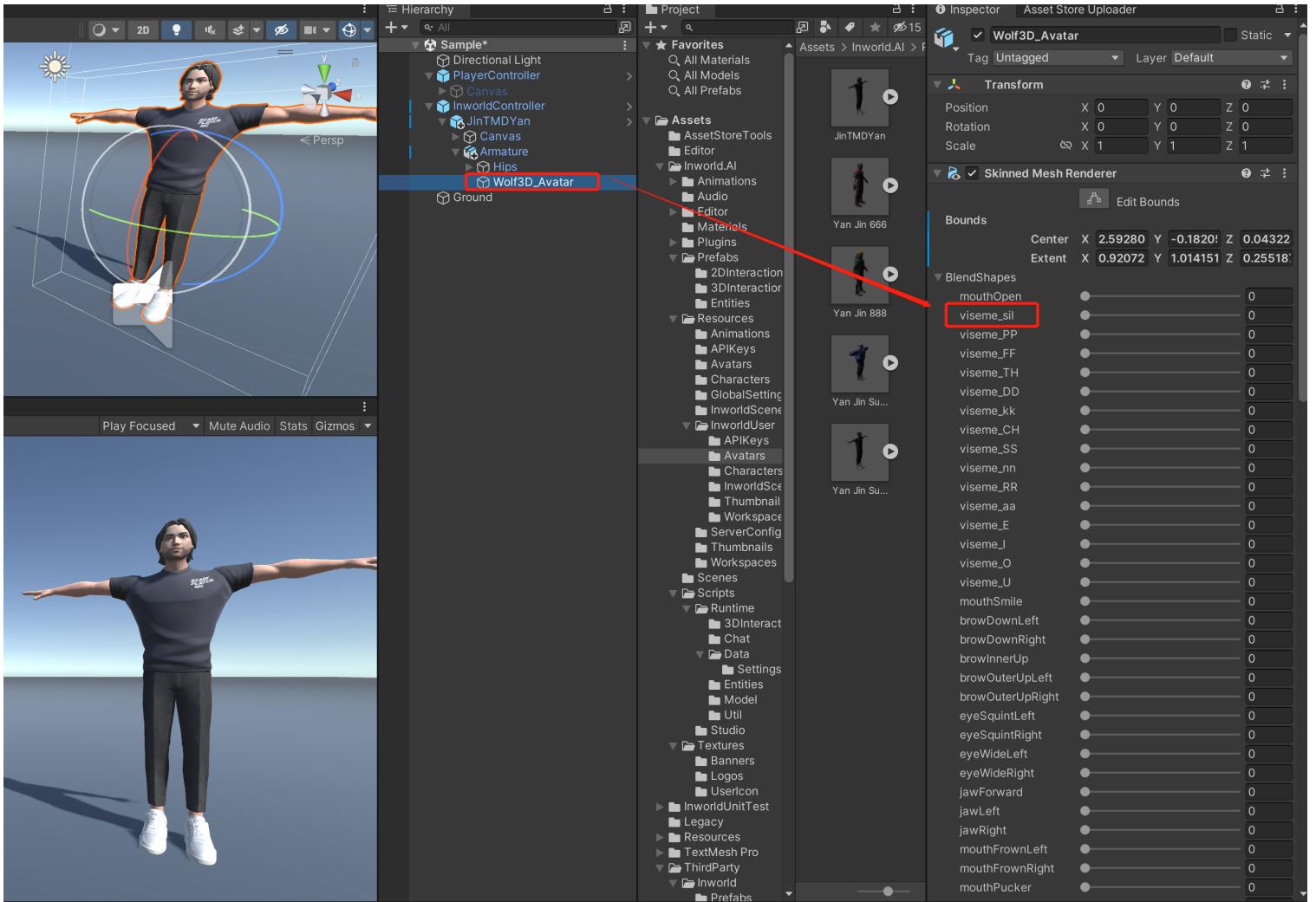
## 5. Done

Now your avatar has already integrated with Inworld animations!



## 6. (OPTIONAL) Lipsyncing.

If your avatar contains **SkinnedMeshRenderer** (**Blend Shape** in 3D modeling software) with viseme index, and the order of viseme indices is continuous and correct (From Sil to U), Lipsyncing would automatically be implemented. Please visit [Lip Sync](#) for more details.



[tutorial-integrations](#)[Unity](#)[create-your-own](#)[save-your-scene](#)

On this page



# save-your-scene

## Save your scene

If you're processing multiple **InworldScenes**, switching **InworldScenes** will sometimes make the `InworldAI.Game.currentScene` mismatch.

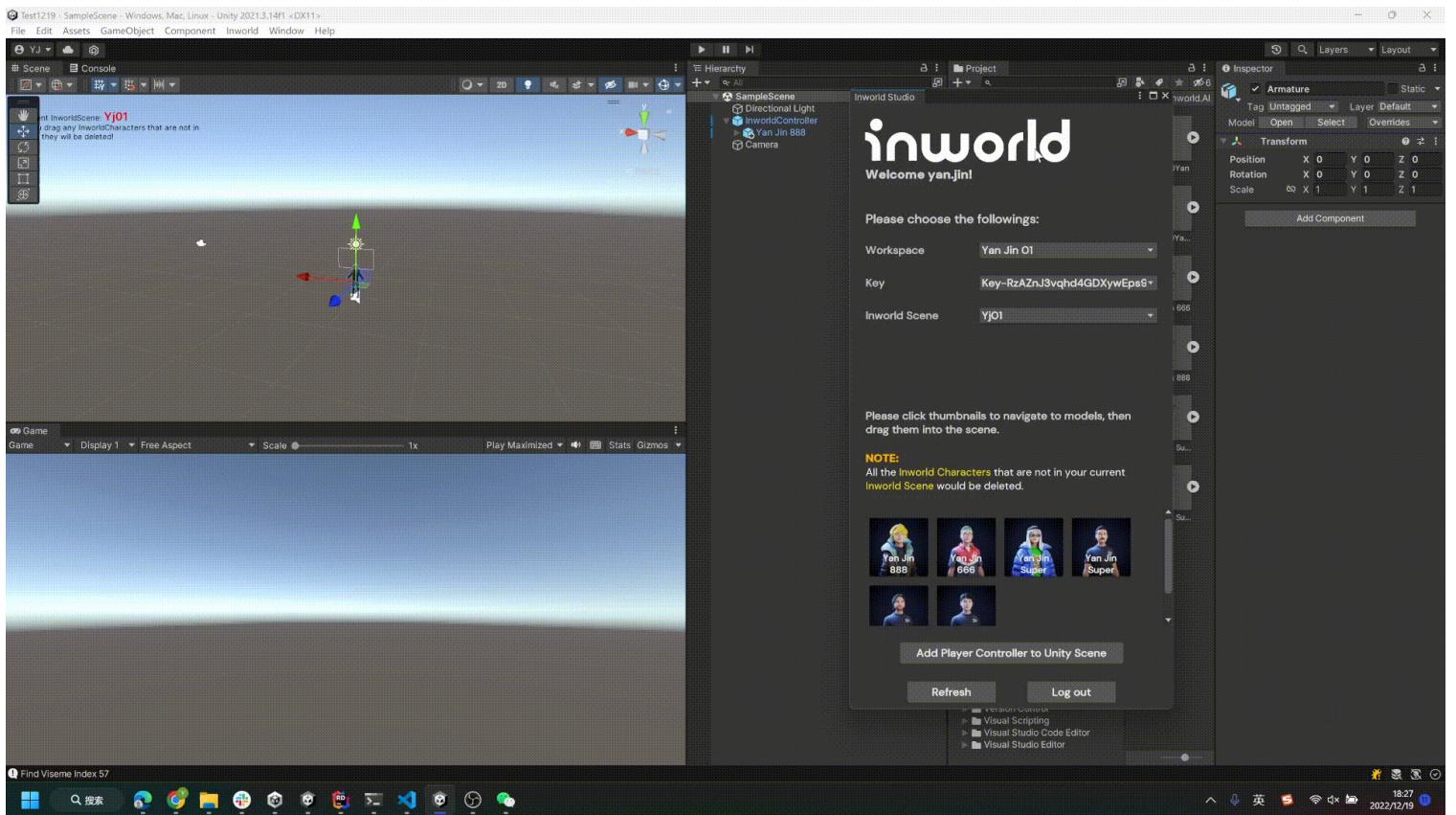
In Inworld Unity SDK 2.1.0 or later, we add a feature to save your data locally, binding with your Unity scene. Let's take a look.

### 1. Search scripts `InitInworld`

Attach this script to any object in your current Scene.

### 2. Set Variables

Set your `InworldWorkspaceData`, `InworldSceneData`, `InworldCharacterData`, and `InworldKeySecret` to the `InitInworld` script.



Done!

Now whenever you switch other scenes, as long as you open this Unity scene, these data would immediately loaded.

On this page



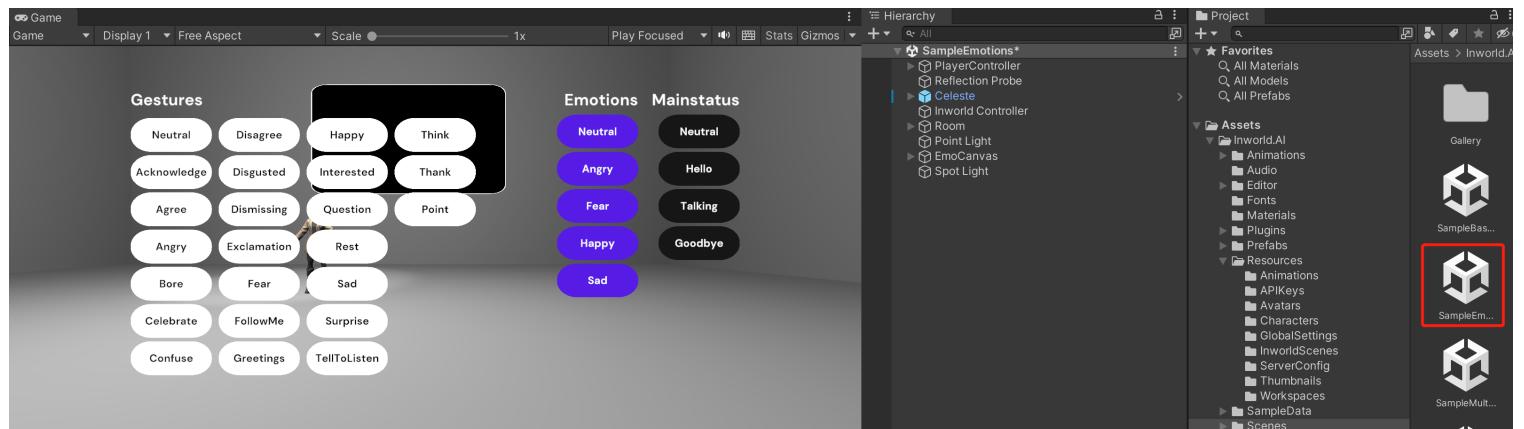
# animations

## Animations

Animations in our Unity SDK are triggered via `GestureEvents` and/or `EmotionEvents` that we receive from the server. This package contains several animations, but they are not covering all the `GestureEvents` and `EmotionEvents`. It is recommended that you import your own animation clips, and set them as states of our **Animator**.

## Demo

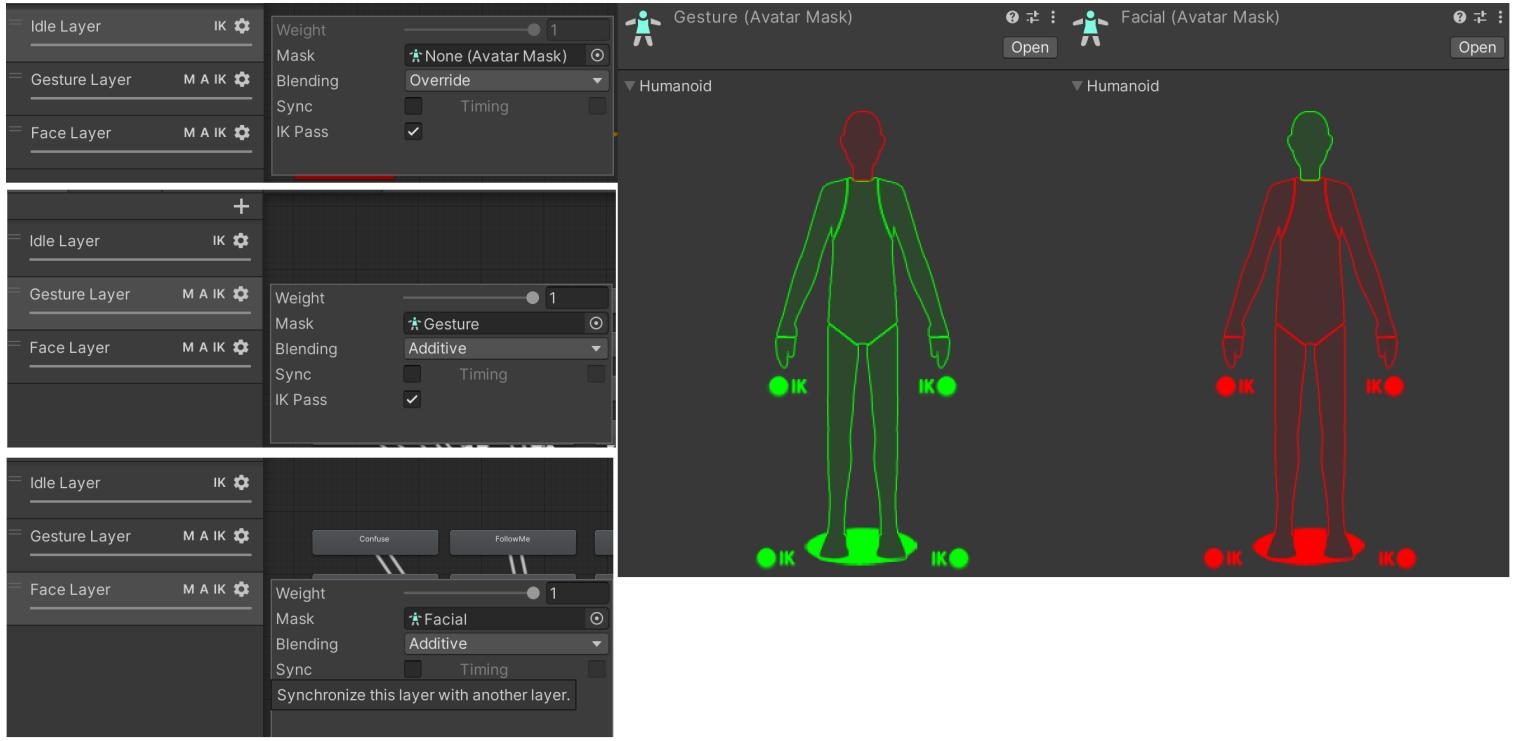
Before you go through this page, we highly recommend you check the **Emotion Sample Demo Scene**



## Architecture

Our animator is called `InworldAnimCtrl`. It contains three different layers. These are the **Idle Layer**, the **Gesture Layer**, and the **Face Layer**. Please note the following,

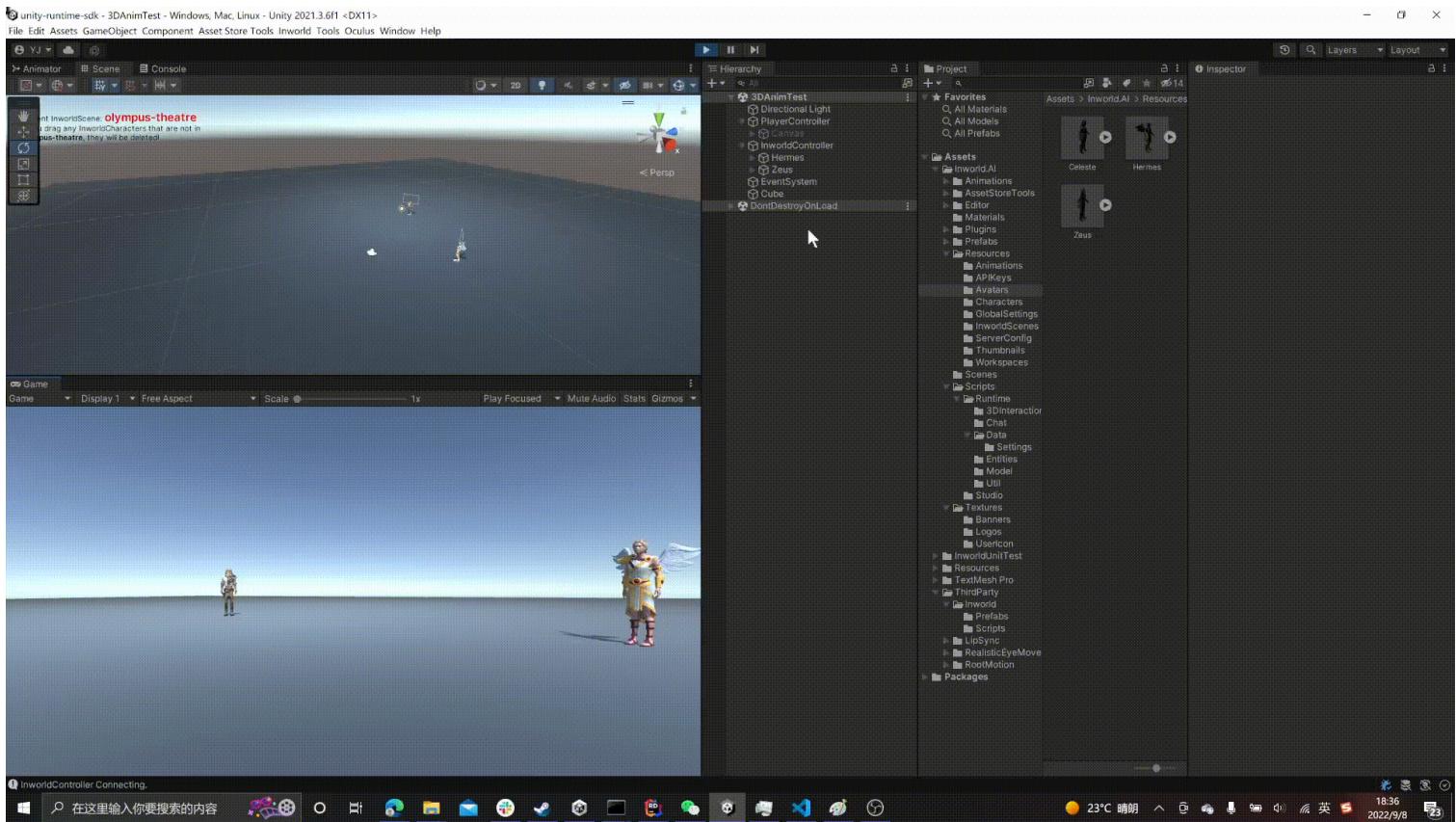
1. All three layers can pass IK data
2. The **Gesture Layer** and **Face Layer** are additive to the **Idle Layer**
3. The **Face Layer** masks faces only
4. The **Gesture Layer** masks all the other parts of the character except the face



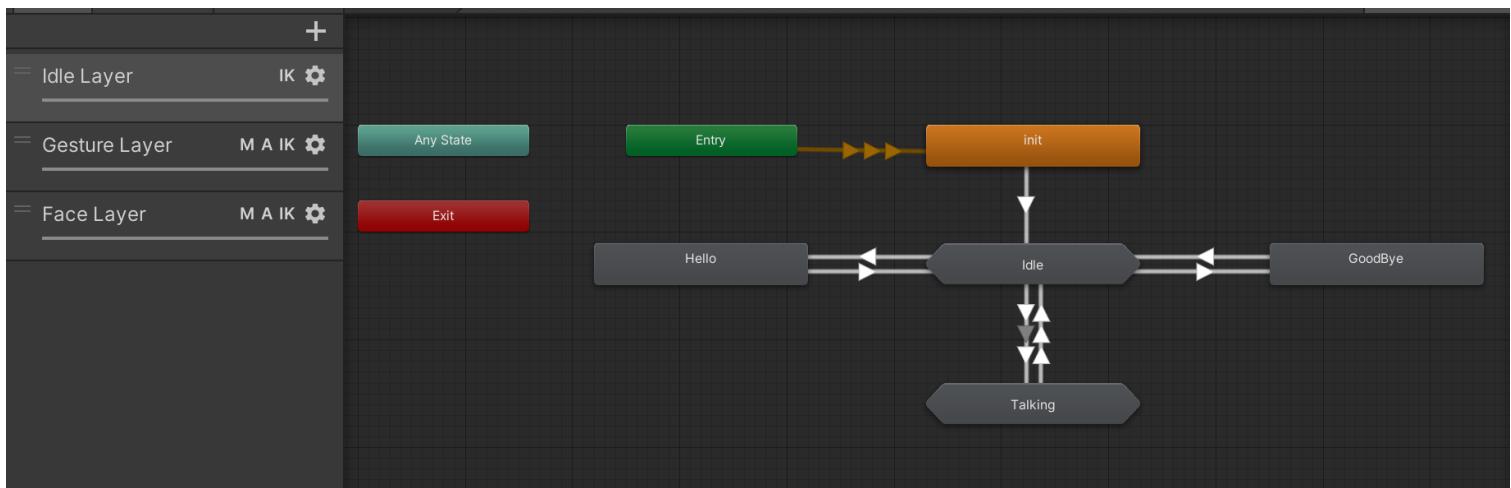
## 1. Idle Layer

The **Idle layer** is triggered by the following game logic,

1. Once the client and server established connection, when you face your character, it switches to a `Hello` state.
2. The character that was previously communicating will switch to a `Goodbye` state
3. When the current character receives audio, it will switch to a `Talking` state
4. Based on breaks in the audio, the character will switch between `Talking` and `Neutral` states

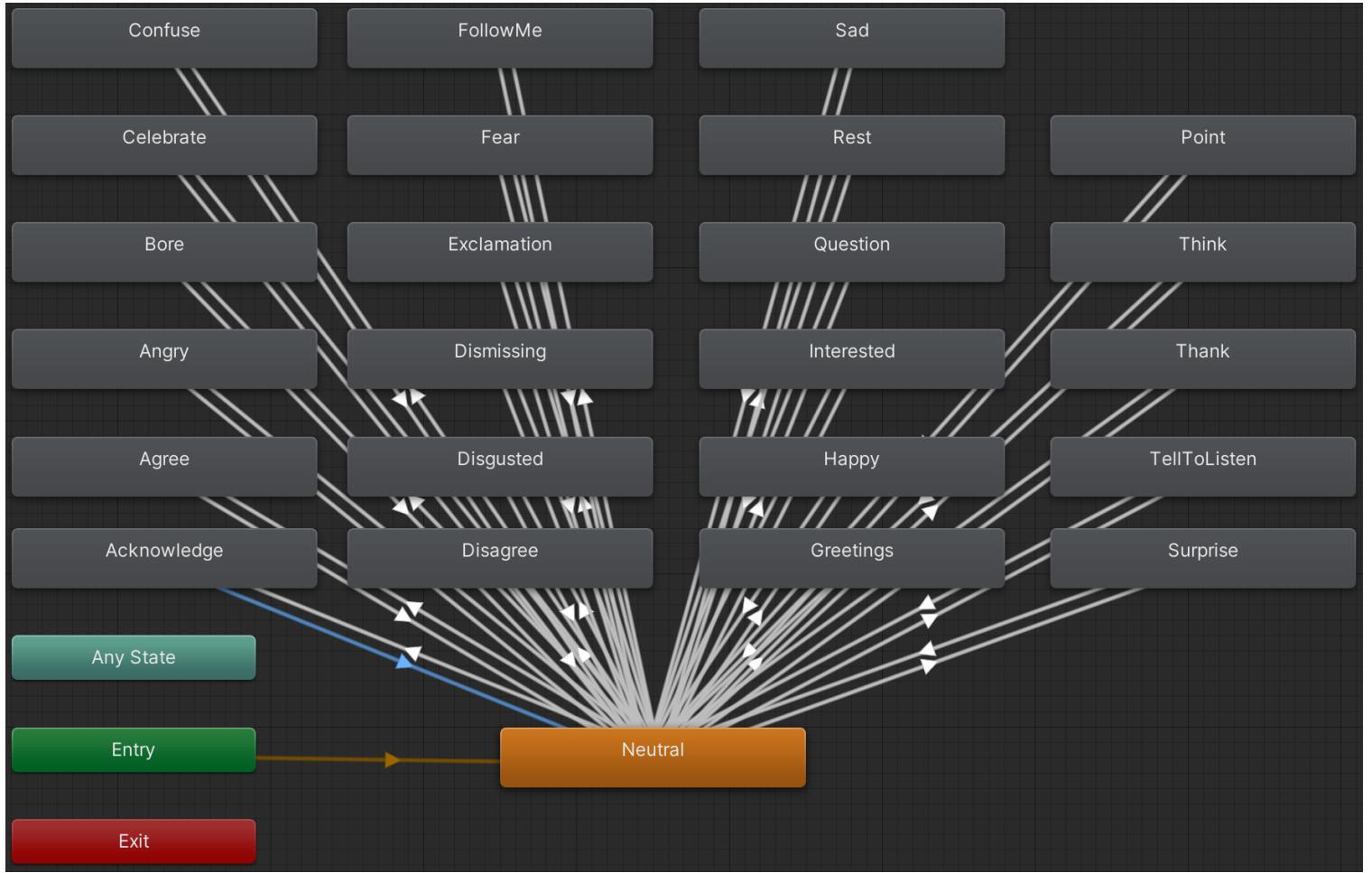


In an `Idle` or `Talking` state, we have provided various types of `Idle` or `Talking` behavior based on the emotion of the character. These are triggered by `EmotionEvents` from the server.



## 2. Gesture Layer

The gesture layer is triggered via `EmotionEvents` or `GestureEvents` sent by the server.



### 3. Face Layer

The face layer is an interface that only has `NeutralIdle`. We implemented facial expressions by modifying **Blend Shapes** directly on this state, via `EmotionEvents` or `GestureEvents` sent by the server.

## Interaction with Server

In live sessions with a character, the server will occasionally send `GestureEvents` or `EmotionEvents` based on the current dialog context. Our code will catch these events and display the appropriate animations. However, our animations and the server events are not in a one-to-one correspondence.

```

public enum SpaffCode
{
    [OriginalName("NEUTRAL")] Neutral,
    [OriginalName("DISGUST")] Disgust,
    [OriginalName("CONTEMPT")] Contempt,
    [OriginalName("BELLIGERENCE")] Belligerence,
    [OriginalName("DOMINEERING")] Domineering,
    [OriginalName("CRITICISM")] Criticism,
    [OriginalName("ANGER")] Anger,
    [OriginalName("TENSION")] Tension,
    [OriginalName("TENSE_HUMOR")] TenseHumor,
    [OriginalName("DEFENSIVENESS")] Defensiveness,
    [OriginalName("WHINING")] Whining,
    [OriginalName("SADNESS")] Sadness,
    [OriginalName("STONEWALLING")] Stonewalling,
    [OriginalName("INTEREST")] Interest,
    [OriginalName("VALIDATION")] Validation,
    [OriginalName("AFFECTION")] Affection,
    [OriginalName("HUMOR")] Humor,
    [OriginalName("SURPRISE")] Surprise,
    [OriginalName("JOY")] Joy,
}

```

```

public enum Type
{
    [OriginalName("GREETING")] Greeting,
    [OriginalName("FAREWELL")] Farewell,
    [OriginalName("AGREEMENT")] Agreement,
    [OriginalName("DISAGREEMENT")] Disagreement,
    [OriginalName("GRATITUDE")] Gratitude,
    [OriginalName("CELEBRATION")] Celebration,
    [OriginalName("BOREDOM")] Boredom,
    [OriginalName("UNCERTAINTY")] Uncertainty,
}

```

If you are interested in the mapping, please check `BodyAnimation::HandleMainStatus()`, `BodyAnimation::HandleEmotion()`, `BodyAnimation::HandleGesture()` for more details.

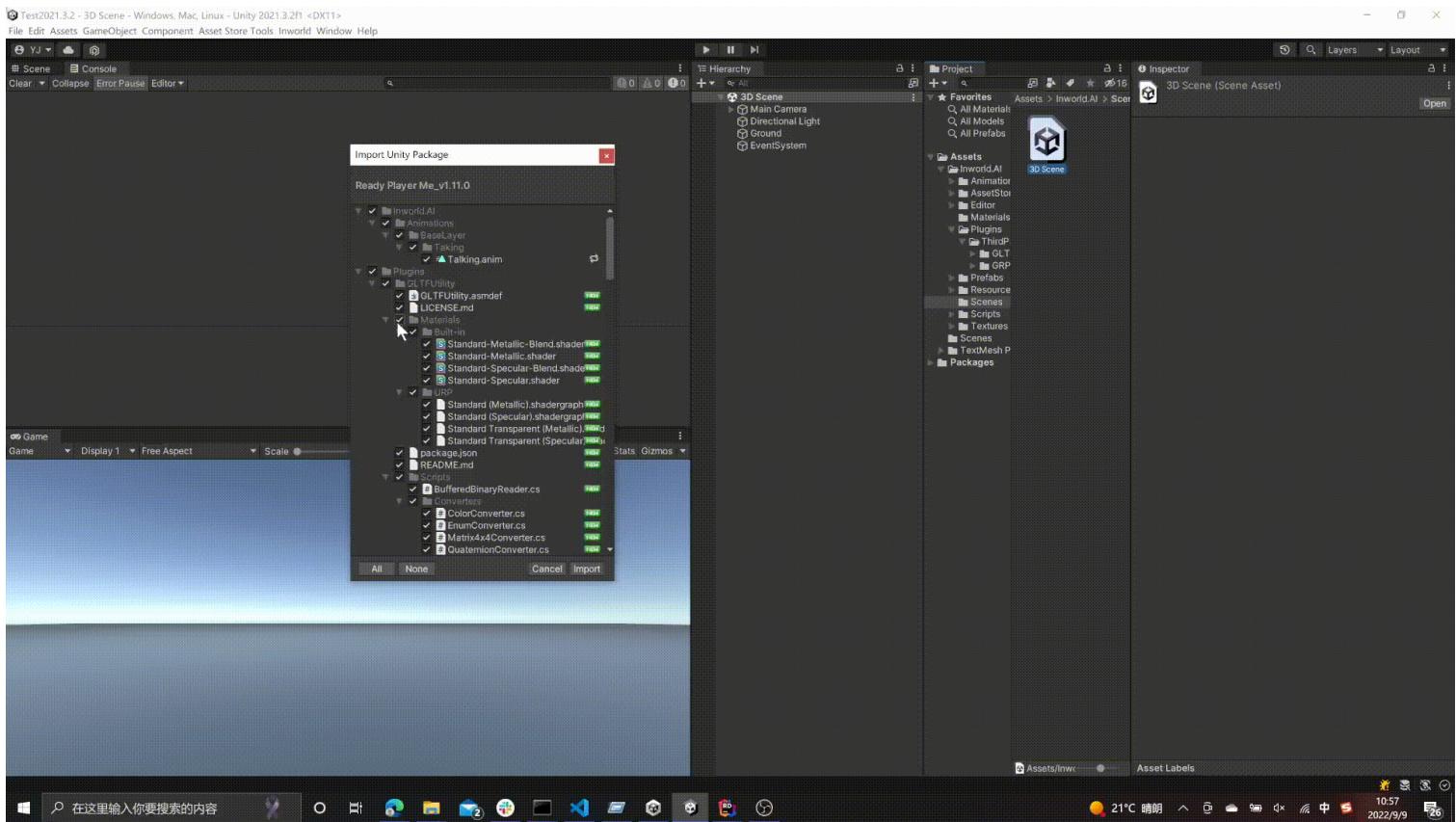
## Using your custom body animations

You may have noticed that legacy characters offered through **Ready Player Me Avatar** are not **T-Posed** when you generate them from the studio website. We recognize that this can make it difficult to perform animation re-targeting. However, **Ready Player Me Avatar** does support **Mixamo** animation. To solve this problem, you can upload to the **Mixamo** server and download the uploaded `.fbx` to replace any of our existing animations.

## 1. Importing the Ready Player Me SDK

Please visit [this site](#) to download the latest **Ready Player Me SDK**. Note the following,

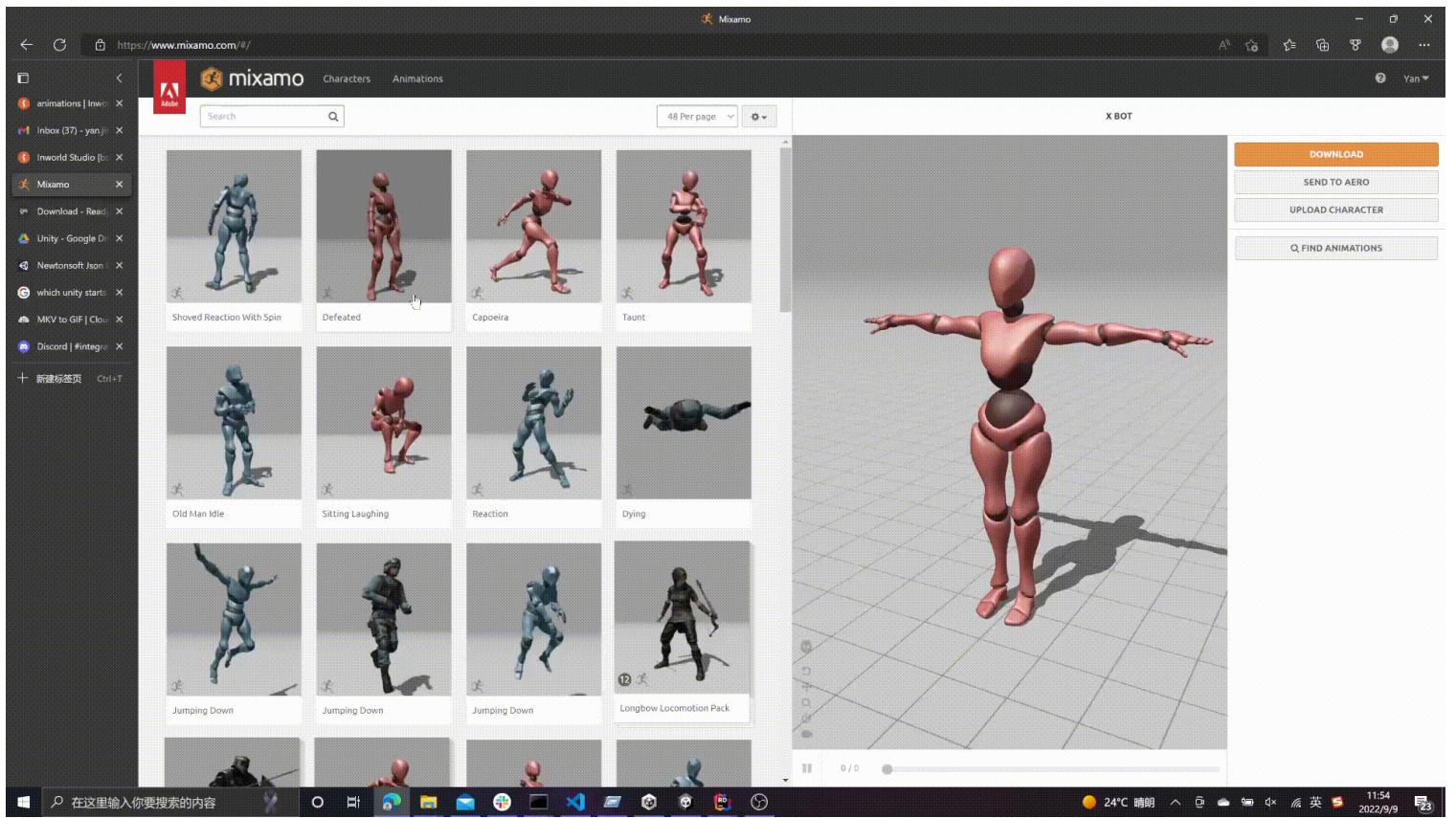
1. The Ready Player Me SDK, as well as Inworld AI, uses **GLTFUtility** as the `.glb` format avatar loader
2. **Newtonsoft Json** was officially included in Unity since 2018.4. When you are importing a package, please exclude these two plugins to avoid a possible conflict.
3. There is a small API formatting bug that involves **GLTFUtility**. It can be solved as follows,



## 2. Downloading Mixamo

**⚠ Note:** To use your own animations, please select [Upload Character](#) on the **Mixamo** website.

When you are downloading the `fbx` from Mixamo, select [Fbx for Unity](#), and [Without skin](#). Drag it into Unity's [Assets](#) folder. These steps are shown below,

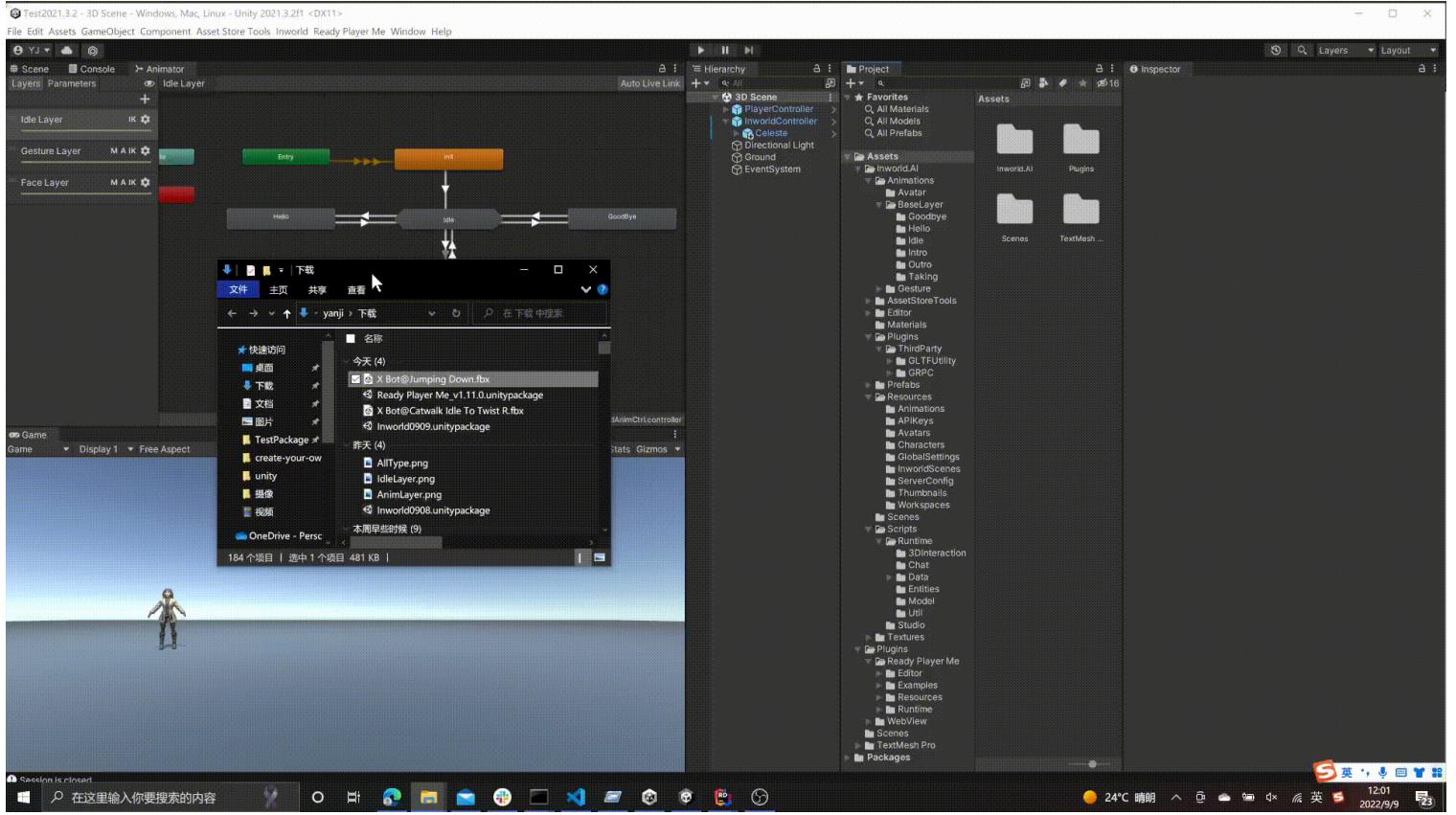


### 3. Replacing animations

**⚠ Note:** The default animation type in the `fbx` you downloaded from Mixamo is `Generic`, which you **CANNOT** re-target.

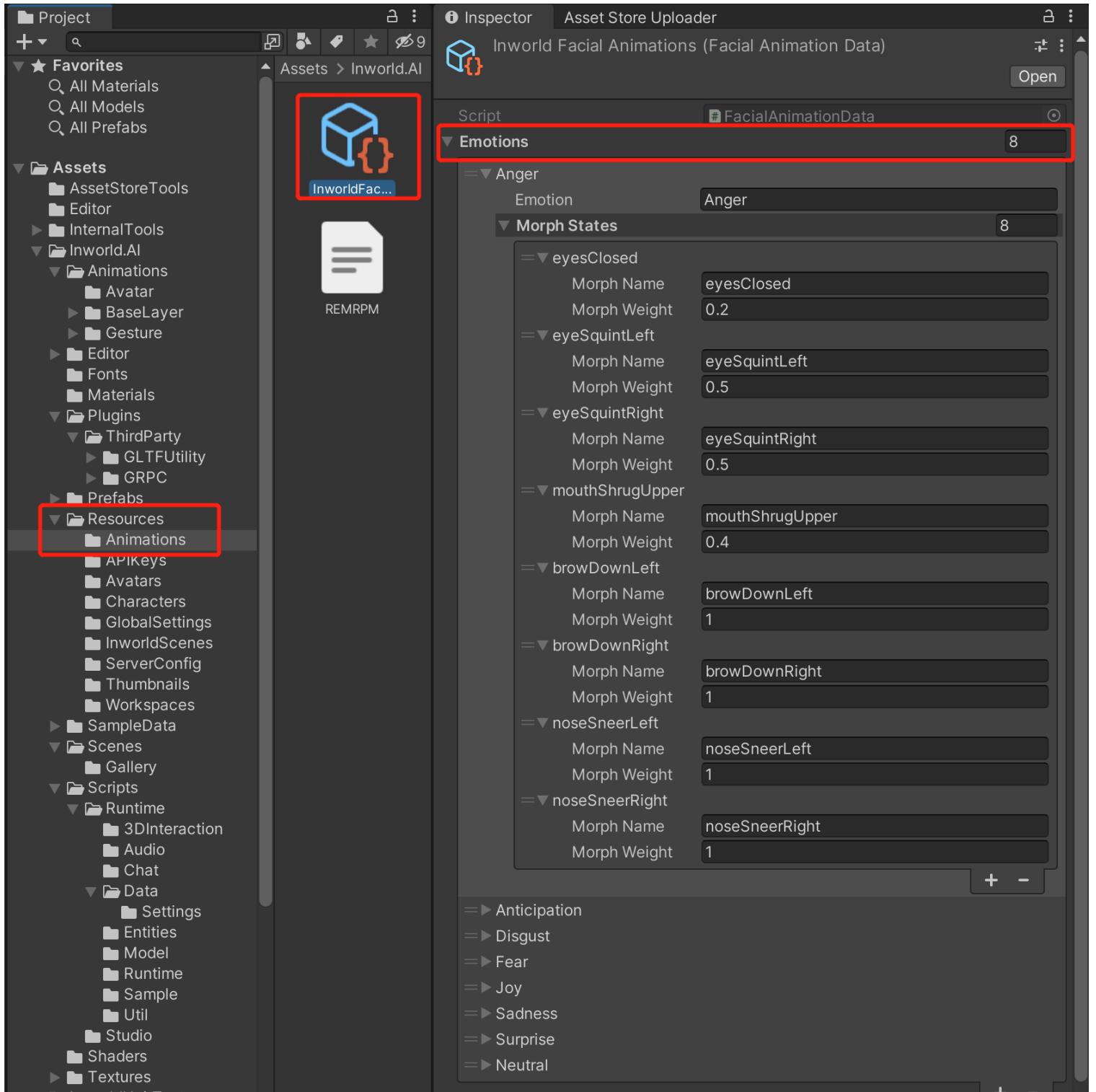
To solve this:

1. Select the `fbx` that you imported at `Rig` section.
2. Switch the `Animation Type` to `Humanoid` and click `Apply`.
3. Right-click the `fbx` and choose `Extract Animations`. This function is provided by the Ready Player Me Unity SDK, and it will generate an `animationclip`.
4. You can select any state in `InworldAnimCtrl` and replace the `animationclip` with your new clip. This is shown below,

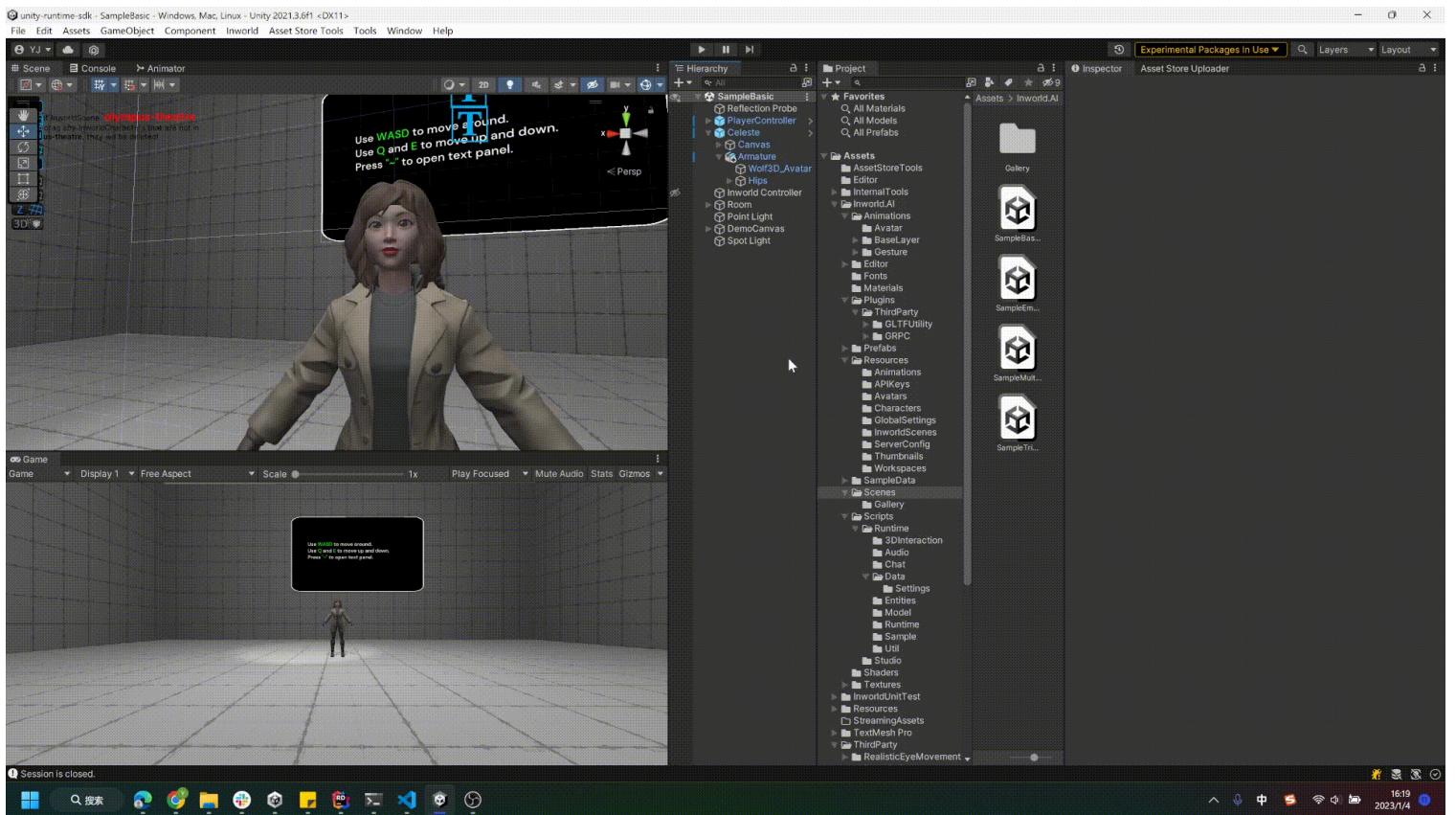


# Configure your custom facial animations

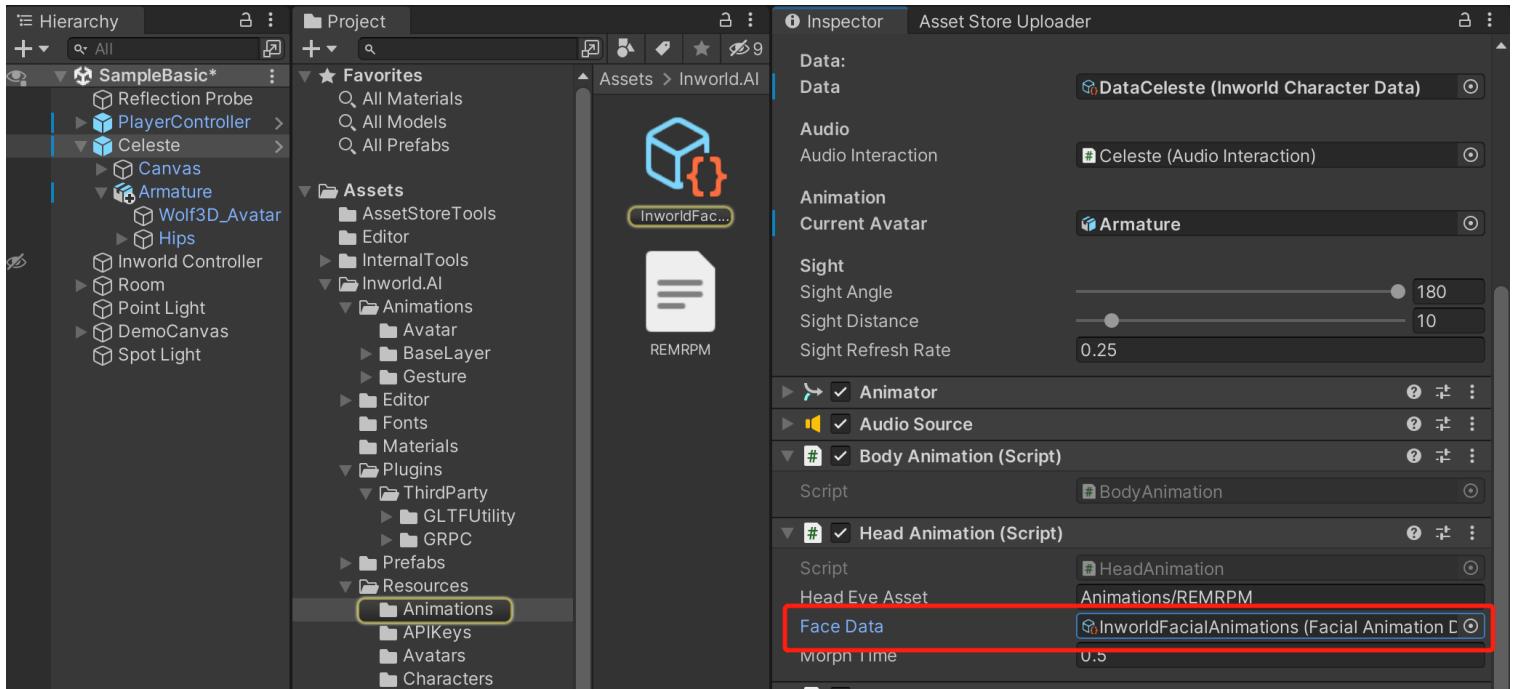
Instead of using **Animation Clips**, we directly modify the **Blend Shapes** on the **SkinnedMeshRenderer** of the avatar. You could check each state at [Resources > Animations > Inworld Face Animations > Emotions](#).



If you want to change the behavior, you can drag each index, until you find the face expression you like, then restore the data.



**Note:** We recommend you duplicate this **Inworld Face Animations**, then update and allocate your generated asset to your **InworldCharacter's Head Animation**.



On this page



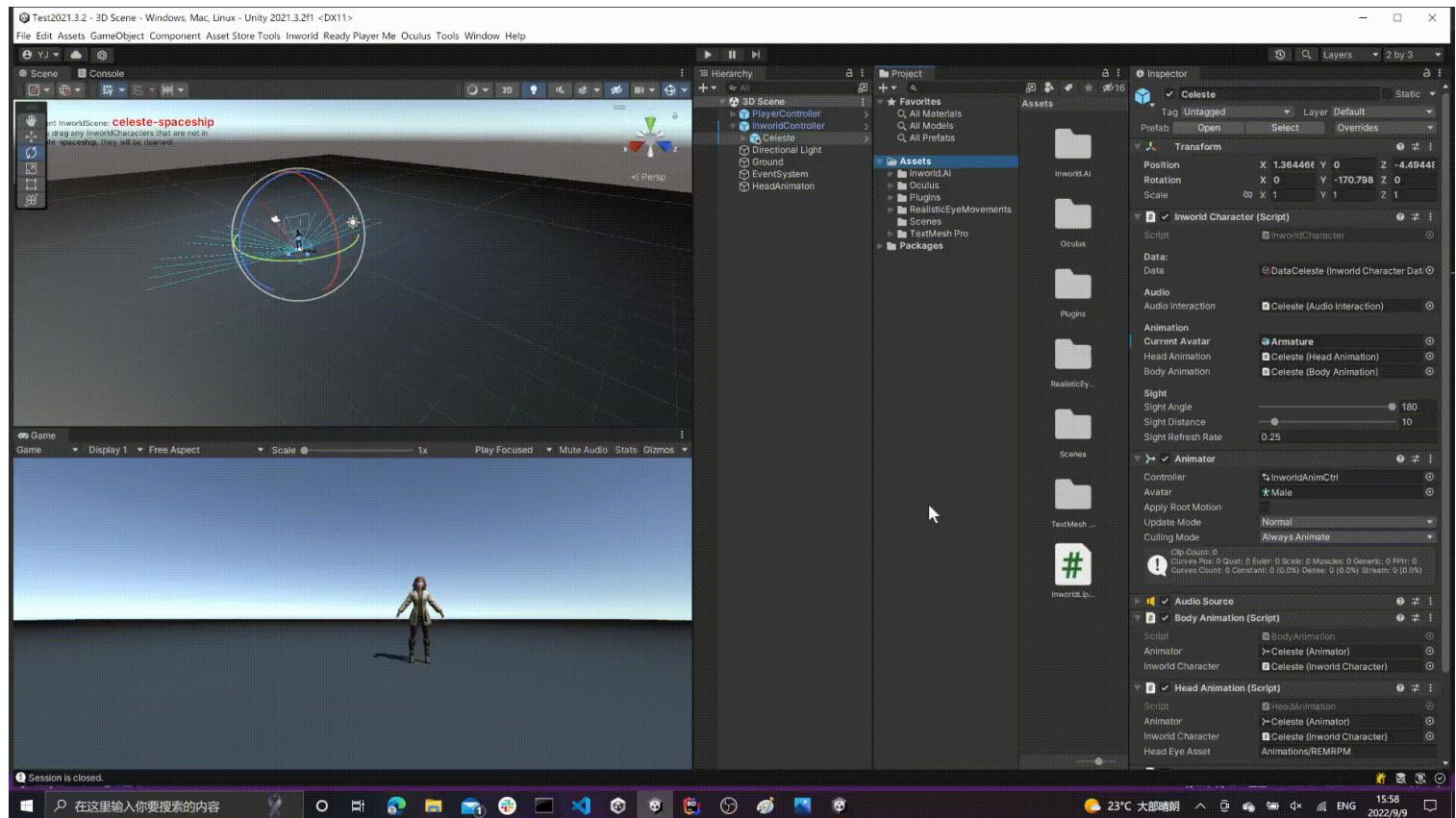
# realistic-eyemovemt

## Head and Eye Movement

By default, avatars generated in the **studio** support basic head movements (e.g., looking at the player). If you want more detailed head and eye movements (e.g., nodding, blinking, etc.), then it is recommend that you use **Realistic Eye Movements** as we have already prepared a `.json` file for loading head/eye related parameters for `Realistic Eye Movement`. It is located at `Assets/Inworld.AI/Resources/Animations/RPMREM.json`.

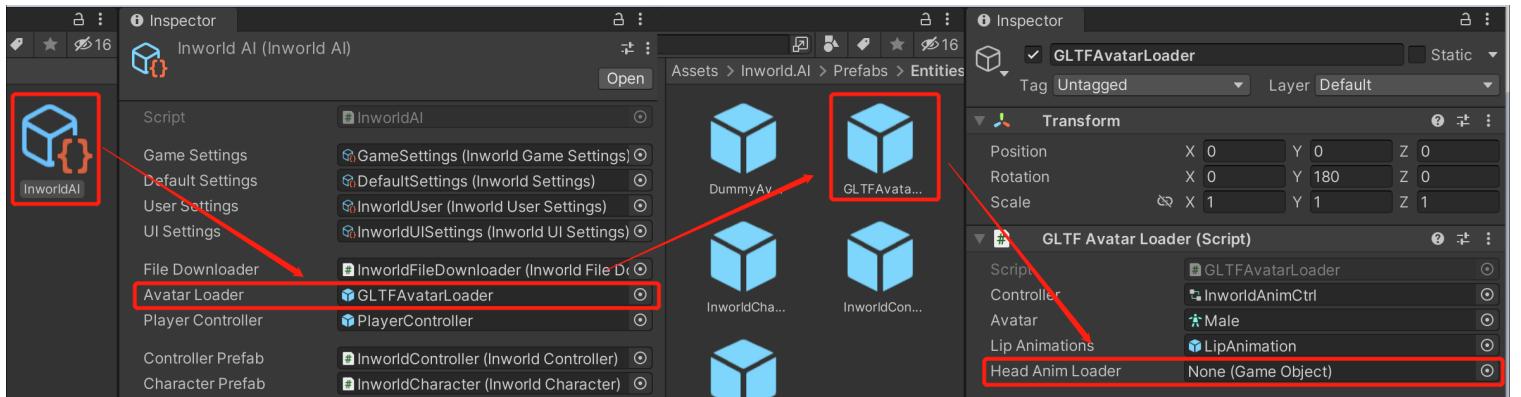
## 1. Preparation

Once the package has been imported, find the prefab `InworldCharacter`, and add the components `LookTargetController` and `EyeAndHeadAnimator`.



In the **Inworld AI Unity SDK**, after a `.glb` model is dragged into the `UnityScene`, it will trigger events that lead to `IAvatarLoader::ConfigureModel()`.

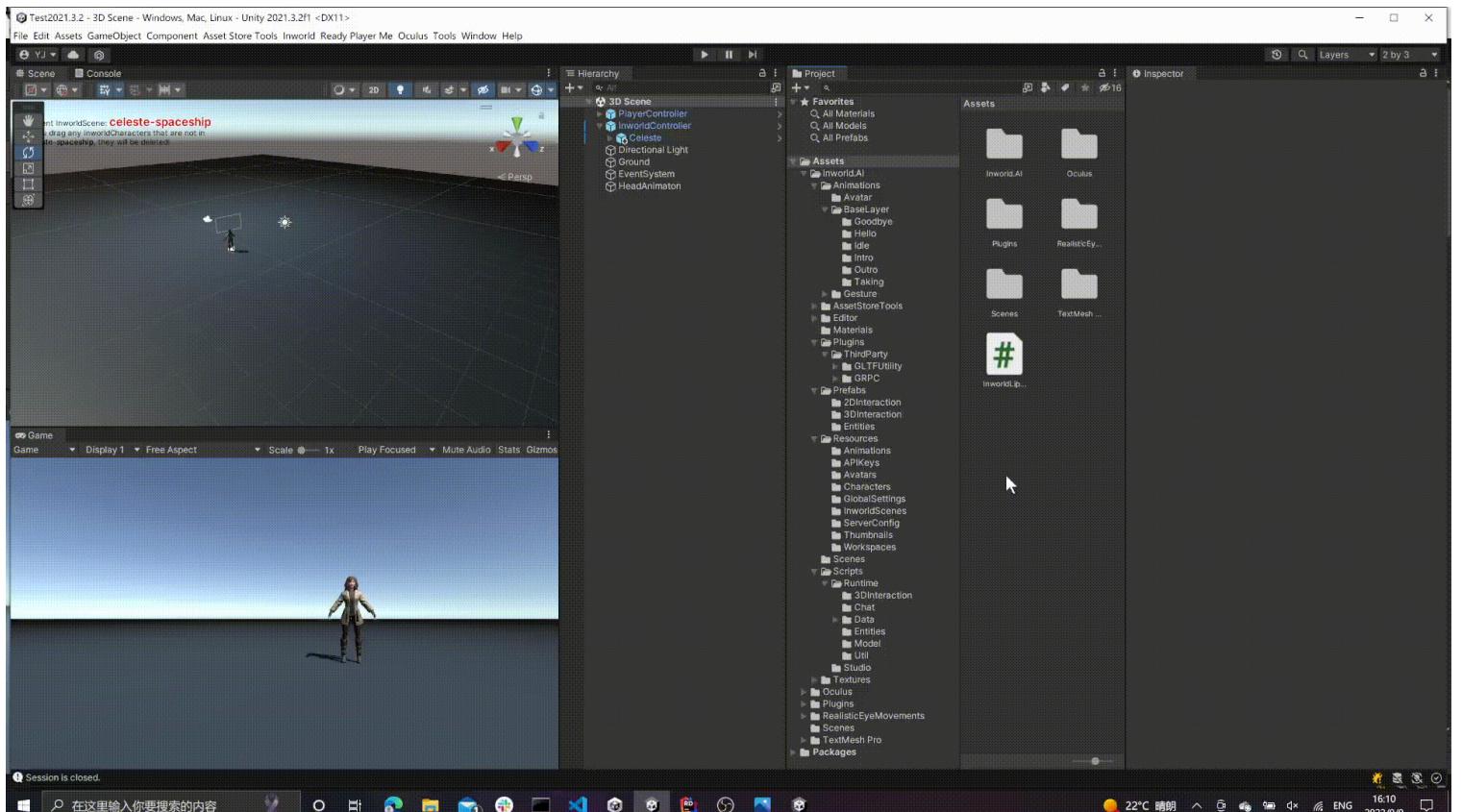
By default, the IAvatarLoader is **GLTFAvatarLoader**, and it will call **IEyeHeadAnimLoader::SetupHeadMovement()** to set up head and eye movement if it exists.



Let's create our own integrations.

## 2. Implement Interfaces

Create a **Monobehavior** class that inherits **IEyeHeadAnimLoader**. You can name it whatever you want, e.g., **InworldHeadAnim**.



Let's load that by implementing the interface function.

```

namespace Inworld.Model.Sample
{
    public class InworldEyeHeadLoader : MonoBehaviour, IEyeHeadAnimLoader
    {
        [SerializeField] string m_HeadEyeAsset = "Animations/REMRPM";

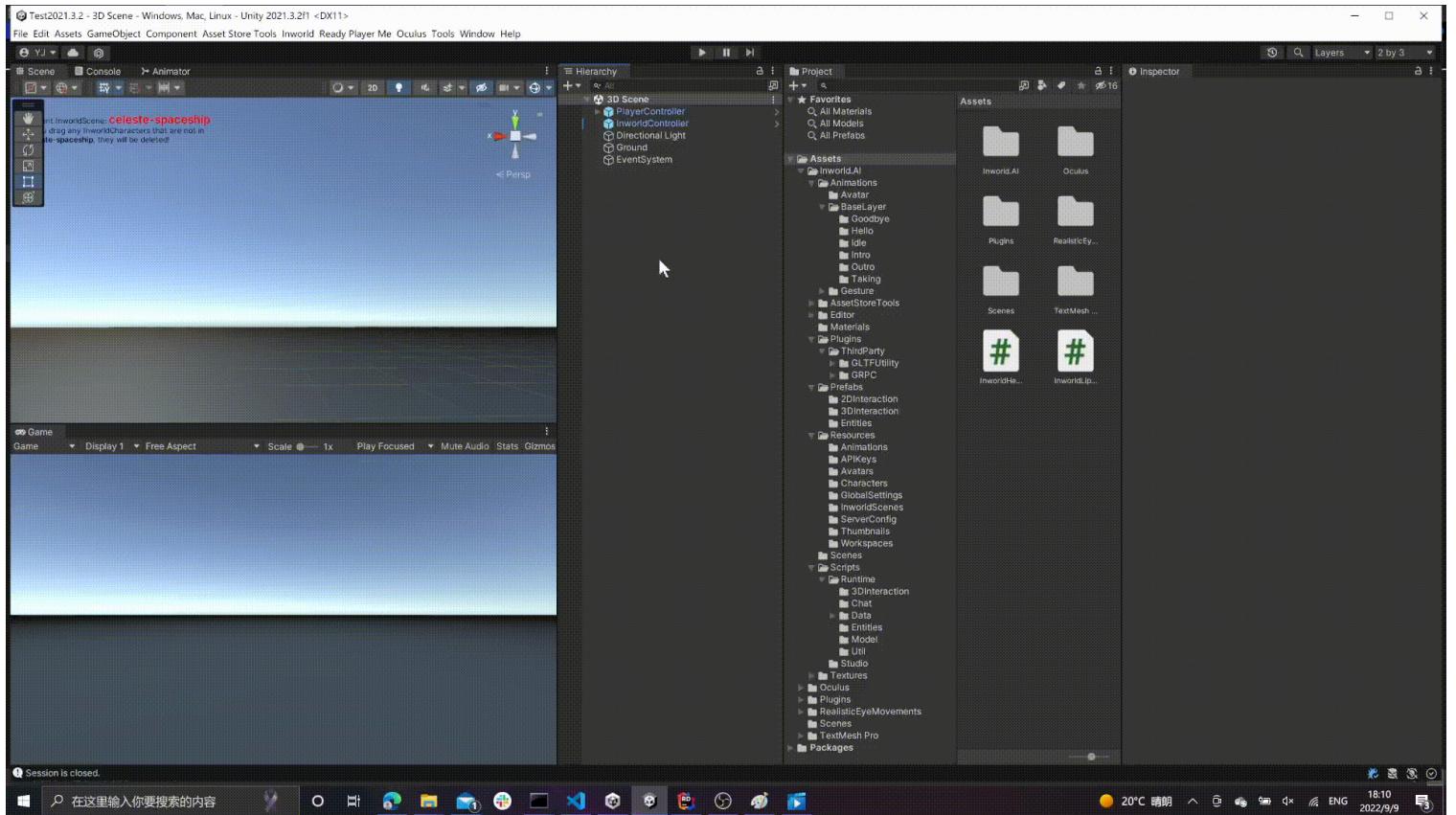
        public void SetupHeadMovement(GameObject avatar)
        {
            EyeAndHeadAnimator eyeHead = avatar.GetComponent<EyeAndHeadAnimator>();
            if (!eyeHead)
                return;
            TextAsset textAsset = Resources.Load<TextAsset>(m_HeadEyeAsset);
            if (!textAsset)
                return;
            EyeAndHeadAnimatorForSerialization import =
JsonUtility.FromJson<EyeAndHeadAnimatorForSerialization>(textAsset.text);
            eyeHead.headBoneNonMecanim = Utils.GetTransformFromPath(avatar.transform,
import.headBonePath);
            eyeHead.ImportFromJson(textAsset.text);
        }
    }
}

```

**⚠ Note:** We need to set the value of `headBoneNonMecanism` because the `EyeHeadAnimator` has not been fully initialized yet, i.e., `Transform` is null. We need to pass the avatar's transform, otherwise it will throw an error.

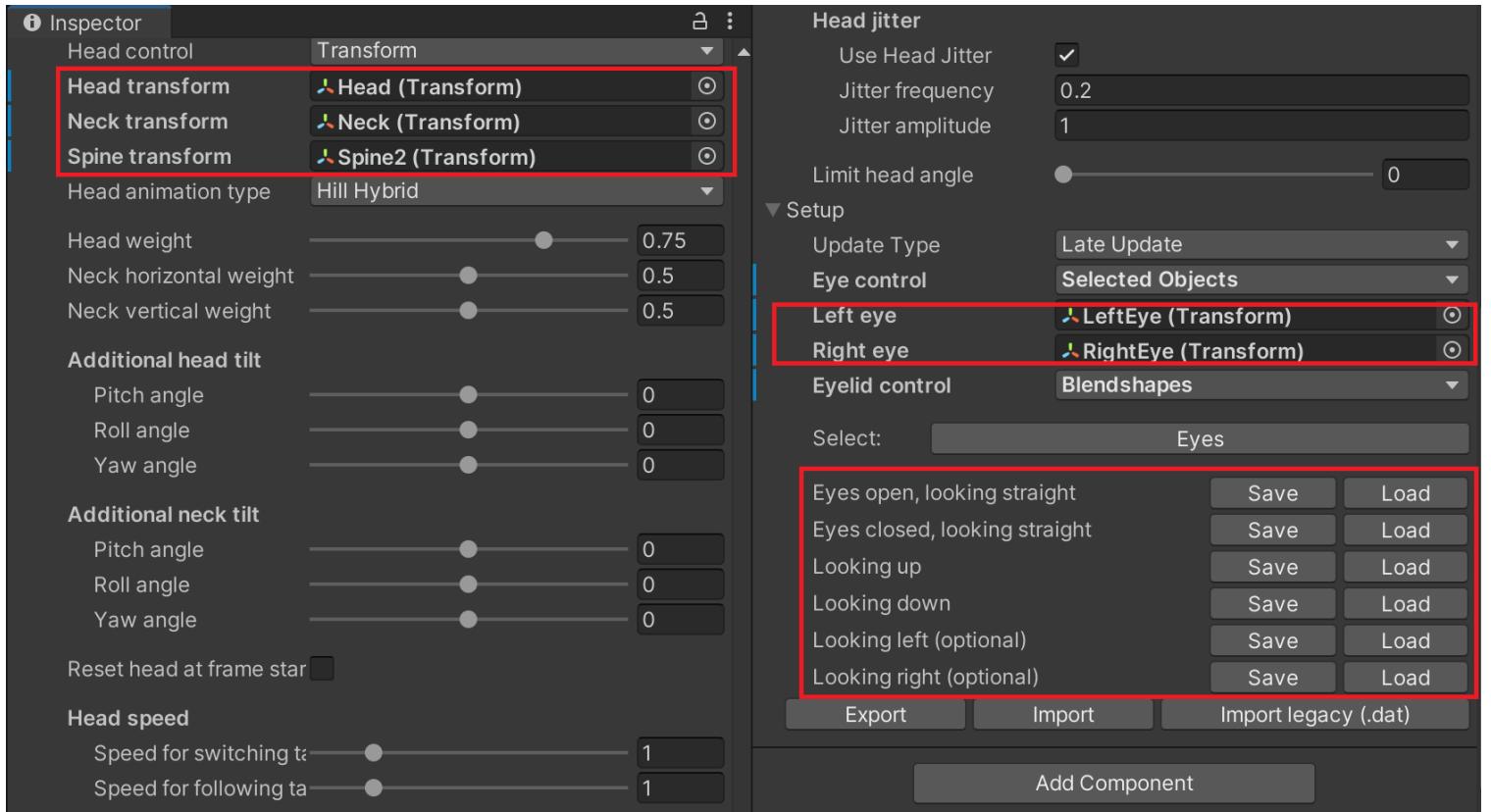
### 3. Create and configure prefabs

1. Create an empty `gameObject`. Attach the script `InworldHeadAnim` that you just implemented.
2. Create a prefab for that object.
3. Delete the object from the scene.
4. Find the `GLTFAvatarLoader` prefab in your `Assets` folder, and drag the prefab that you created into the `GLTFAvatarLoader`'s `Head Animation`.

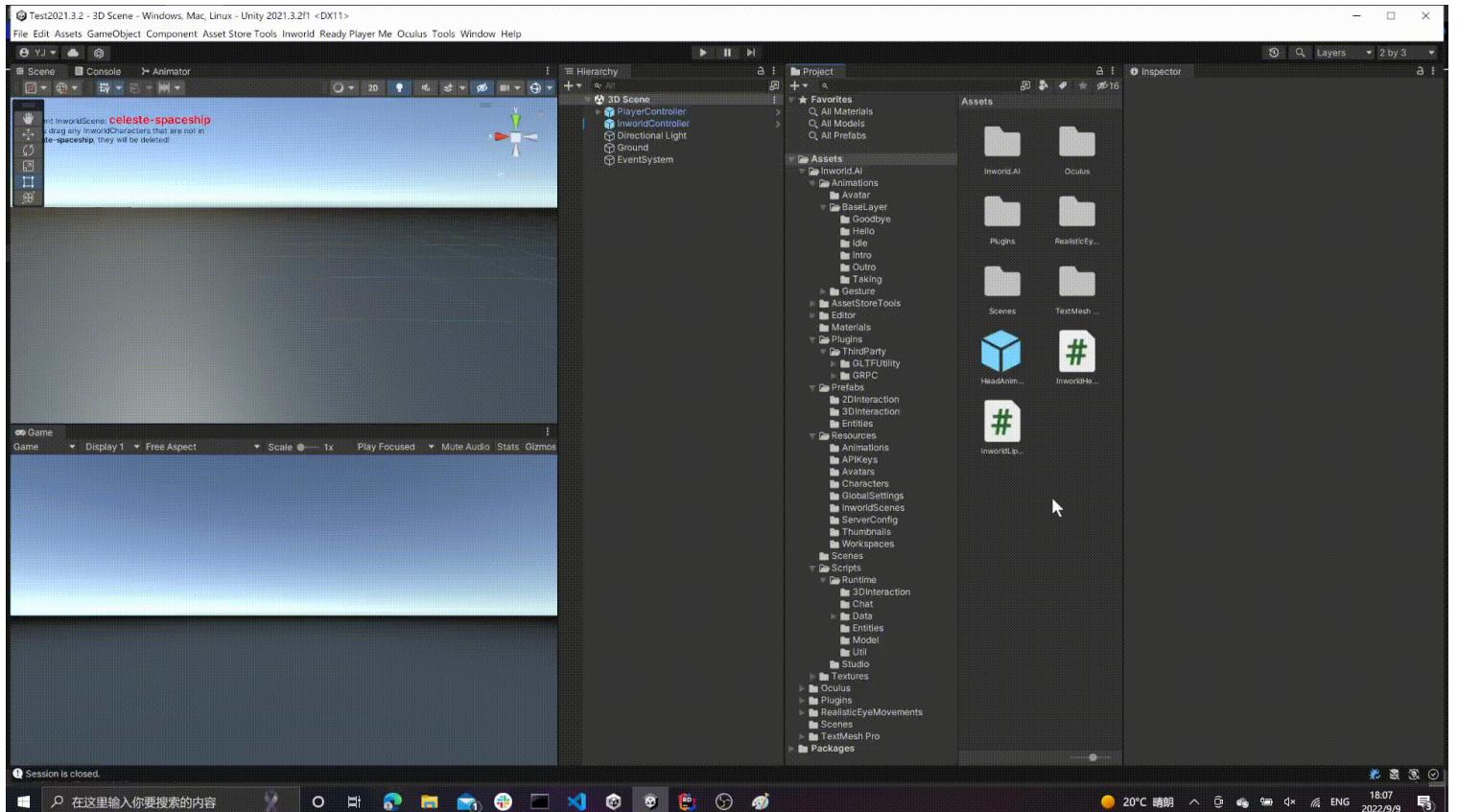


## 4. Runtime

You can test your avatar's head animation. Once the avatar has been dragged into the scene, check if `LookTargetController` and `EyeAndHeadAnimator` exist, and whether the data for the head and eyes has been set.



You can adjust the parameters and observe differences in nodding and blinking behavior. Check the [website](#) for more information.



On this page



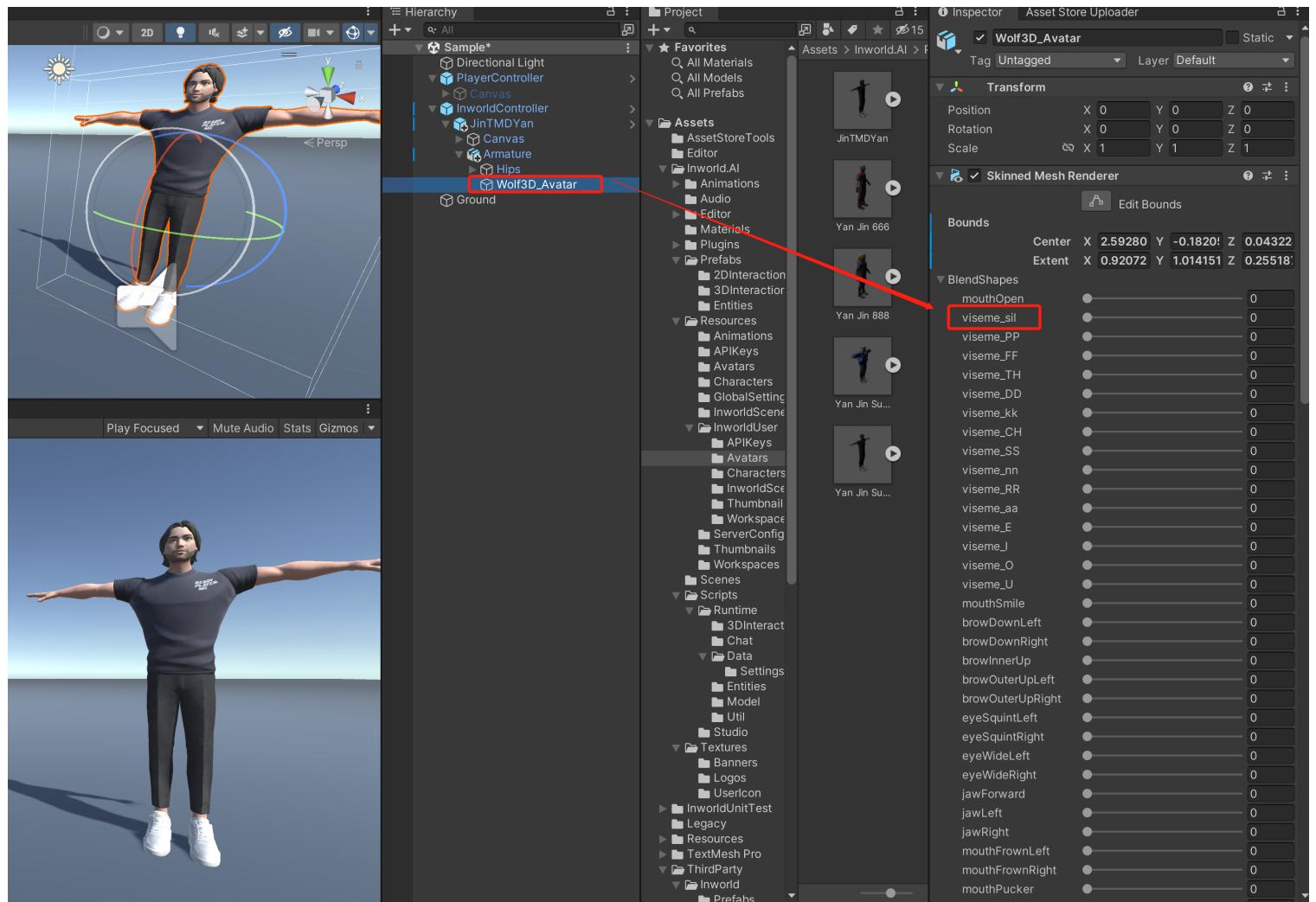
# ovr-lipsync

## Lip syncing

After Inworld Unity SDK Ver. 2.0.3, Lipsyncing is automatically embedded in the package.

All the characters fetched by **Inworld Studio Panel** would automatically embed this feature.

Our Lipsync received phoneme data from server time to time, transferred into viseme data, then modified on the **Skinned Mesh Renderer(Blend Shape)**.

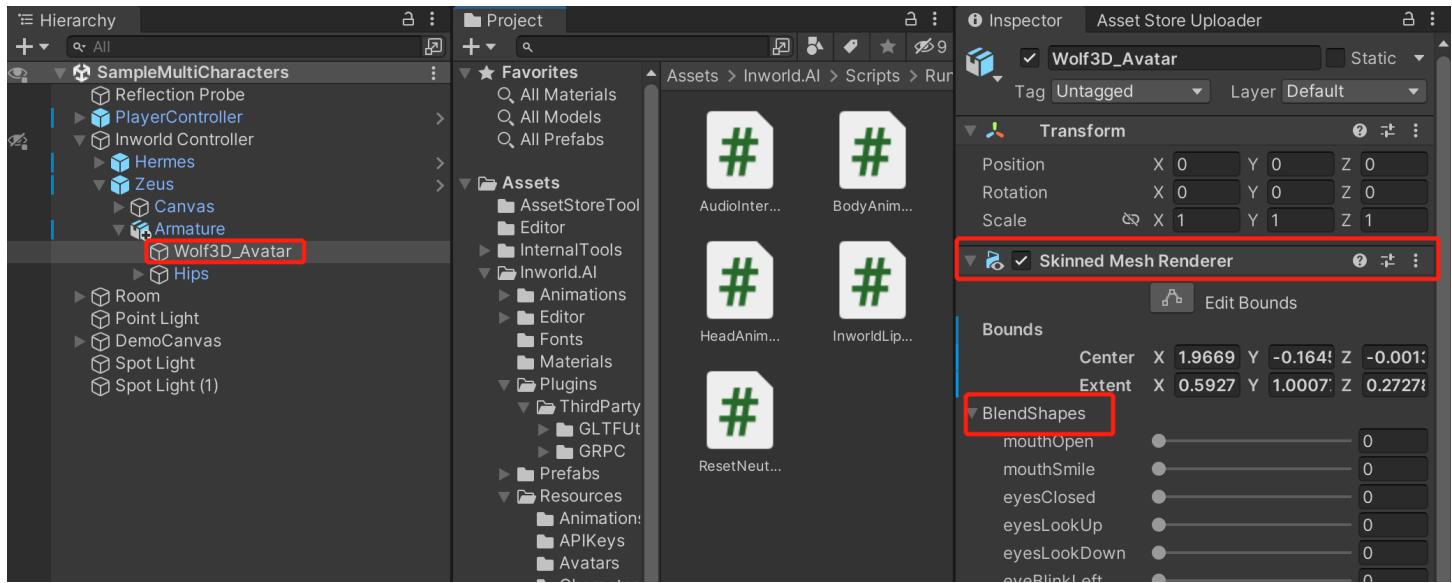


**Enable custom avatar with lip syncing**

If you'd like to enable lip syncing for your own avatar instead of Ready Player Me, please check the followings.

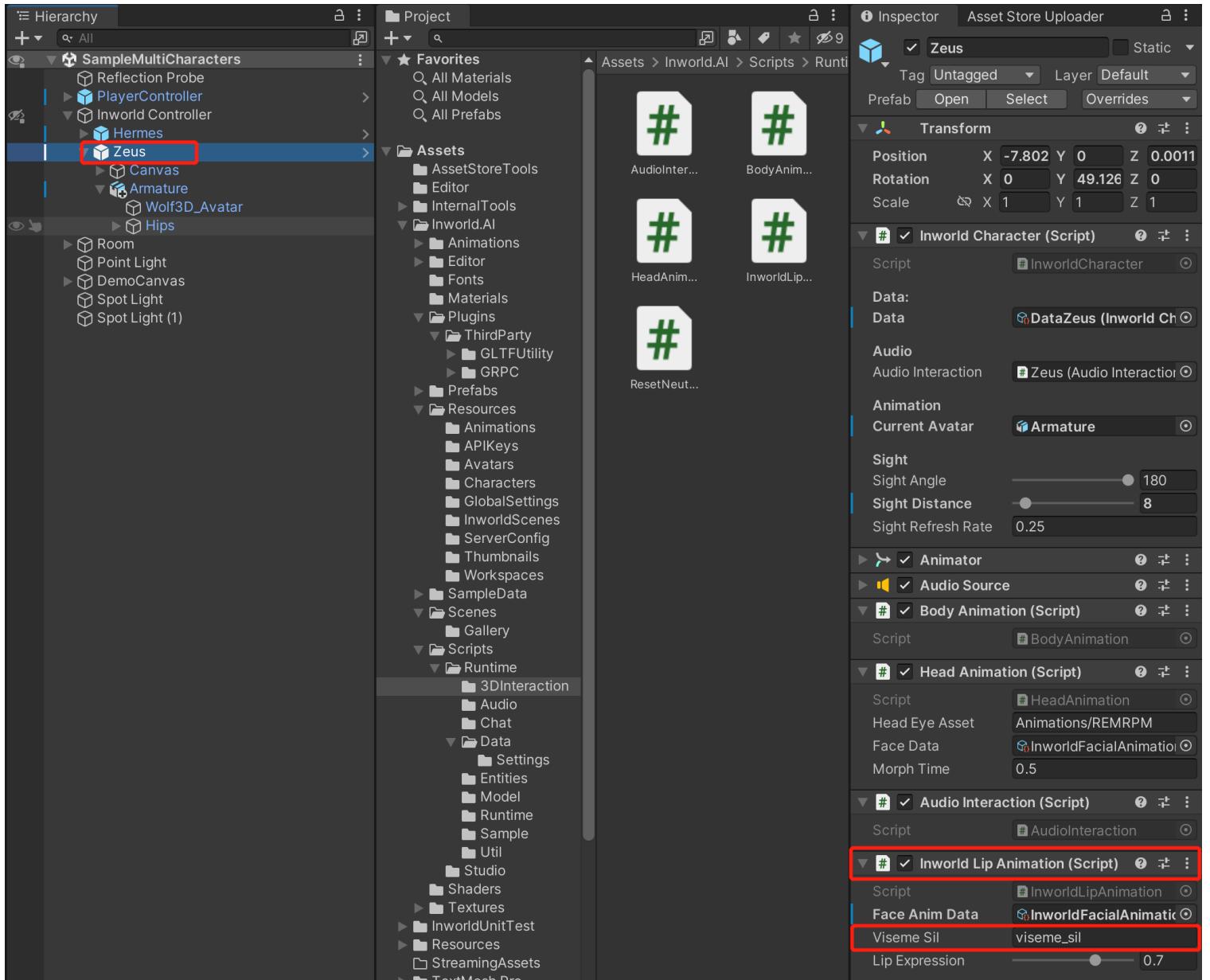
## Prerequisites

1. Your avatar is exported with **Blend Shape** in 3D modeling software.
2. After imported, under the **BlendShapes** of **Skinned Mesh Renderer**, there are indices for visemes.



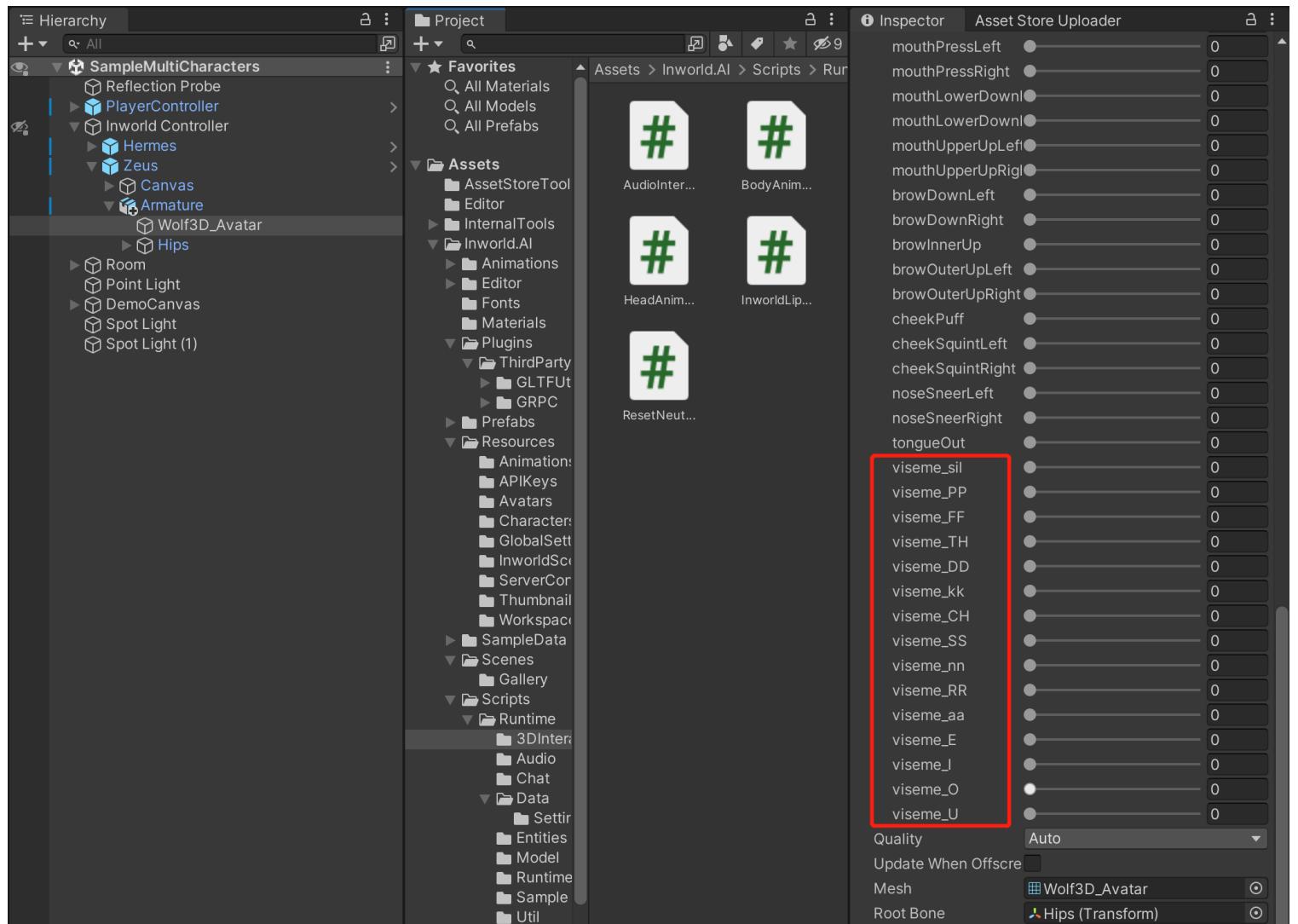
## Set the starting index

After Inworld SDK Ver. 2.1.3, if the name of your first viseme is not `viseme_sil`, you can change the name at **InworldLipAnimation**.

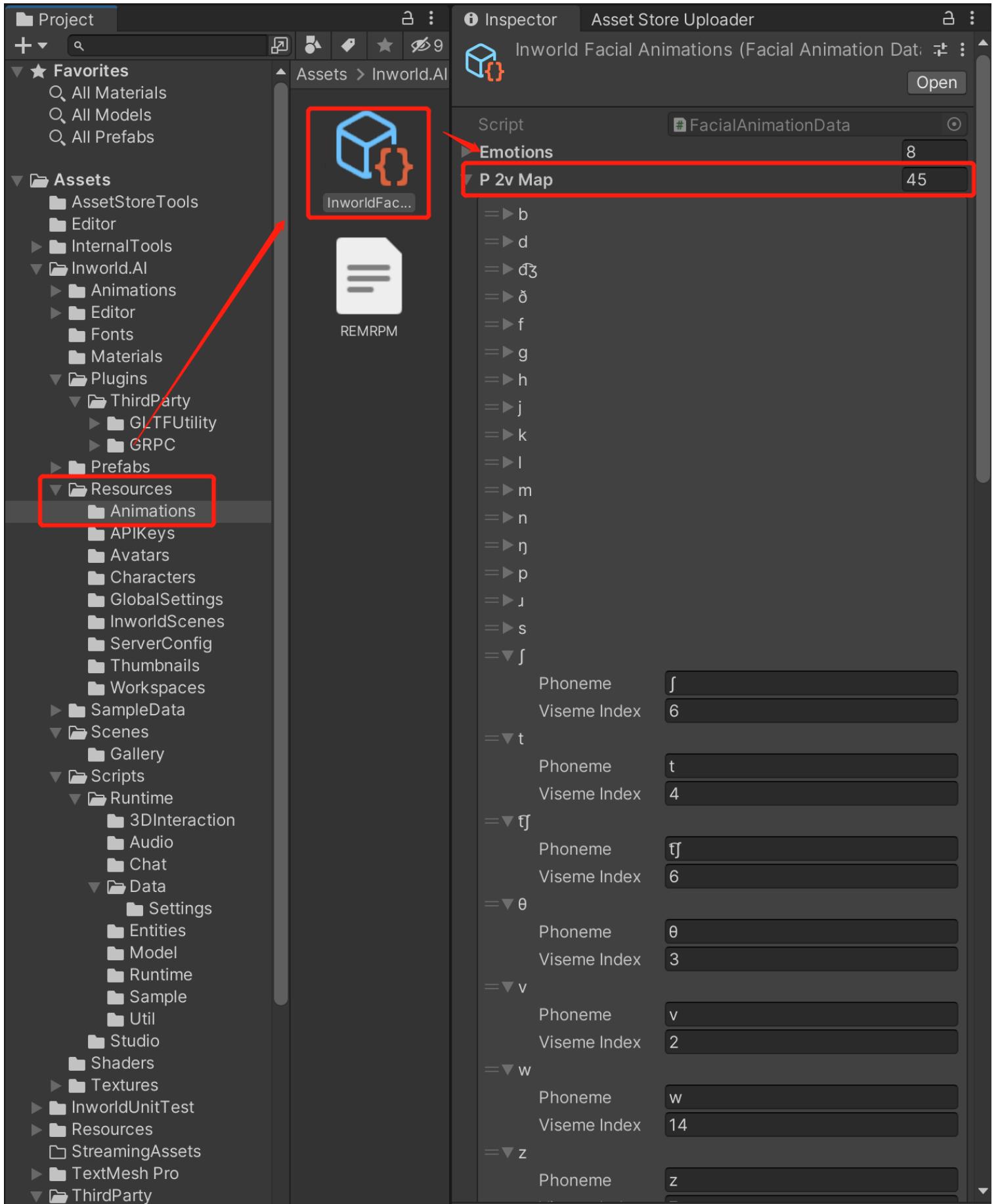


## (Optional) Configure P2V Map

If the viseme indices is continuous and in the right order (From Sil to U), you can skip this step.

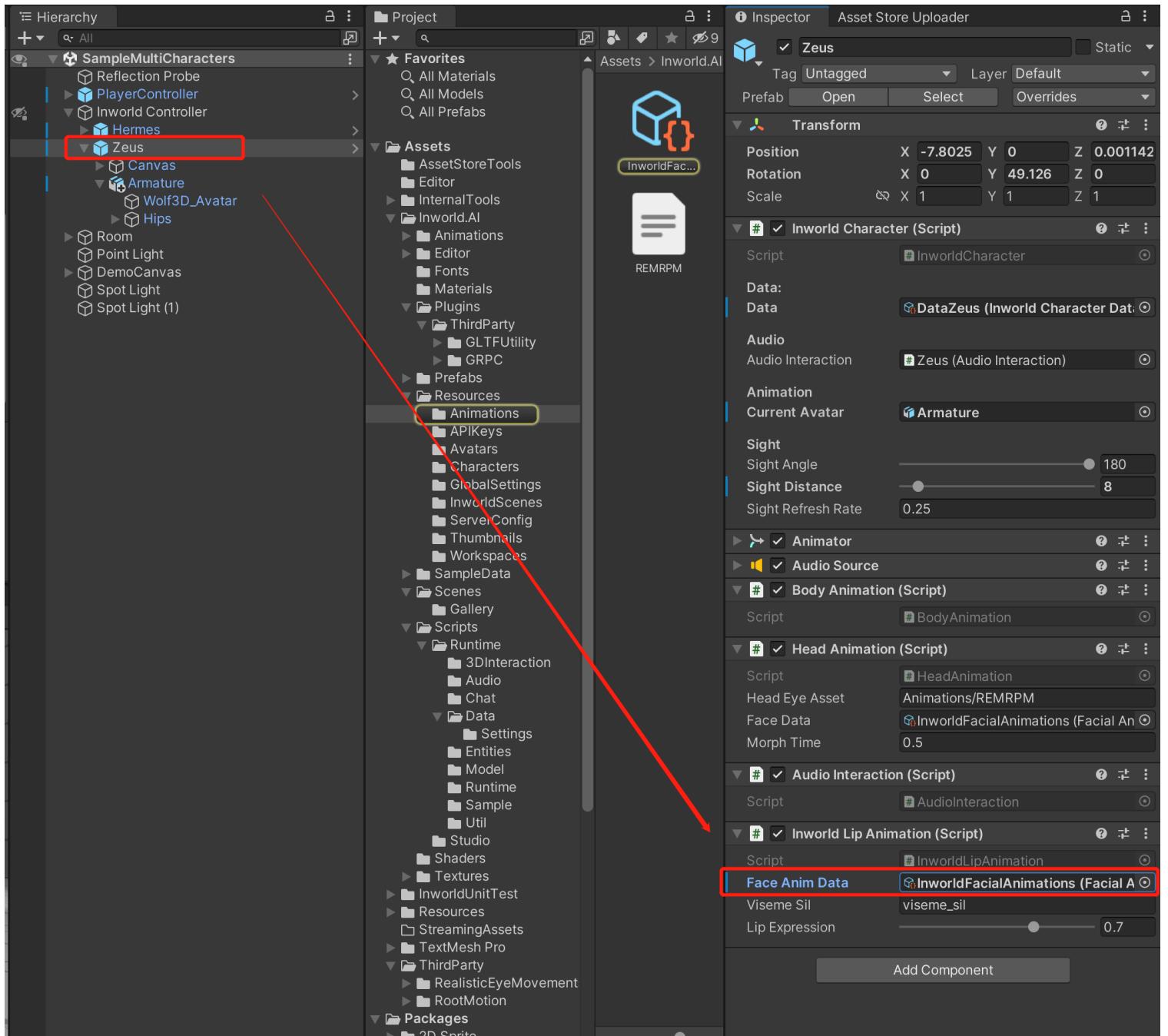


Otherwise, you need to configure the P2V map. You can check the data at [Resources > Animations > Inworld Face Animations > P2V Map](#).



If you don't know how to set the data, you can check the reference [here](#).

**Note:** We recommend you duplicate this **Inworld Face Animations**, then update and allocate your generated asset to your **InworldCharacter's InworldLipAnimation**.



## Before 2.1.2

The Lipsync library before Ver. 2.1.2 uses **Oculus lipsync**. If you want to continue using this package, and integrate with your own lipsync, please replace it in `GLTFAvatarLoader > LipAnimations`.

## Upgrade from 2.1.1 or lower

If you want to upgrade to Ver. 2.1.2 and higher, please notice that the legacy package (Ver. 2.1.0 or lower) may overwrite **Oculus lipsync** package if both packages have been installed in your project. We recommend you also upgrade **Oculus lipsync** as well.

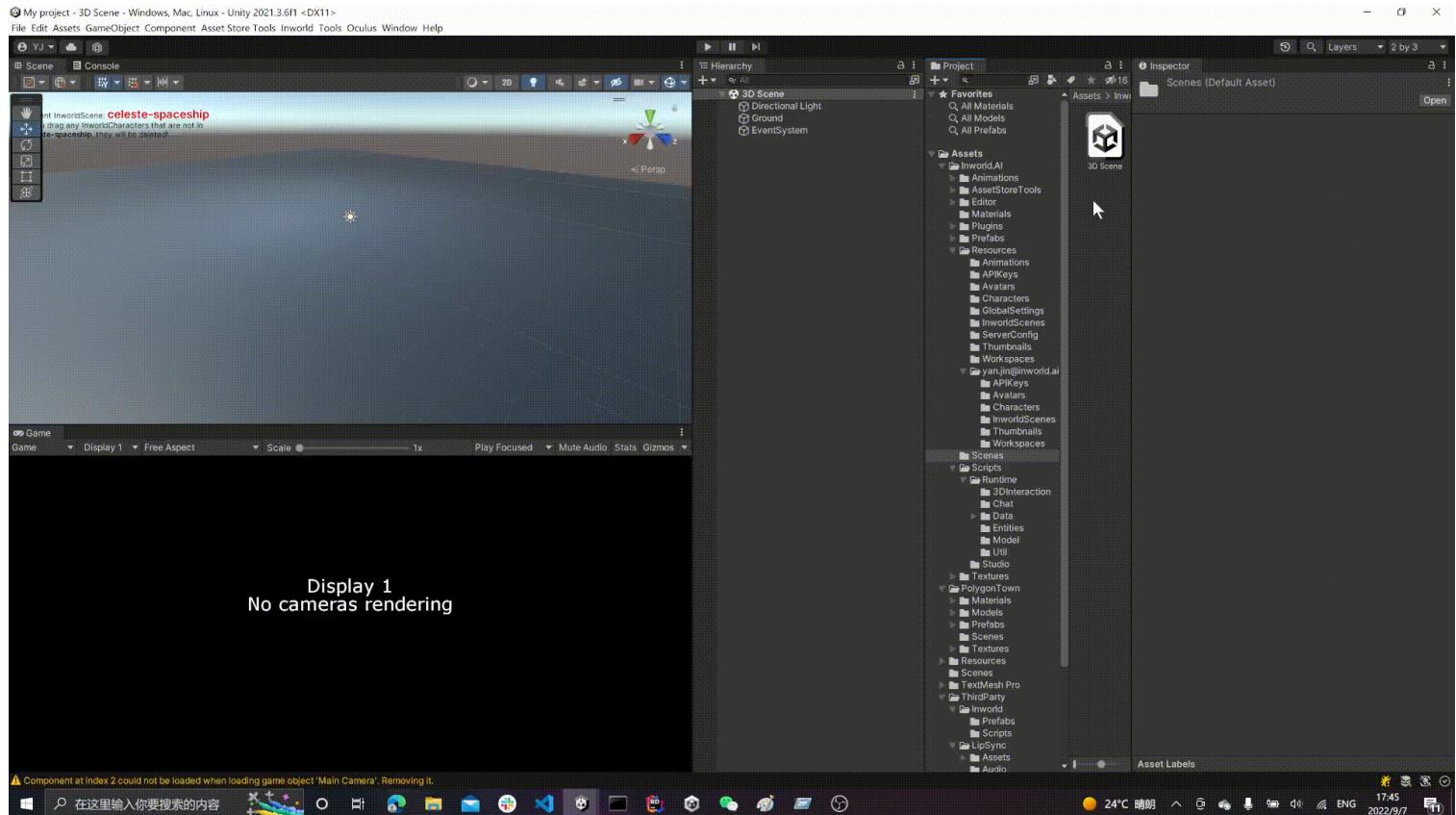
On this page



# editors

## Inworld Editor

The editor files are located in `Assets/Inworld.AI/Editor`. They are mainly used to implement the `Inworld Studio Panel` which is displayed below.

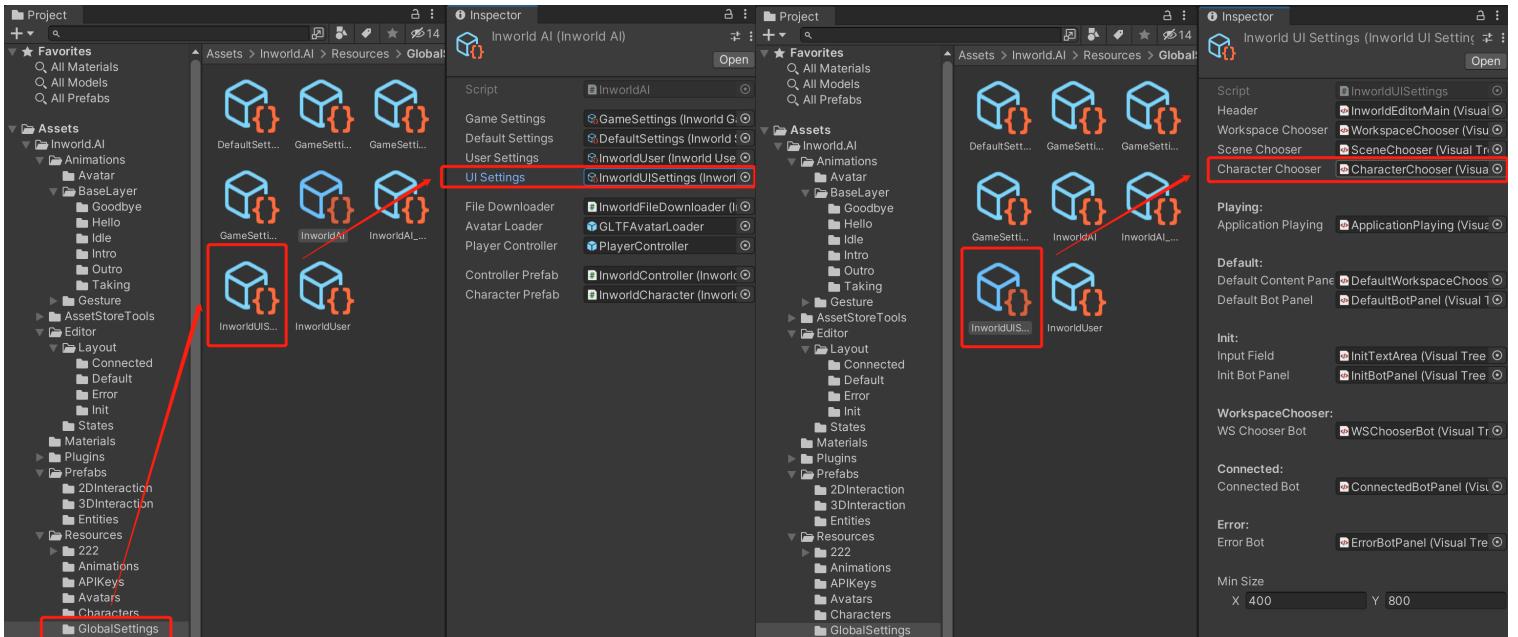


## UI Elements

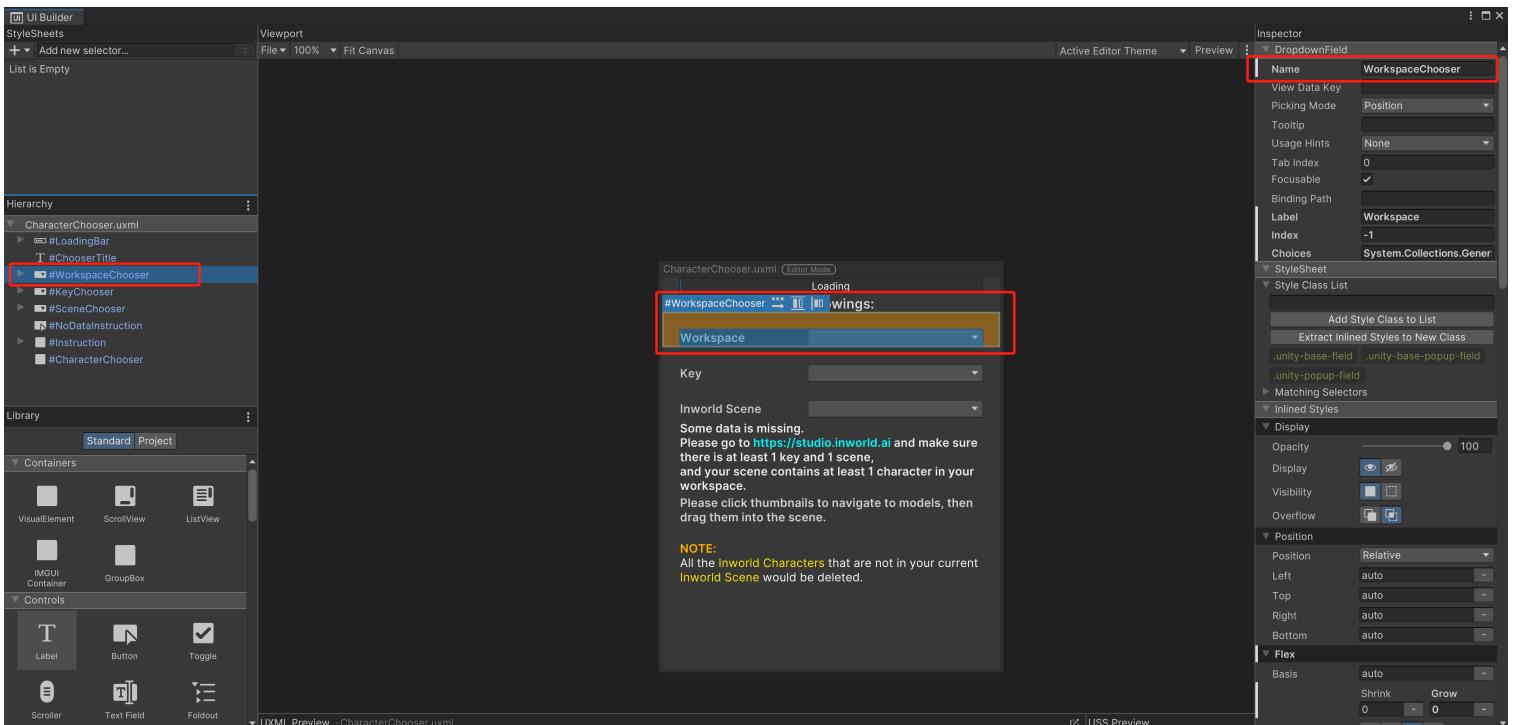
The **Inworld AI Unity SDK** uses **UI Elements** to paint the panel. Each page contains a `Content` panel for the contents and a `Bot` panel for the buttons.

Let's take the `Content` panel in the Character Choosing Page as an example.

1. In `Project` tab, right-click and select `Inworld Setting Panel`



2. Double click `UI Settings` in the setting panel. It will navigate to the Unity's UI Element building tool.



3. From that UI editor, you can see the components are listed in the `Hierarchy` (on the left). You can reference these components after naming them in the `Inspector` (on the right). In this example, we named it `WorkspaceChooser`.

4. In Unity C# scripts, you can get these components by calling `InworldEditor.Root.Q()` by the name. e.g., `InworldEditor.Root.Q<VisualElement>("WorkspaceChooser")`.

**⚠ Note:** The file format is `uxml`, which is a similar version of `xml`. If you know how to code in `xml`, then you can modify it directly.

```

protected override void _SetupContents()

    DropdownField sceneChooser = SetupDropDown( tag: "SceneChooser");
    List<string> listScenes = InworldAI.Game.currentWorkspace.scenes.Select(scene => scene.ShortName).ToList();
    ActivateDropDown(ref sceneChooser, listScenes, OnSceneChanged);

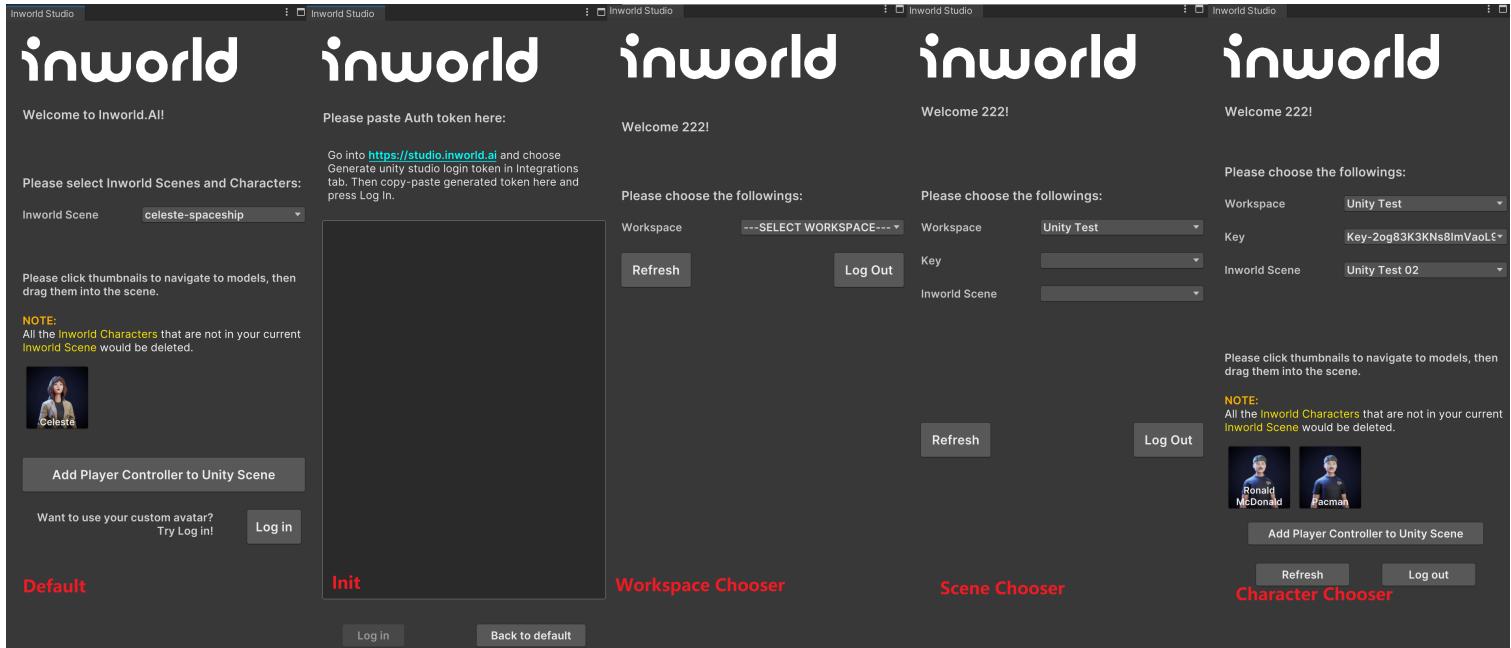
    m_LoadingBar = SetupProgressBar( tag: "LoadingBar", isVisible: false);
    m_CharacterChooser = InworldEditor.Root.Q<VisualElement>(name: "CharacterChooser");
    m_Instruction = InworldEditor.Root.Q<VisualElement>(name: "Instruction");
    m_Instruction.visible = false;

<ui:UXML xmlns:ui="UnityEngine.UIElements" xmlns:uve="UnityEngine.UIElements" xsi="http://www.w3.org/2001/XMLSchema-instance" engine="UnityEngine.UIEl ▲1 ✘ 7 ^ >
    <ui:ProgressBar title="Loading" name="LoadingBar" style="margin-left: 20px; margin-right: 20px; margin-top: 0; margin-bottom: 0;" />
    <ui:Label text="Please choose the followings:" display-tooltip-when-elided="true" name="ChooserTitle" style="height: 20px; font-size: 16px; -unity-text-align: center; margin-top: 20px; margin-bottom: 10px;" />
    <ui:DropdownField label="Workspace" index="-1" choices="System.Collections.Generic.List`1[System.String]" name="WorkspaceChooser" style="margin-top: 20px; margin-bottom: 10px;" />
    <ui:DropdownField label="Key" index="-1" choices="System.Collections.Generic.List`1[System.String]" name="KeyChooser" style="margin-top: 0; margin-left: 0; margin-bottom: 10px;" />
    <ui:DropdownField label="Inworld Scene" index="-1" choices="System.Collections.Generic.List`1[System.String]" name="SceneChooser" style="margin-top: 0; margin-left: 0; margin-bottom: 10px;" />
    <ui:Button text="Some data is missing. Please go to https://studio.inworld.ai/color and make sure there is at least 1 key and 1 workspace." style="background-color: #e0e0e0; border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;" />
    <ui:VisualElement name="Instruction">
        <ui:Label text="Please click thumbnails to navigate to models, then drag them into the scene." display-tooltip-when-elided="true" style="margin-left: 20px; margin-bottom: 10px;" />
        <ui:Label text="NOTE: All the Inworld Characters that are not in your current Inworld Scene would be deleted." display-tooltip-when-elided="true" style="margin-left: 20px; margin-top: 10px; font-size: 14px; color: yellow;" />
        <ui:Label text="All the Inworld Characters that are not in your current Inworld Scene would be deleted." style="margin-left: 20px; margin-top: 10px; color: yellow;" />
    </ui:VisualElement>
    <ui:VisualElement name="CharacterChooser" text="Whenever you are dragging avatar gameobjects into the scene, You need to make sure this color is yellow and it is initialized and implemented as below." style="background-color: #e0e0e0; border: 1px solid #ccc; padding: 5px; margin-top: 10px;" />
</ui:UXML>

```

# Finite-State Machine

The Inworld Studio Panel is implemented by a **finite-state machine**. Each page has its own state, and it is initialized and implemented as below.



## 1. Editor Default

This is the default page in editor mode for developers to use the default scenes and characters.

## 2. Editor Init

This is the page that developers can obtain their studio access tokens. This state is triggered if the user selects **Login** in the **EditorDefault**.

## 3. Editor Workspace Chooser

This state is triggered when the user token is pasted and the login is clicked in **EditorInit**, or when you select different **Workspaces** in other states. This state will fetch the user token from our server, and then list all the workspaces belonging to it.

## 4. Editor Scene Chooser

This state is triggered when one of the workspaces has been selected in the **EditorWorkspaceChooser**, or when different scenes are selected in **EditorCharacterChooser**. This state will fetch the API key and secret, and the scenes of that workspace. It will then provide a drop-down field for users to choose from.

## 5. Editor Character Chooser

This state is triggered when a workspace, API key and secret, and scene have been selected. In this state, it will start fetching thumbnails, avatars, and generating character buttons.

## 6. Editor Playing

This state is triggered under two conditions,

1. It is the default state for runtime
2. If the editor code has been changed when you modified code related to the **InworldEditor**

## 7. Editor Error

An error page is triggered when the default resource is incorrect or the token fails to be received. This triggers a back-off reconnecting system, and, if possible, it will switch to a previous state.

```

internal void Init()
{
    m_Studio = new InworldStudio(owner:this);
    if (m_States.Count != 0)
        return;
    m_States[InworldEditorStatus.AppPlaying] = new EditorPlaying();
    m_States[InworldEditorStatus.Default] = new EditorDefault();
    m_States[InworldEditorStatus.Init] = new EditorInit();
    m_States[InworldEditorStatus.WorkspaceChooser] = new EditorWorkspaceChooser();
    m_States[InworldEditorStatus.SceneChooser] = new EditorSceneChooser();
    m_States[InworldEditorStatus.CharacterChooser] = new EditorCharacterChooser();
    m_States[InworldEditorStatus.Error] = new EditorError();
    Status = InworldAI.User.EditorStatus;
}

public static InworldEditorStatus Status
{
    & Frequently called
    get => InworldAI.User.EditorStatus;
    & Frequently called
    set
    {
        InworldAI.User.EditorStatus = value;
        Instance.m_CurrentState?.OnExit();
        Instance.m_CurrentState = Instance.m_Status[InworldAI.User.EditorStatus];
        Instance.m_CurrentState?.OnEnter();
    }
}

void OnInspectorUpdate()
{
    // Call Repaint on OnInspectorUpdate as it repaints the windows
    // less times as if it was OnGUI/Update
    m_CurrentState?.PostUpdate();
    Repaint();
}

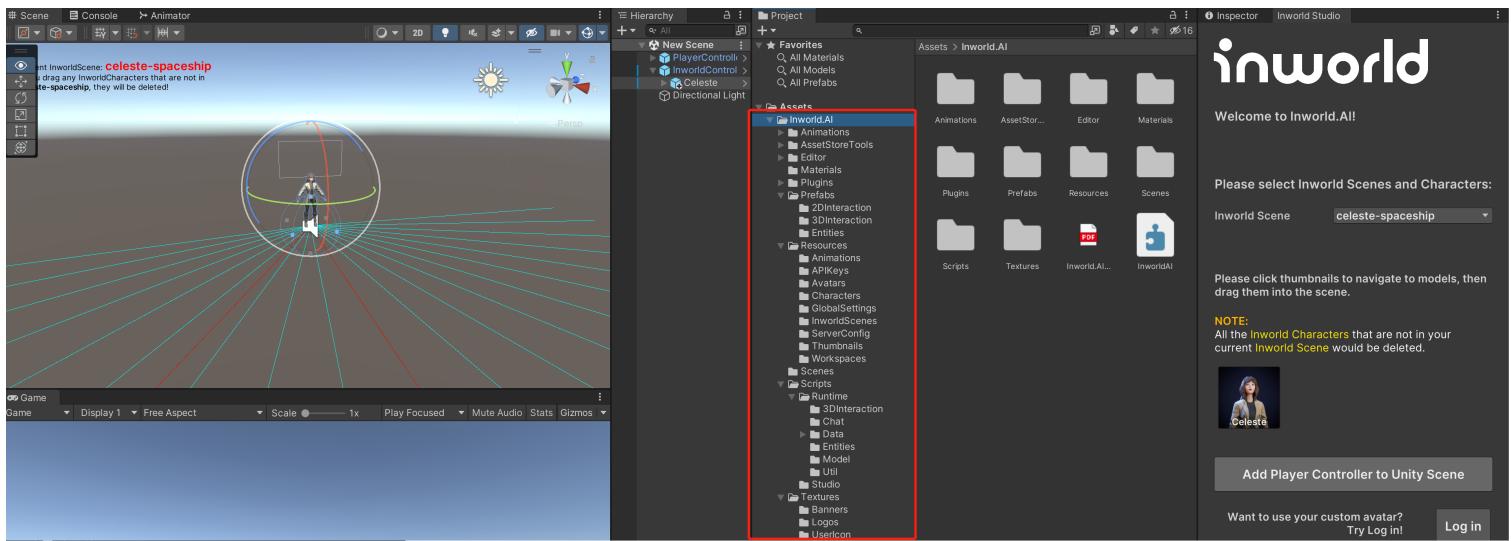
```

If you would like to create your own customized `Inworld Studio Panel`, then it is recommended that you create your own state. To do this, implement `OnEnter()`, `OnExit()`, `OnError()`, and `PostUpdate()` as needed.

Please visit **InworldEditor** for more references.

# package-resources

## Unity Package Structure



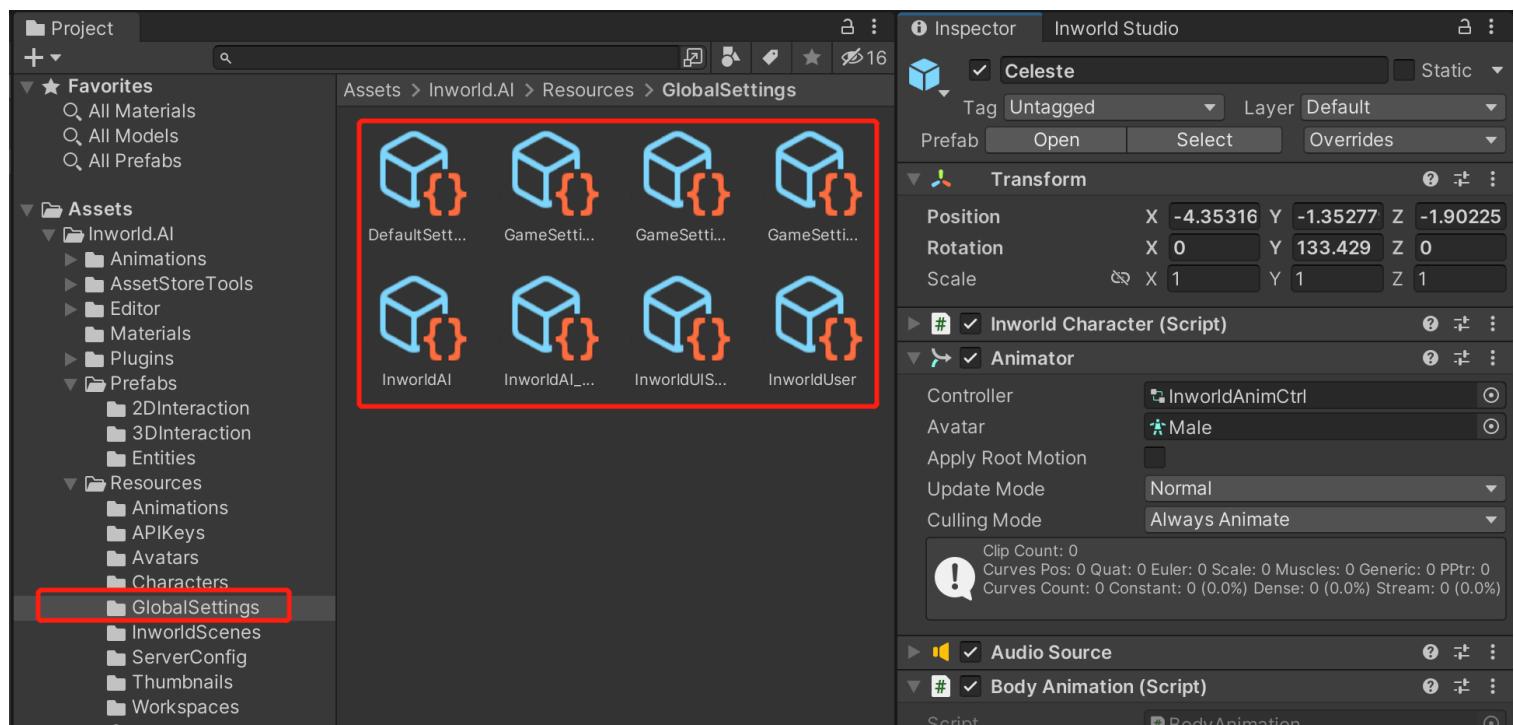
The Inworld AI Unity SDK asset package contains five major folders:

- **Animations/**: Contains all the animators, avatars, avatar masks, animation clips, etc.
  - **Avatar/**: Contains the humanoid avatars.
  - **BaseLayer/**: Contains all the animation clips in the base layer.
  - **Gesture/**: Contains all the animation clips in gestures.
- **AssetStoreTools/**: This part is used to integrate with the **Unity Asset Store**.
- **Editor/**: Contains all the UI elements, editor states, and global editor features.
  - **Layouts/**: Contains all the `uxmls`.
  - **States/**: Contains all the editor state implementations.
- **Materials/**: Contains all the material used in bubbles, samples, grounds, etc.
- **Plugins/**: Contains all native plugin files.
- **Prefabs/**: Contains all prefabs.
  - **2D Interactions/**: Contains the bubbles shown in the global chat panel.
  - **3D Interactions/**: Contains the bubbles shown in the world space, as well as the **PlayerController**.
  - **Entities/**: Contains the prefabs that are referenced in other scriptable object instances
- **Resources/**: Contains the data assets that are loaded at run-time. When you are using the **Inworld Studio Panel**, data will also be downloaded and generated in this folder. There is a separate folder named after your `userName`, which can be set in **Editor > Preferences > Inworld.AI**.

- **Animations/**: Contains the data for loading **Realistic Eye Movements**.
  - **APIKeys/**: Contains the API key for default workspaces.
  - **Avatars/**: Contains the default `.glb` models.
  - **Characters/**: Contains the default **InworldCharacterData**.
  - **GlobalSettings/**: Contains scriptable assets for all the settings in this SDK.
  - **InworldScenes/**: Contains the default **InworldSceneData**.
  - **ServerConfig/**: Contains information for our run-time and studio server.
  - **Thumbnails/**: Contains thumbnails for the default characters.
  - **Workspaces/**: Contains the default **InworldWorkspaceData**.
  - **Scenes/**: Contains the sample scene.
- **Scripts/**: Contains all related scripts.
    - **Runtime/**: Contains all the scripts that are specially used at run-time.
      - **3D Interactions/**: Contains audio and animation implementations.
      - **Chat/**: Contains all the chat implementations.
      - **Data/**: Contains all the scriptable objects.
        - **Settings/**: Contains all the scriptable objects instantiated in `Resources/Inworld.AI/GlobalSettings/`.
      - **Entities/**: Contains the **InworldController** and related files.
      - **Models/**: Contains model-related implementations.
      - **Util/**: Contains tools, enums, Unity events, etc.
    - **Studio/**: Contains all the files that are not used at run-time.
  - **Textures/**: Contains all the other texture resources.
    - **Banners/**: Contains all the banners of the Inworld title.
    - **Logos/**: Contains all the Inworld logos.
    - **UserIcon/**: Contains the bubbles, default thumbnails, etc.

# global-assets

## Global Assets



These global assets are stored as **ScriptableObjects**, saved in `Assets/Inworld.AI/Resources/GlobalSettings`. They are editable, but do not move or delete them.

ScriptableObject	Description
InworldAI	The master data object for storing settings. It can be visited by right-clicking and selecting <code>Inworld Setting Panel</code> in the <code>Project</code> tab, or at <code>Inworld &gt; Setting Panel</code> in the <code>TopMenu</code> .
GameSettings	This is the data object for storing current data, e.g., current workspace, current scene, or current key.
DefaultSettings	The default setting class for the path containing default assets or references. You can also visit it at <code>Edit &gt; Preferences &gt; Inworld.AI</code>
InworldUser	The class for storing user data. For public use, it supports changing <code>userName</code> and <code>companyUid</code> . It also stores references for locally downloaded assets.

ScriptableObject	Description
<b>InworldUISettings</b>	This scriptable object stores all the <code>uxml</code> that is used in the display <b>Inworld Studio Panel</b> .

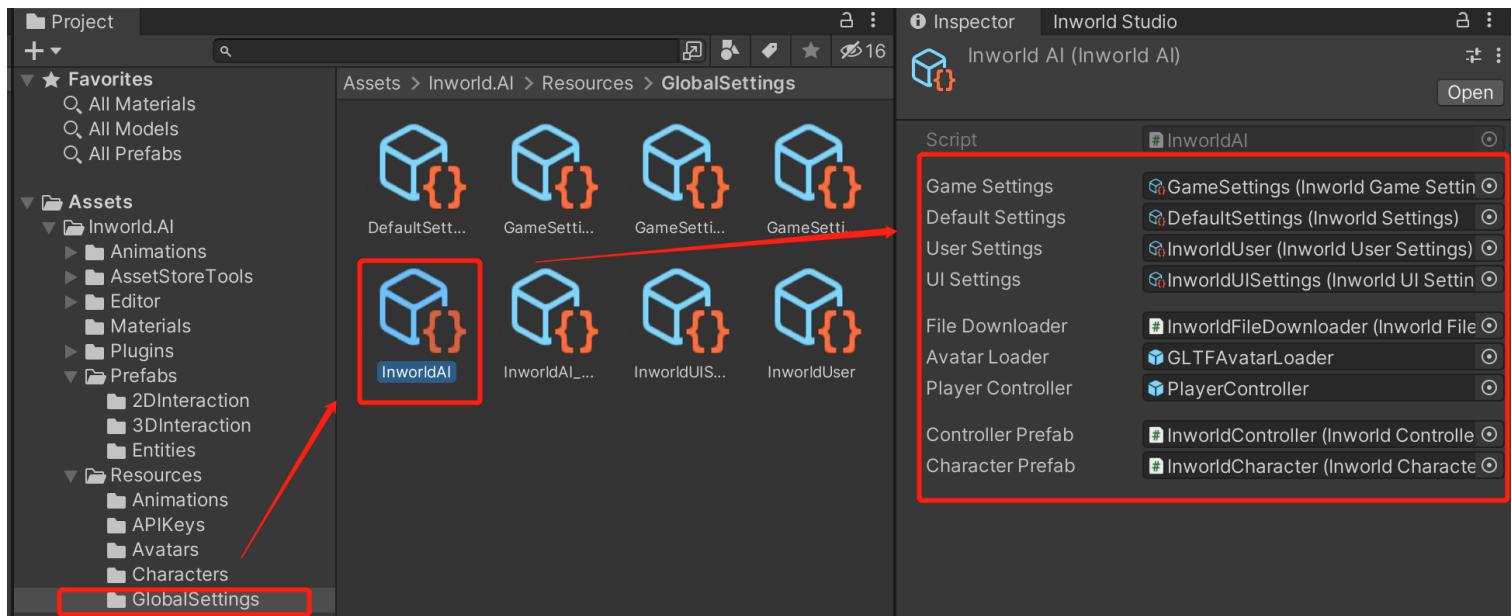
On this page



# InworldAI

## InworldAI

This is the master data object for storing settings. It is a singleton. Please ensure that there is only one copy of it, and that it is stored at the following location: `Assets/.../Resources/GlobalSettings/InworldAI.asset`.



## Inspector Variables

Variable	Description
Game Settings	The current <b>InworldGameSettings</b>
Default Settings	The current <b>InworldSettings</b>
User Settings	The current <b>InworldUserSettings</b>
UI Settings	The current <b>InworldUISettings</b>
File Downloader	The current <b>InworldFileDownloader</b>

Variable	Description
Avatar Loader	The current <b>AvatarLoader</b> , by default it is <b>GLTFAvatarLoader</b>
Player Controller	The current <b>PlayerController</b>
Controller Prefab	The current <b>InworldController</b> prefab for instantiate
Character Prefab	The current <b>InworldCharacter</b> prefab for instantiate

## Properties

Property	Type	Description
Instance	<b>InworldAI</b>	Gets an instance of <b>InworldAI</b> . By default, it is at <code>Inworld.AI/Resources/GlobalSettings/InworldAI.asset</code> . Please do not modify it.
File	<b>InworldFileDownloader</b>	Gets <b>InworldFileDownloader</b> scriptableObject. InworldFileDownloader is the downloader to fetch thumb or models of <b>InworldCharacterData</b> .
Game	<b>InworldGameSettings</b>	Gets <b>InworldGameSettings</b> scriptableObject. <b>InworldGameSettings</b> is the scriptableObject instance that always updates the current workspace, scene, and character data.
UI	<b>InworldUISettings</b>	Gets <b>InworldUISettings</b> scriptableObject. These are UI elements used in the <b>InworldEditor</b> .
Settings	<b>InworldSettings</b>	Gets DefaultSettings scriptableObject. This data can also be modified in <code>Edit &gt; Preferences &gt; Inworld.AI</code> .

Property	Type	Description
User	InworldUserSettings	Gets <b>InworldUserSettings</b> scriptableObject. This data can also be modified in <code>Edit &gt; Project Settings &gt; Inworld.AI</code> . Not only does it contain the <code>userName</code> and <code>companyName</code> , but it also stores the <code>userToken</code> , as well as the workspace, scene, character, model, and thumbnails are retrieved from the server
AvatarLoader	AvatarLoader	Gets the AvatarLoader prefab of the current package set. By default, Inworld uses <b>GLTFAvatarLoader</b> to load <b>Ready Player Me Avatars</b> . If you would like to use your own avatar loader please let your own Avatarloader and inherit with <b>IAvatarLoader</b> .
ControllerPrefab	InworldController	Gets the <b>InworldController</b> prefab. InworldController is a singleton object. If you want to use the Inworld feature, you should ensure that there is <b>one and only one InworldController</b> in your Unity scene.
CharacterPrefab	InworldCharacter	Gets the <b>InworldCharacter</b> prefab. Your <b>InworldCharacter</b> should contain an audio source, but it does not contain models. Usually, the <b>InworldCharacter</b> should be added and loaded models using the <b>GLTFAvatarLoader</b> .
PlayerControllerPrefab	PlayerController	Gets the <b>PlayerController</b> prefab. PlayerController is the object that contains the <code>mainCamera</code> and <code>SimpleCameraController</code> . It also contains a global chat panel.
IsDebugMode	bool	Gets if it is in <b>Debug Mode</b> . Debug Mode can be set in <code>File &gt; Preferences &gt; Inworld.AI &gt; IsVerboseLog</code> , or <code>Default Settings.asset</code> .

## API

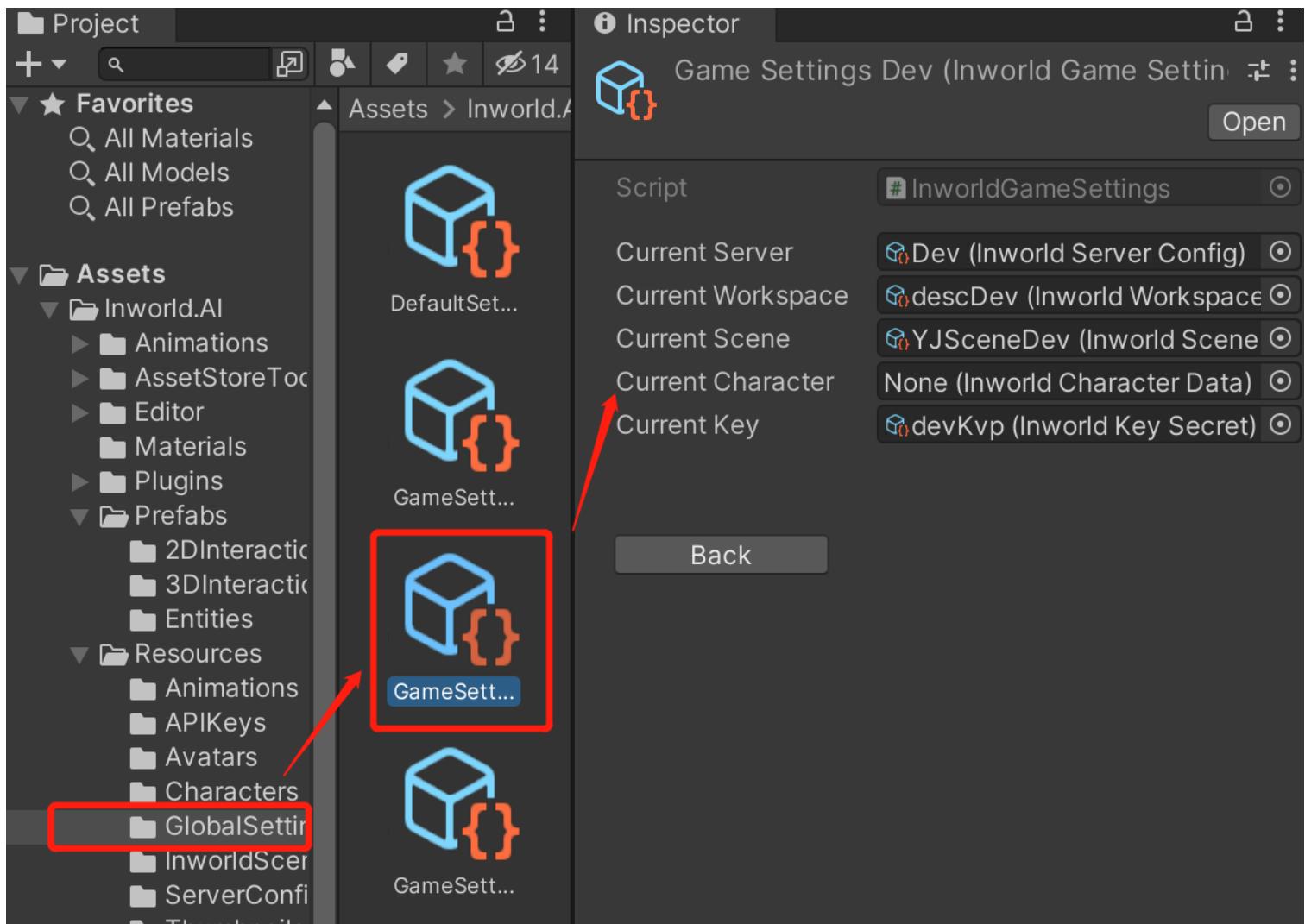
Function	Return Type	Description	Parameters
Log	void	Send <b>debug</b> log. If <code>IsVerboseLog</code> is checked, it will also be displayed in the console.	<b>log</b> : log to send
.LogWarning	void	Send <b>warning</b> log. If <code>IsVerboseLog</code> is checked, it will also be displayed in the console.	<b>log</b> : log to send
.LogError	void	Send <b>error</b> log. If <code>IsVerboseLog</code> is checked, it will also be displayed in the console.	<b>log</b> : log to send

On this page



# InworldGameSettings

The settings for run-time use are shown below.



This can have multiple copies, but the current one in use is set in **InworldAI**

## Inspector Variables

The following **inspector variables** are public,

Variable	Description

Variable	Description
currentServer	The current <b>InworldServerConfig</b>
currentWorkspace	The current <b>InworldWorkspaceData</b>
currentScene	The current <b>InworldSceneData</b>
currentCharacter	The current <b>InworldCharacterData</b>
currentKey	The current <b>InworldKeySecret</b>

## Properties

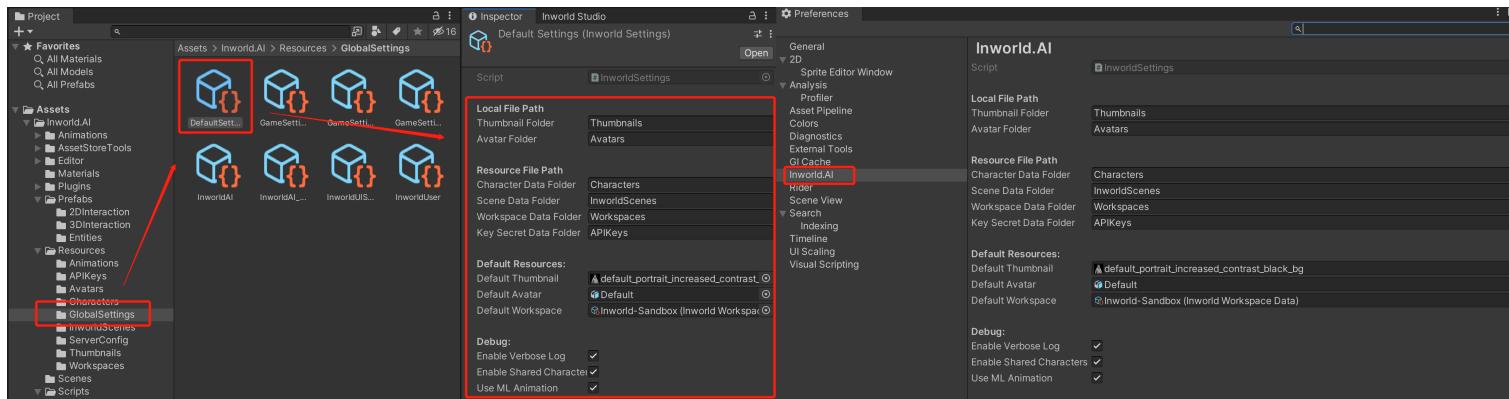
Property	Type	Description
RuntimeServer	string	Gets the acceptable URL Scheme for the run-time server. The format will be <code>https://xxx:port</code> . RuntimeServer is the server you communicate with when entering play mode.
StudioServer	string	Gets the acceptable URL Scheme for the run-time server. The format will be <code>https://xxx:port</code> . StudioServer is the server for fetching all kinds of data, including the run-time access token.
CurrentWorkspace	<b>InworldWorkspaceData</b>	Gets or sets the current Workspace.
KeySecret	<b>InworldKeySecret</b>	Gets or sets the current KeySecret pair.
APIKey	string	Gets or sets the current APIKey of the KeySecret pair.
APISecret	string	Gets or sets the current APISecret of the KeySecret pair.

On this page



# InworldSettings

This is the default setting class for the path of all the default assets or references. It can have multiple copies, but the actual active one is set in **InworldAI**. By default, the instance is `DefaultSettings`, and it can be configured by both `DefaultSettings.asset` or `Edit > Preferences > Inworld.AI`



## Inspector Variables

The following **inspector variables** are public:

Variable	Description
Thumbnail Folder	The current folder for thumbnails
Avatar Folder	The current folder for avatars
Character Data Folder	The current path for storing <b>InworldCharacterData</b>
Scene Data Folder	The current path for <b>InworldSceneData</b>
Workspace Data Folder	The current path for <b>InworldWorkspaceData</b>
KeySecret Data Folder	The current path for <b>InworldKeySecret</b>
Default Thumbnail	The default thumbnail

Variable	Description
Default Avatar	The default <code>.g1b</code> model
Default Workspace	The default <b>InworldWorkspaceData</b>
Enable VerboseLog	Set true if you would like to display <code>Verbose Log</code> in the console
EnableSharedCharacters	Set true if you would like to receive <code>InworldCharacters</code> from other accounts
Auto Generate Character	If it's true, whenever you drag a model with same InworldCharacter's name in current <b>InworldScene</b> , This SDK will generate an avatar and attach related scripts for you.

## Properties

Property	Type	Description
ThumbnailPath	string	Returns the thumbnail's path.
AvatarPath	string	Returns the avatar's path.
WorkspaceDataPath	string	Returns the workspace data path.
KeySecretDataPath	string	Returns the KeySecret data path.
InworldSceneDataPath	string	Returns the scene data path.
CharacterDataPath	string	Returns the character data path.
DefaultThumbnail	string	Returns the default thumbnail.
DefaultAvatar	string	Returns the default avatar.
DefaultWorkspace	string	Returns the default workspace data.
IsVerboseLog	bool	Returns if it is in Verbose Log mode.

Property	Type	Description
AutoGenerateCharacter	bool	Returns if avatar are allowed to be automatically composed in the Unity Scene.
EnableSharedCharacters	bool	Returns if sharing characters are enabled to be loaded.
Capabilities	Inworld.Grpc.CapabilitiesRequest	Returns the capabilities settings for communicating with our server.

On this page



# InworldUserSettings

This class stores all the data from the user who is using our package. This data includes the access token, and the corresponding workspace, scene, and character data fetched from the server.

## Inspector Variables

Variable	Description
User Name	The username belonging to the current user. It is generated by default using the Unity account name.
Organization ID	The current user's organization ID

## Properties

Property	Type	Description
Workspaces	Dictionary<string, <a href="#">InworldWorkspaceData</a> >	Returns the Inworld Workspace Data for the current user, indexed by Workspace MR Code.
InworldScenes	Dictionary<string, <a href="#">InworldSceneData</a> >	Returns the scene data for the current user, indexed by Scene MR Code.
Characters	Dictionary<string, <a href="#">InworldCharacterData</a> >	Returns the character data for the current user, indexed by Character MR Code.
Header	Grpc.Core.Metadata	Returns the header of the user. The header is used to send ListWorkspace, ListScene, and ListCharacter Request and is retrieved by the response of GetUserAccessToken.

Property	Type	Description
IsExpired	bool	Indicates whether or not the current token is expired.
Name	string	Gets or sets the username. This value can also be set from <a href="#">Edit &gt; Project Settings &gt; Inworld.AI</a>
Account	string	Get your registered account name in <a href="https://studio.inworld.ai">https://studio.inworld.ai</a>
OrganizationID	string	Gets or sets the user's organization ID. This value can also be set from <a href="#">Edit &gt; Project Settings &gt; Inworld.AI</a>
EditorStatus	enum	Gets or sets the editor's status according to this user. The reason it is saved in UserSettings is that whenever you reopened the Editor Application, the history page will be opened.
IDToken	string	Gets or sets the ID token. The ID token is used to get the studio access token.
RefreshToken	string	Gets or sets the Refresh Token. Refresh is used to get an ID token.
Request	Inworld.Grpc.UserRequest	Gets the User Request, which is to be sent to the server. Let the server know the username.

## API

Function	Description	Parameters
LoadData	LoadData is called whenever the Editor/Application is opened. It will load all the locally saved data by name, and set it to a user.	N/A
LogOut	Clears all the tokens and logs out of the Studio Server.	N/A

Function	Description	Parameters
RefreshTokens	Refresh the user's token	<b>strIDToken</b> : ID token to save. <b>strRefreshToken</b> : Refresh token to save.
OnLoggedInCompleted	Callback once the server has been logged in.	<b>inworldToken</b> : The inworld token received. <b>expireTime</b> : The inworld token's expiretime. It only lasts 1 hour. It should be refreshed frequently.

On this page



# InworldUISettings

This scriptable object stores all the `uxml` that is used to display the Inworld Studio Panel.

## Inspector Variables

Variable	Description
Header	The header section of <b>Inworld Studio Panel</b>
Workspace Chooser	The page for choosing workspace
Scene Chooser	The page for choosing <b>InworldSceneData</b> and <b>InworldKeySecret</b>
Character Chooser	The page for choosing characters
Application Playing	The page for Unity is in run-time mode
Default Content Panel	The content panel on the default page
Default Bot Panel	The button panel on the default page
InputField	The content panel on the Init page, for users to input tokens
Init Bot Panel	The bot panel on the Init page
WS Chooser Bot	The bot panel in choosing workspace
Connected Bot	The bot panel in choosing characters or scenes
Error Bot	The bot panel on the error page
Min Size	The minimum size of the <b>Inworld Studio Panel</b>

# Properties

Property	Type	Description
Header	VisualTreeAsset	Returns the header (Inworld Logo) page.
WorkspaceChooser	VisualTreeAsset	Returns the Inworld Workspace Chooser page.
SceneChooser	VisualTreeAsset	Returns the Inworld Scene Chooser page.
CharacterChooser	VisualTreeAsset	Returns the Inworld Character chooser page.
DefaultContentPanel	VisualTreeAsset	Returns the Content panel for the default page. (Default Workspace/ Celeste-spaceship and Olympus Theatre)
DefaultBotPanel	VisualTreeAsset	Returns the button panel for the default page.
InputField	VisualTreeAsset	Returns the content panel for Init (Inputting token).
InitBotPanel	VisualTreeAsset	Returns the buttons panel for Init.
WSChooserBot	VisualTreeAsset	Returns the buttons panel for the workspace chooser page.
ApplicationPlaying	VisualTreeAsset	Returns the "Actual" default page (showcasing Application is playing, or data needs to be refreshed) Because, at that time, the State is null.
RequesConnectedBotPanel	VisualTreeAsset	Returns the buttons panel for "InworldScene Chooser/Character chooser" page.
ErrorBotPanel	VisualTreeAsset	Returns the buttons panel for Error page. (Back to the Default page.)
MinSize	Vector2	Returns the minimal size of the Inworld Studio Panel. Otherwise, some data can not be displayed.

On this page

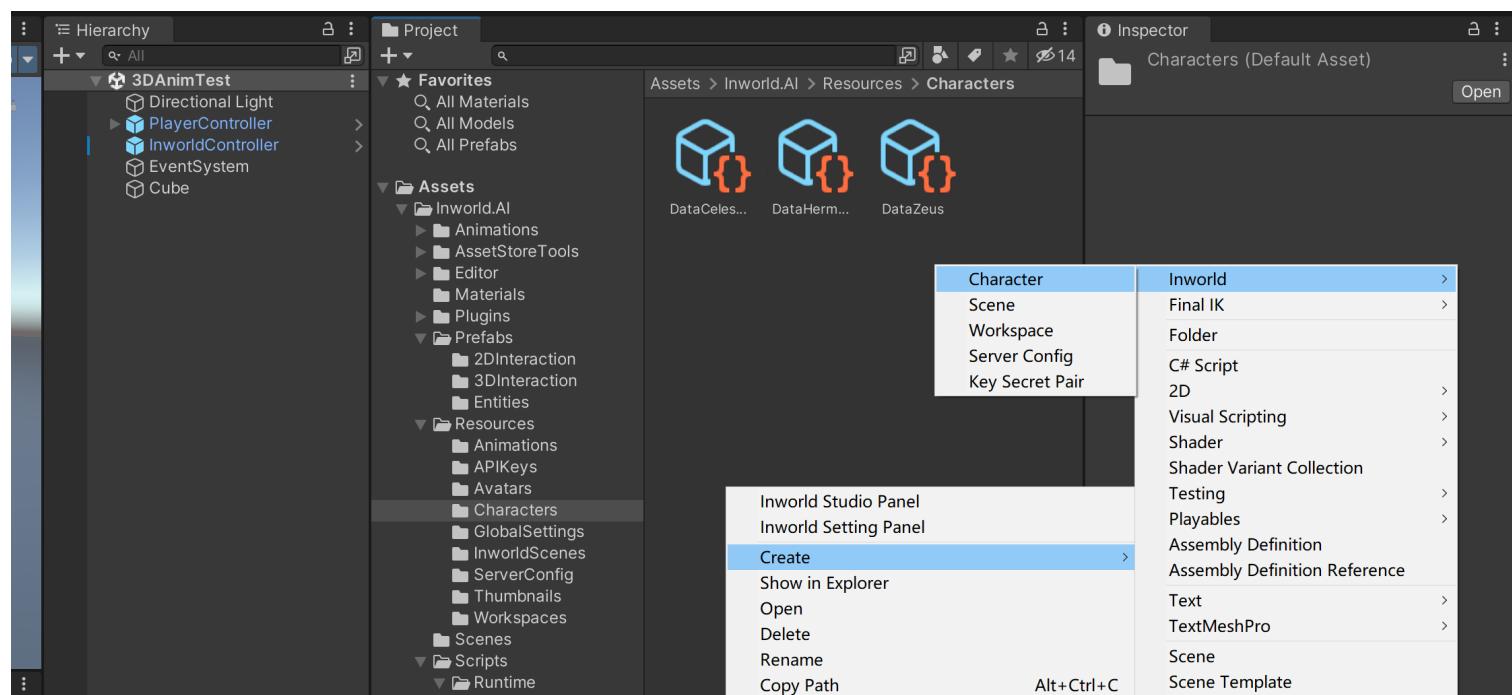


# scriptableobjects

## Scriptable Objects

### ScriptableObject

The following table is a summary of the data that is stored as a **ScriptableObject**. It can be generated by the user, or it can be generated and transferred via our internal logic. Create a **ScriptableObject** by right-clicking and selecting `Create > Inworld` in the `Project` tab.



ScriptableObject	Description
<b>InworldWorkspaceData</b>	This is the data for the workspace. It contains data from your <b>InworldScene</b> and <b>InworldCharacter</b>
<b>InworldSceneData</b>	This is the data for the scene. It contains data from your <b>InworldScene</b> and <b>InworldCharacter</b>

ScriptableObject	Description
<b>InworldCharacterData</b>	The is the data for your <b>InworldCharacter</b>
<b>InworldServerConfig</b>	This contains the related <b>URIs</b> for the server that your app is going to connect to
<b>InworldKeySecret</b>	This scriptable object is not used in our demo, but you can use it to pass along data for connecting

On this page

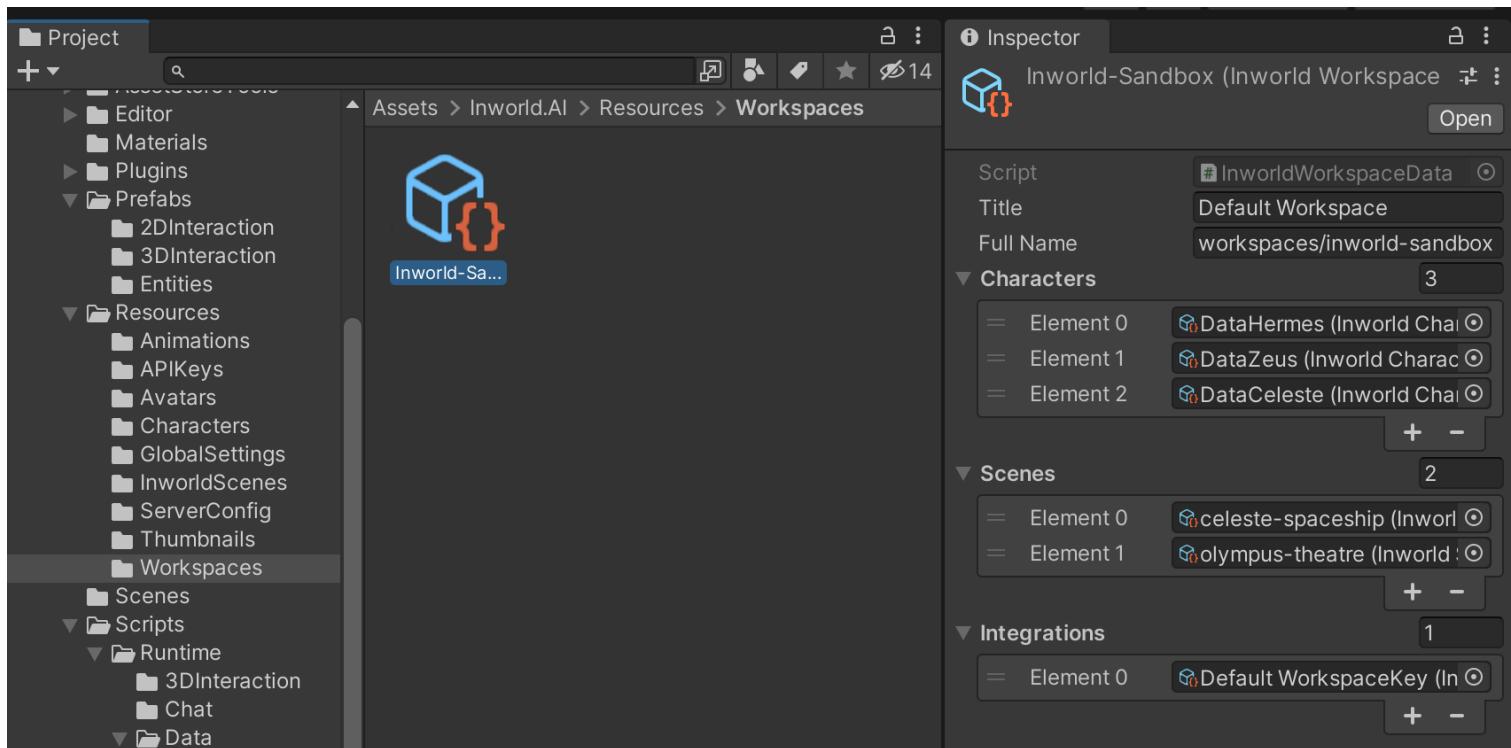


# InworldWorkspaceData

## Inworld Workspace Data

This `gameObject` is essentially the information data for an `Inworld Workspace`.

**⚠ Note:** File names are locally saved by `Title`, but that title is not necessarily unique. Please ensure that all of your `Title` fields are **unique** on your studio website. This will prevent any possible data collisions.



## Variables

Variable	Description
Title	The short name for an <code>Inworld Workspace</code>
Full Name	The MR Code for an <code>Inworld Workspace</code> .

Variable	Description
Characters	A list of <b>InworldCharacterData</b> corresponding to the characters associated with your workspace
Scenes	A list of <b>InworldSceneData</b> corresponding to the scenes associated with your workspace
Integrations	The <b>InworldKeySecret</b> corresponding to your Inworld Workspace

## Properties

Property	Type	Description
IsValid	bool	Checks if the workspace has correct data
DefaultKey	<b>InworldKeySecret</b>	Gets the default key for your workspace, if one exists

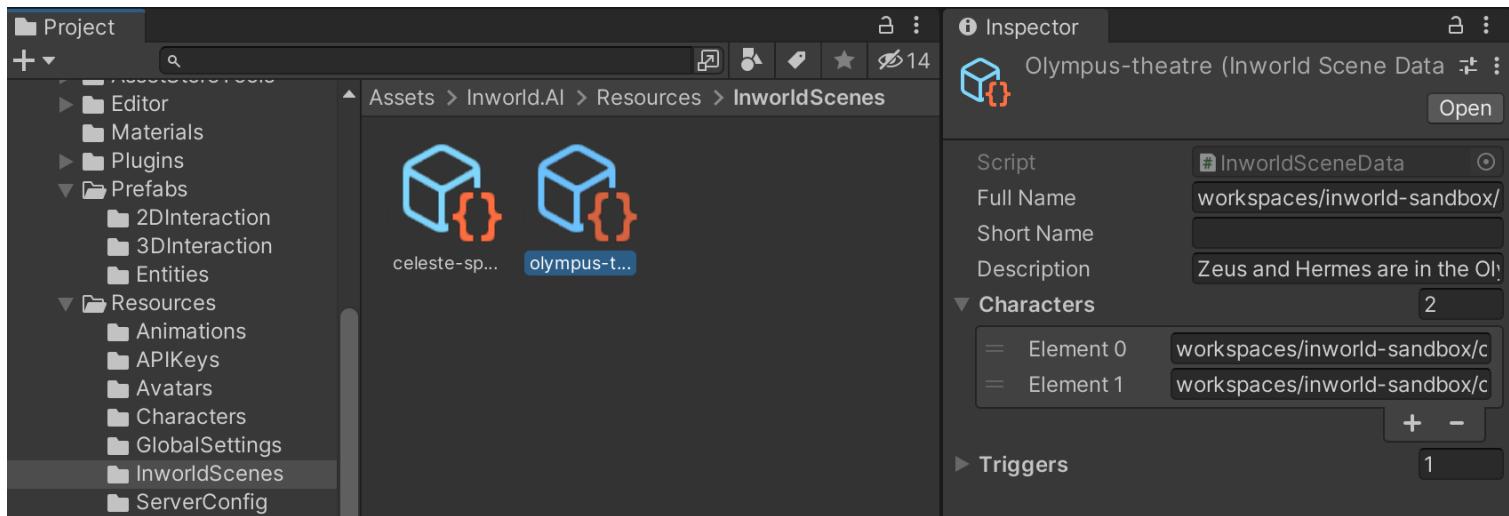
On this page



# InworldSceneData

## Inworld Scene Data

**InworldSceneData** is the `scriptableObject` for an **InworldScene**. While you can create it locally, it is intended to be downloaded from the server. An **InworldScene** is stored in the **InworldController**, and it contains characters. For each Unity Scene, you should prepare an **InworldController** containing an **InworldScene**.



## Inspector Variables

Variable	Description
Full Name	The MR Code for the <b>InworldScene</b>
Short Name	The short name for the <b>InworldScene</b> . This is also the file name that is saved locally at <code>Assets/Inworld.AI/Resources/</code>
Description	The <b>InworldScene</b> description
Characters	A list of <b>InworldCharacterData</b>
Triggers	A list of <b>triggers</b> for the scene

# Properties

Property	Description
ShortName	Gets the short name for the <b>InworldScene</b> . Please note that <b>ShortName</b> is also used as the stored file name in your resources. It is not uniquely retrievable from the server. Please ensure that your data of <code>ShortName</code> is unique to prevent data collision.

# API

Function	Return Type	Description	Parameters
CopyFrom	void	Copies the data from another <b>InworldScene</b> with the same <code>fullName</code> .	<code>rhs</code> : The reference for the <b>InworldSceneData</b> to copy

On this page



# FaceAnimData

## Face Anim Data

This scriptable object used to store the facial animation data.

## Inspector Variables

Variable	Description
emotions	The emotions data. Each Emotion contains a data set for blending shapes.
p2vMap	The p2v data for mapping phonemes to viseme indices.

On this page



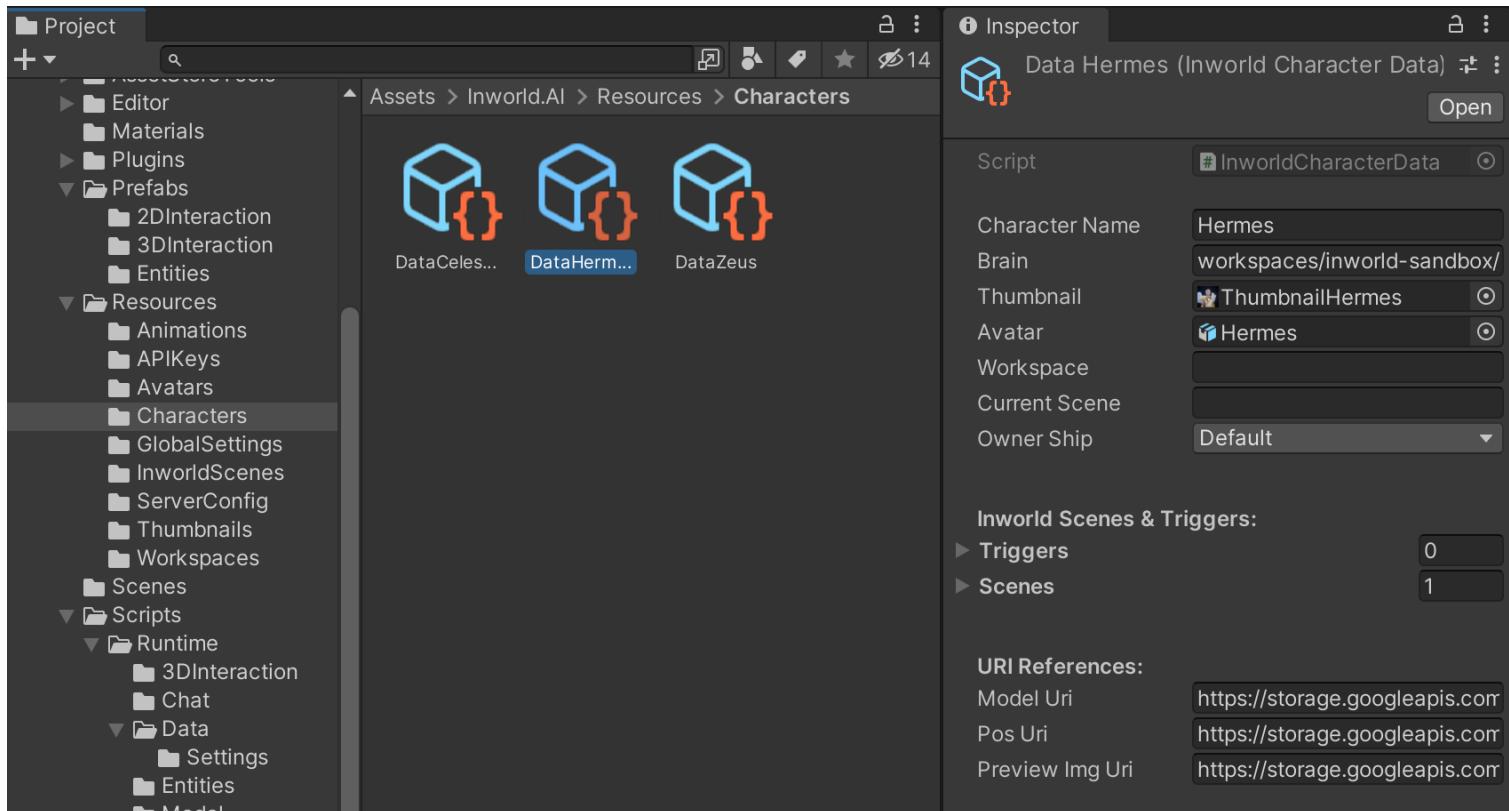
# InworldCharacterData

## Inworld Character Data

**InworldCharacterData** contains the the data for an **InworldCharacter**. It can be created locally, but it is mostly loaded and updated from the server. Note that,

1. The **Inworld Studio Server** can update everything except the **ID**
2. The **Inworld Run-Time Server** can update the **ID**

**⚠ Note:** The file containing **InworldCharacterData** is stored locally via **CharacterName**. The **CharacterName** field is not unique when retrieving data from the server. Please ensure that the **CharacterName** is set uniquely to prevent any possible data collisions.



## Public Variables

Public Variable	Description
Character Name	The display name
Brain	The MR Code
Thumbnail	The sprite, which, by default, is generated by <a href="#">Preview Image URI</a>
Avatar	The <code>.glb</code> model
Workspace	The <a href="#">Inworld Workspace</a> that a character is in
Current Scene	The current <b>InworldScene</b> that a character is in. This data will be refreshed at run-time
Owner Ship	The <b>enum</b> to distinguish detailed characters from characters that are fetched from the server
Triggers	The list of triggers that you set in the studio
Scenes	The list of scenes that the <b>InworldCharacter</b> is in
Model Uri	The URI of the <code>.glb</code> model from the <a href="#">Ready Player Me</a> avatar
PosUri	The URI of the pose picture
Preview Image Uri	The URI of the thumbnail

## Properties

Property	Type	Description
LocalThumbnailFileName	string	Gets the character's local thumbnail file name
LocalAvatarFileName	string	Gets the character's local avatar file name

Property	Type	Description
Thumbnail	Texture2D	Gets the character's thumbnail. If it contains data, then that data is returned directly. If it contains <b>previewImgUri</b> , then that is downloaded and loaded. Otherwise, this will return the default thumbnail
Avatar	<b>gameObject</b>	Gets the character's avatar in <b>.glb</b> format. If it has data, then that data is returned directly. If it contains <b>ModelUri</b> , then that is downloaded and loaded it. Otherwise, this will return the default avatar
Progress	float	Returns the data fetching progress for the character.

## API

Function	Return Type	Description	Parameters
CopyFrom	void	Copies the data from another <b>InworldCharacter</b> with the same <b>characterName</b> .	<b>rhs</b> : The reference for which <b>InworldCharacterData</b> to copy

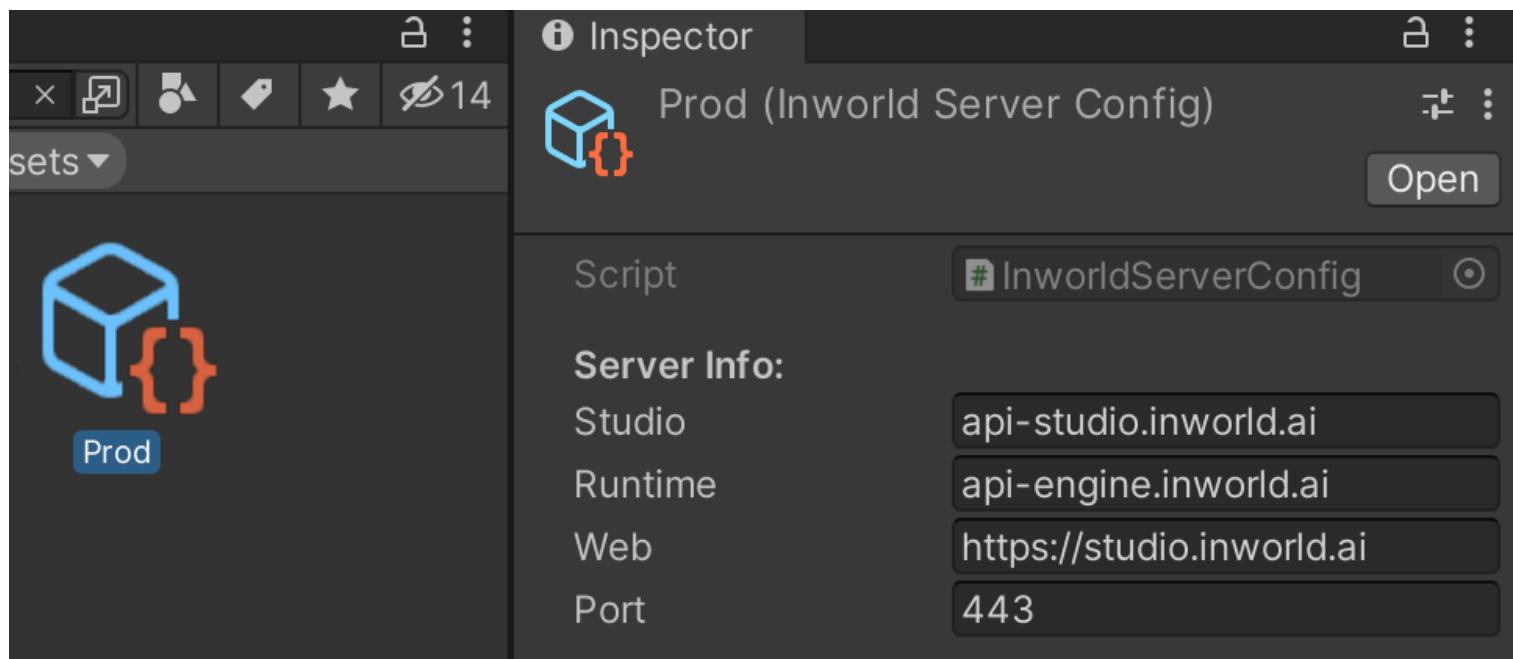
On this page



# InworldServerConfig

## Inworld Server Config

This class is used to save all kinds of URLs related to the the Inworld server.



## Variables

Variable	Description
Studio	The URI of the studio server
Run-time	The URI of the run-time server
Web	The URL of the studio website
Port	The IP port of the server

# Properties

Property	Type	Description
RuntimeServer	string	Returns the acceptable version of the run-time server
StudioServer	string	Returns the acceptable version of the studio server

On this page



# InworldKeySecret

## Inworld Key Secret

The data corresponding to an API Key and Secret. The Inworld Key and Inworld Secret is used to get a [SessionToken](#) at run-time.

## Variables

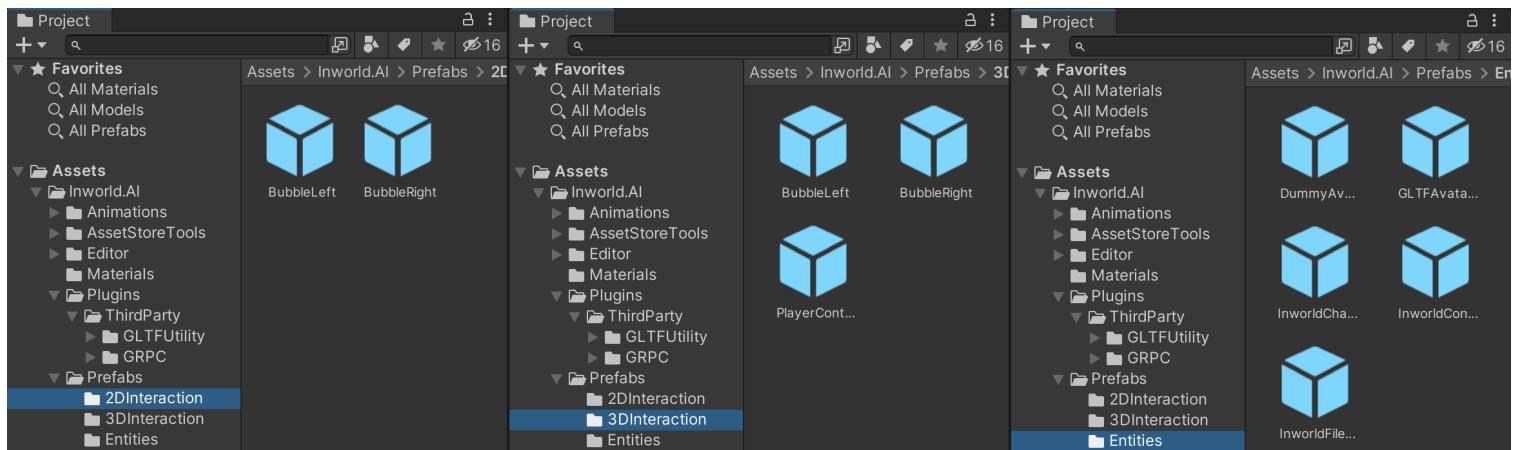
Variable	Description
Key	Used to store the API Key
Secret	Used to store the API Secret

## Properties

Property	Type	Description
ShortName	string	The file name that is stored locally in the resources folder

# Prefabs

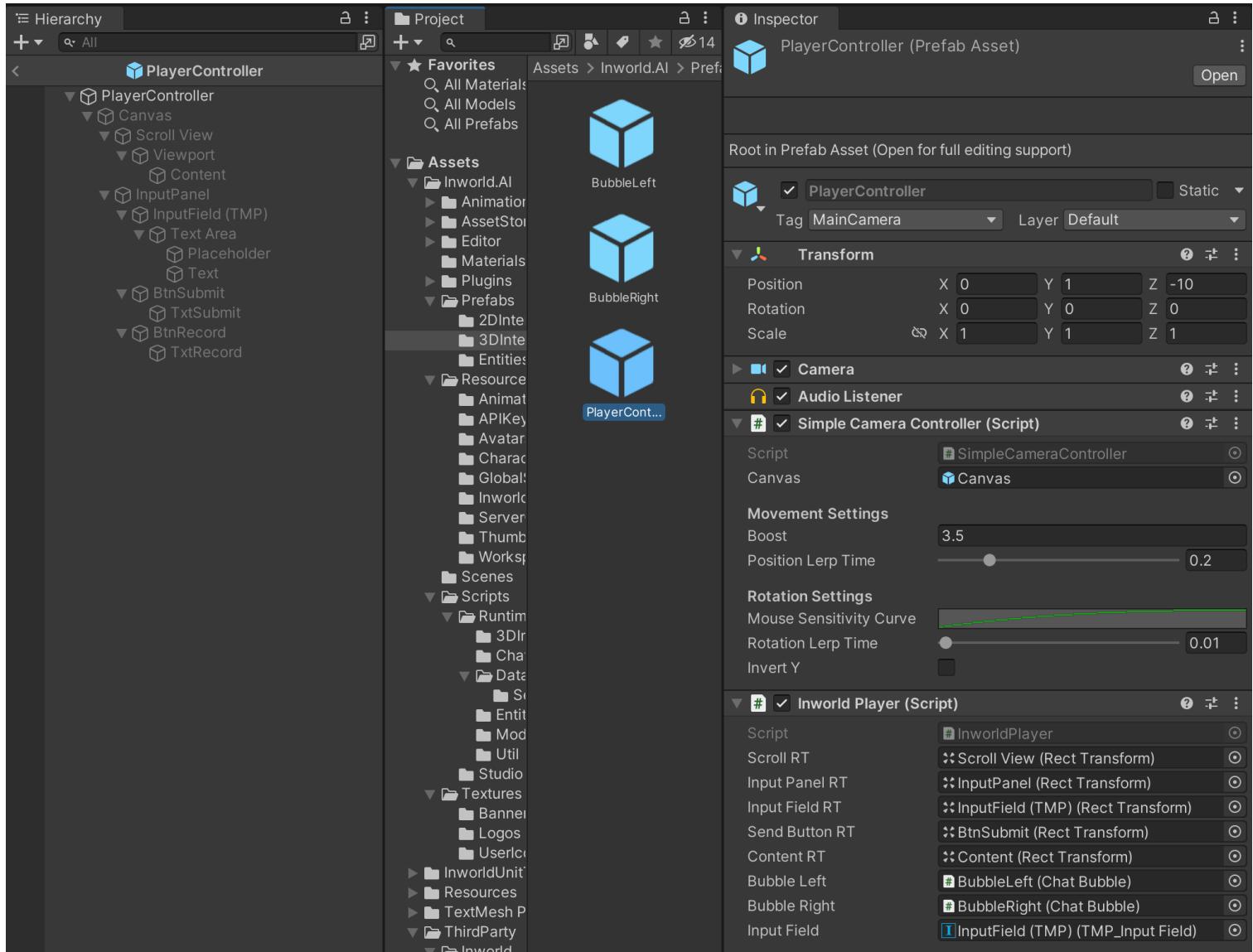
## Prefabs



Prefab	Description
PlayerController	The player controller interactable with <b>InworldCharacters</b> . It contains a <code>MainCamera</code> and <code>SimpleCameraController</code>
DummyAvatarLoader	If you do not want to load a 3D avatar, use this prefab to replace <code>GLTFAvatarLoader</code>
GLTFAvatarLoader	This uses <code>GLTFUtility</code> to load avatars
InworldFileDownloader	The player controller interactable with <b>InworldCharacters</b> . It contains a <code>MainCamera</code> and <code>SimpleCameraController</code>
InworldController	This is the main controller used at run-time. It is a singleton <code>gameObject</code> . Please ensure that there is only one <b>InworldController</b> in the scene
InworldCharacter	These are the AI character controller classes for players to communicate with. They are typically attached to the <code>.glb</code> model
Bubbles	These <code>BubblesLeft</code> and <code>BubblesRight</code> have two versions for the global chat panel and character's chat panel

# PlayerController

## Player Controller

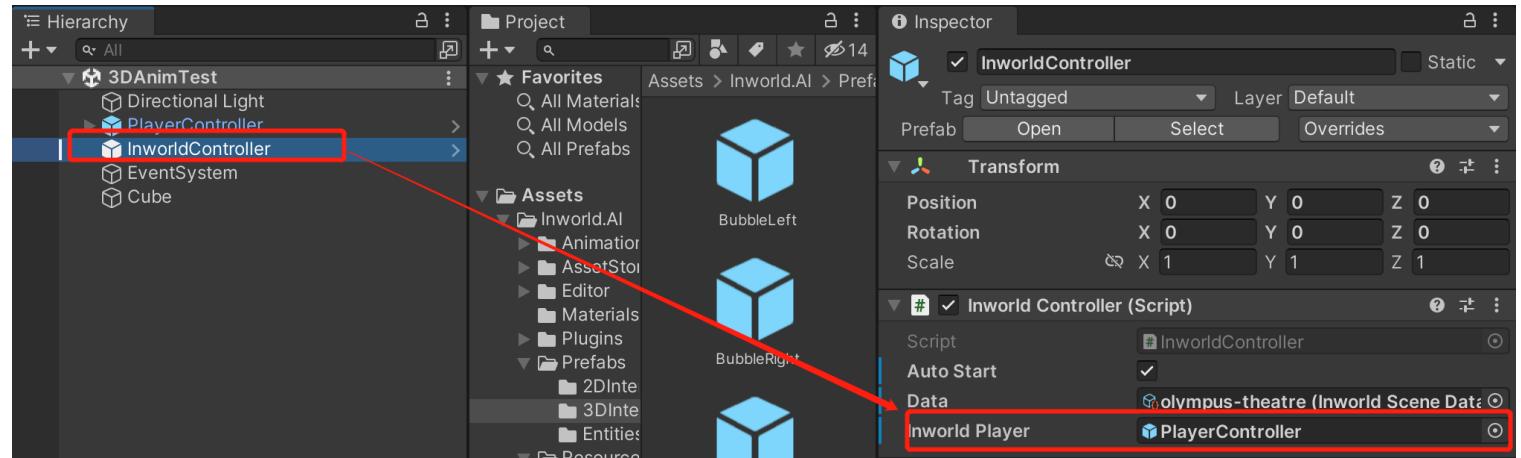


As mentioned, a **Player Controller** is a `gameObject` that contains the main camera, as well as a simple camera controller. It also contains a script called **InworldPlayer** for the global chat panel.

The **Player Controller** is also the `gameObject` for communicating with **InworldCharacters**.

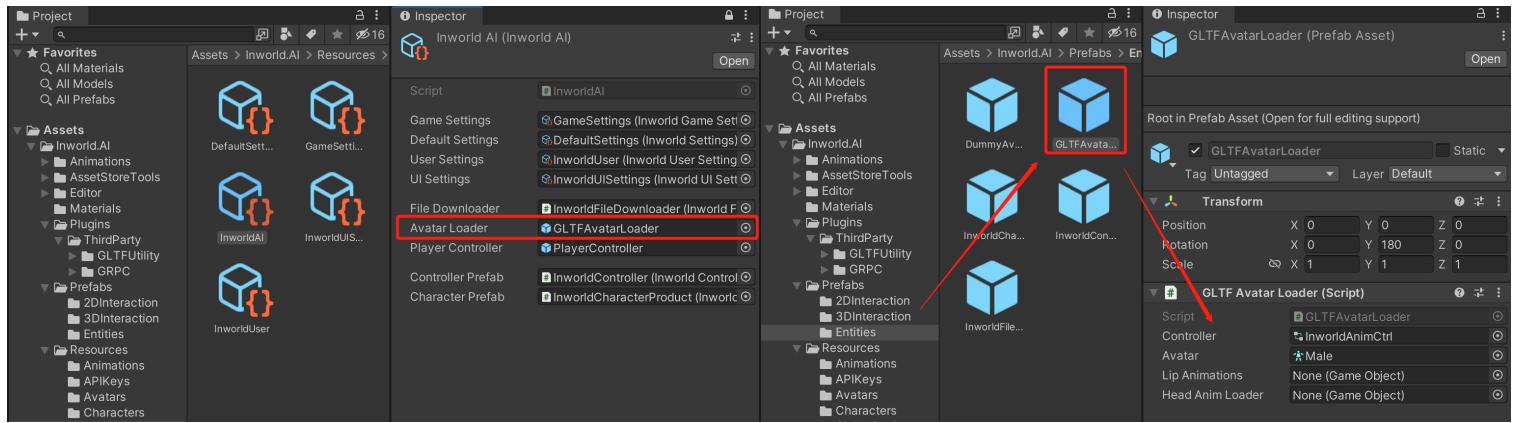
You can click `Add Player Controller to Scene` in the **Studio Panel** to delete the previous `mainCamera` in the scene.

If you want to implement your own **Player Controller**, then drag it to the **InworldPlayer** section of the **InworldController** tab. This is shown below.



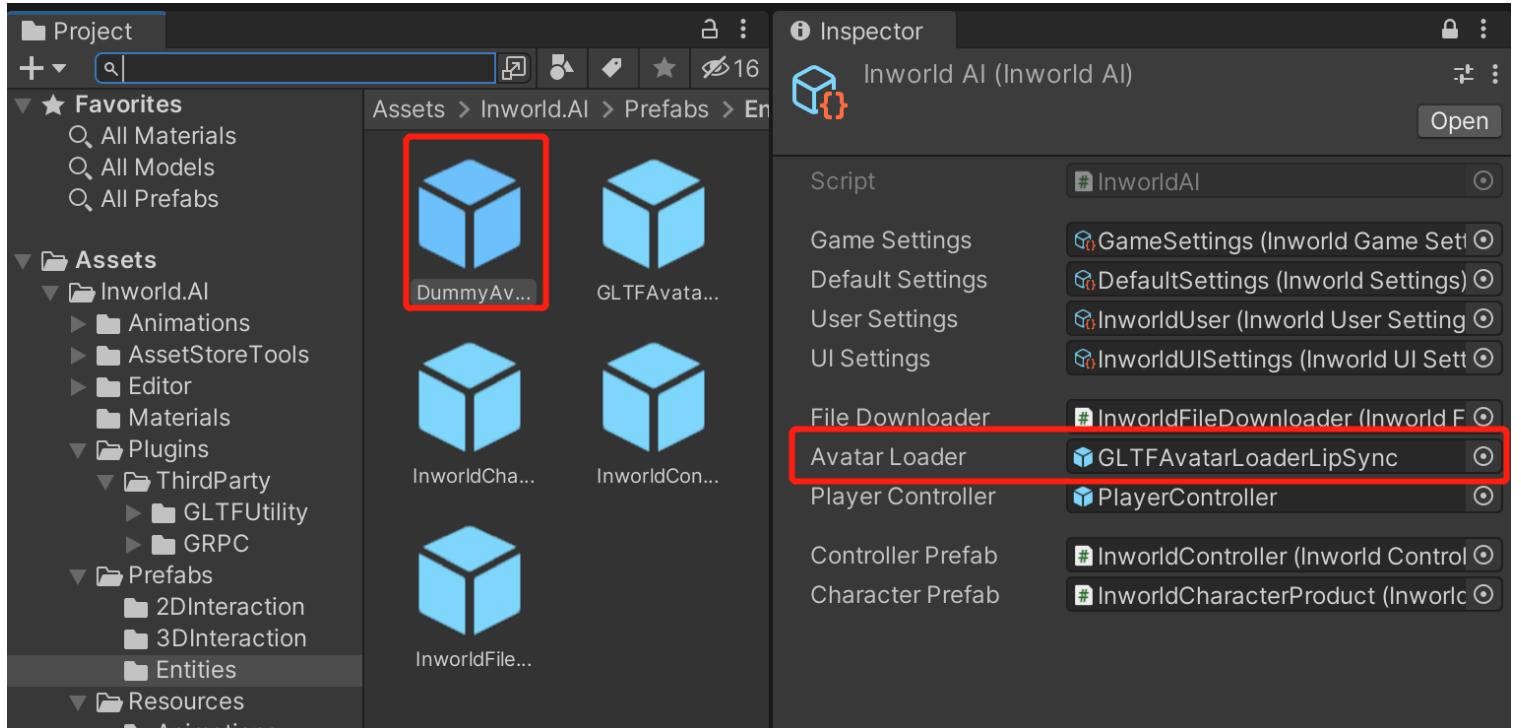
# GLTFAvatarLoader

## GLTFAvatar Loader



**GLTFAvatarLoader** is the default **IAvatarLoader** prefab in **InworldAI**. It supports loading **Ready Player Me** avatars, binding animations, and it has interfaces for lipsyncing and head-eye movement integrations.

If you would like to use your own implementation of 3D avatars, please create a prefab with a script inherited from **IAvatarLoader**. Replace **Avatar Loader** with your implementation.



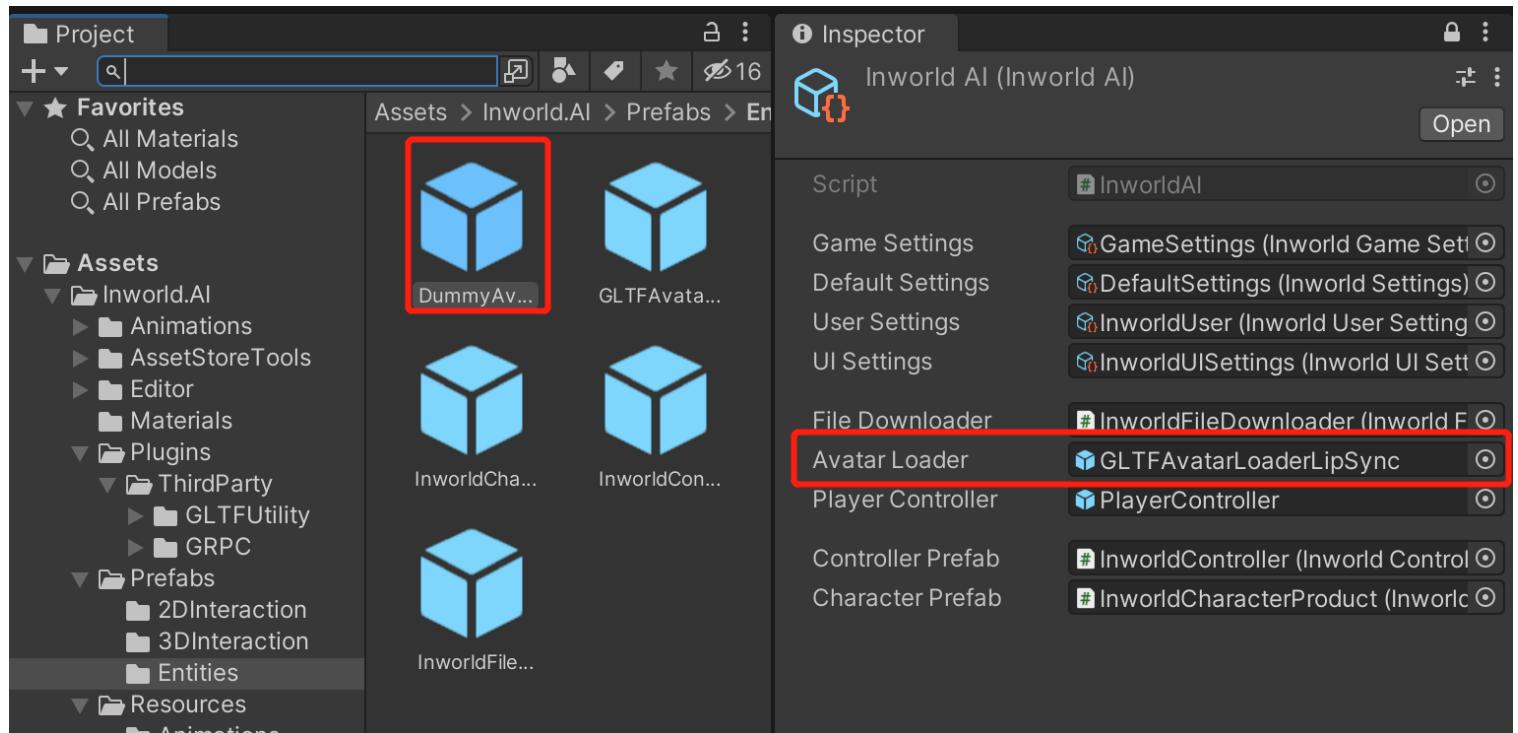
You can check [here](#) for more details.

# DummyAvatarLoader

## Dummy Avatar Loader

**DummyAvatarLoader** is a prefab that is not referenced in any scene or asset. If you do not want any 3D avatars, please replace **GLTFAvatarLoader** with the **DummyAvatarLoader**.

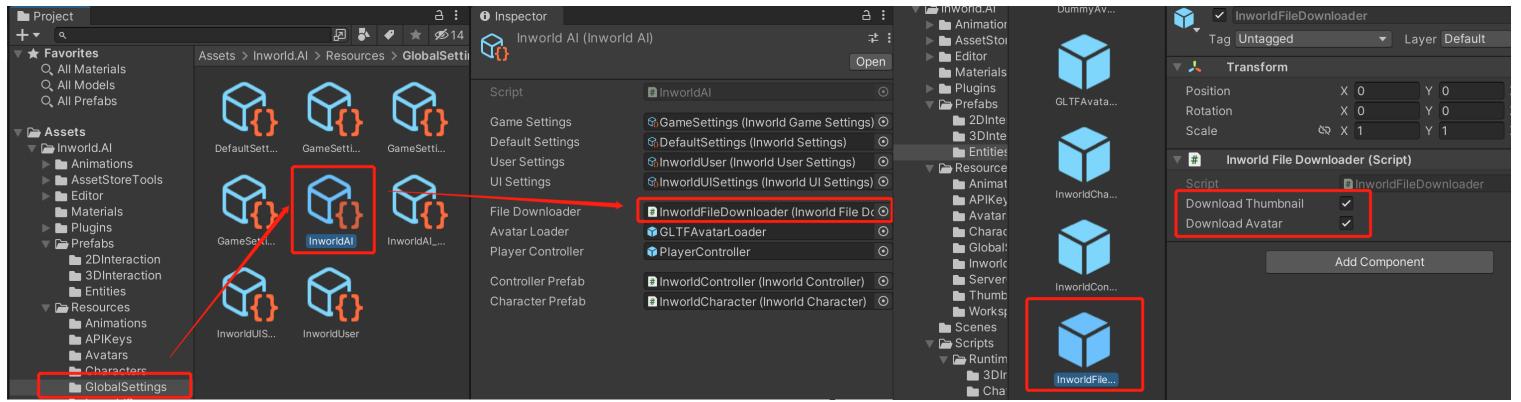
If you would like to use your own implementation of 3D avatars, then you will need to create a prefab with a script inherited from **IAvatarLoader**. Replace the **Avatar Loader** with your implementations.



You can check [here](#) for more details.

# InworldFileDownloader

## Inworld File Downloader



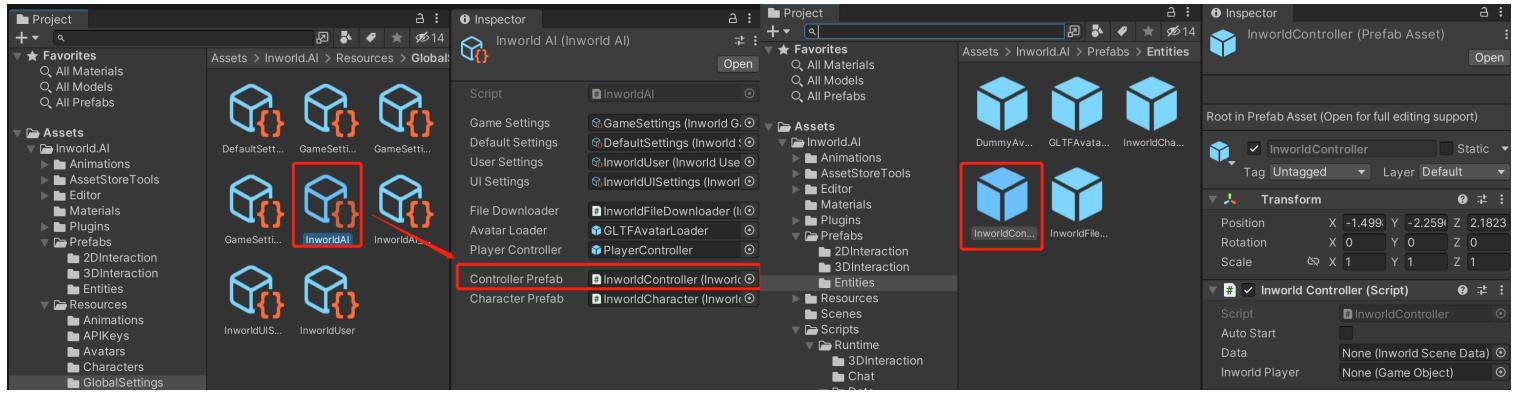
The **InworldFileDownloader** is the default downloader prefab in **InworldAI**. It supports downloading **Ready Player Me** thumbnails and avatars.

You can choose whether you want to download avatars, thumbnails or both.

If you would like to use your own implementation for downloading avatars, then create a prefab and replace the **File Downloader** with your implementation.

# InworldController

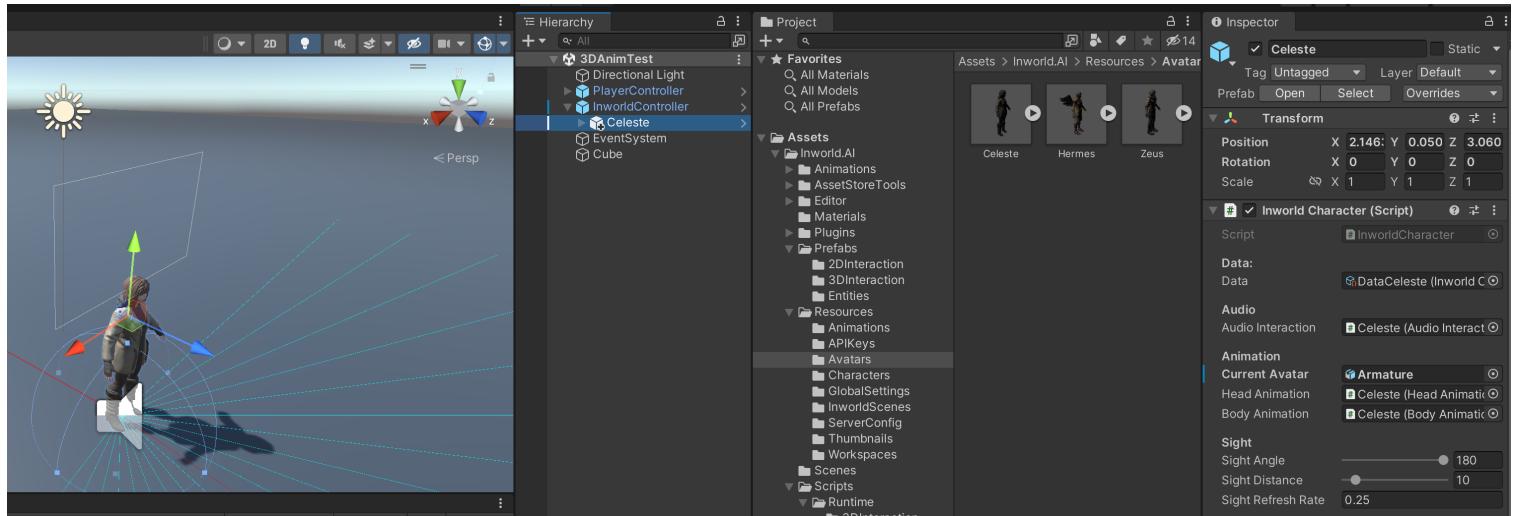
## Inworld Controller



The **InworldController** is the singleton object in a Unity run-time scene that loads **InworldSceneData** and an **InworldCharacter**, letting them communicate with the player. You can check [this](#) page for further information.

# InworldCharacter

## Inworld Character



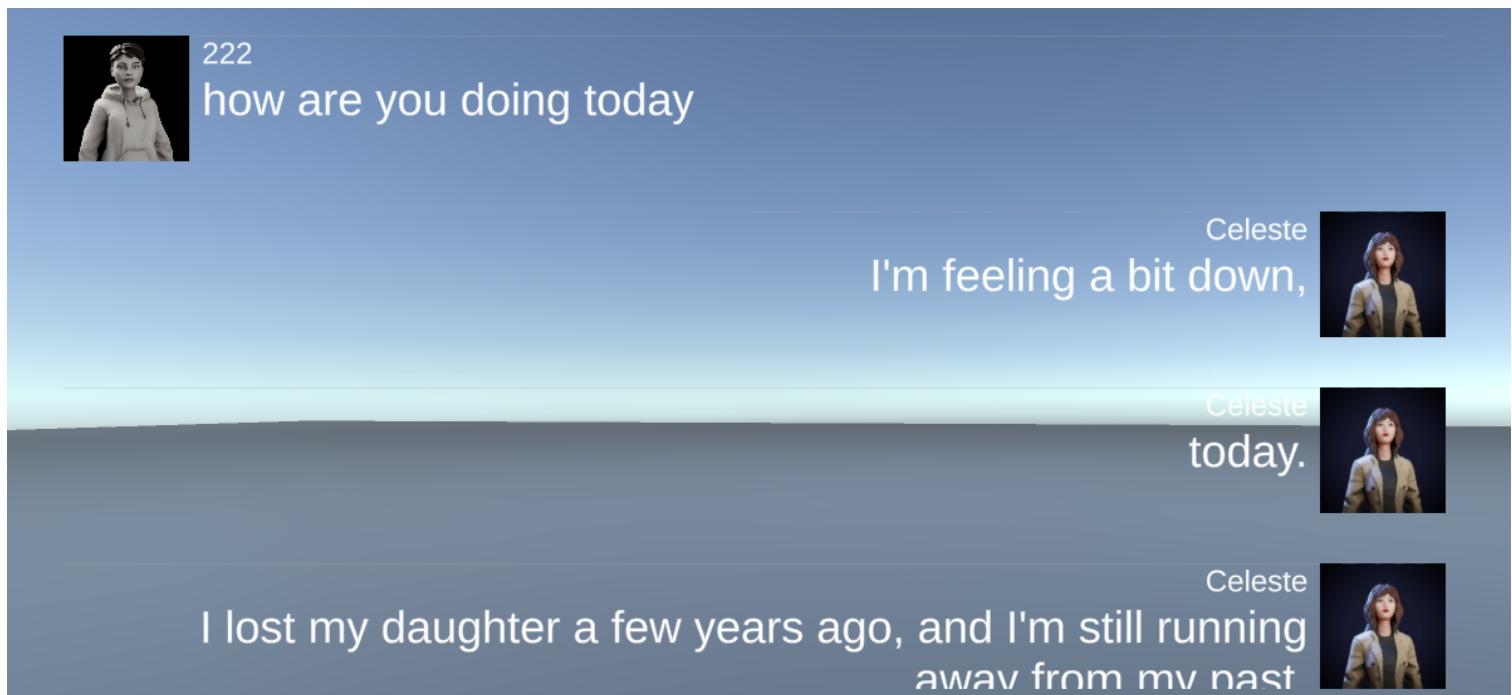
An **InworldCharacter** is the script that is attached to the `.glb` model once the model is dragged into the Unity Scene. It is the actual logic component for sending and receiving data to and from our run-time server. You can check [this](#) page for further information.

# Bubbles

## Bubbles

There are two kinds of chat bubbles:

**2D Bubbles** are used to display in the global chat panel.



**3D Bubbles** are used to display in the workspace.

222

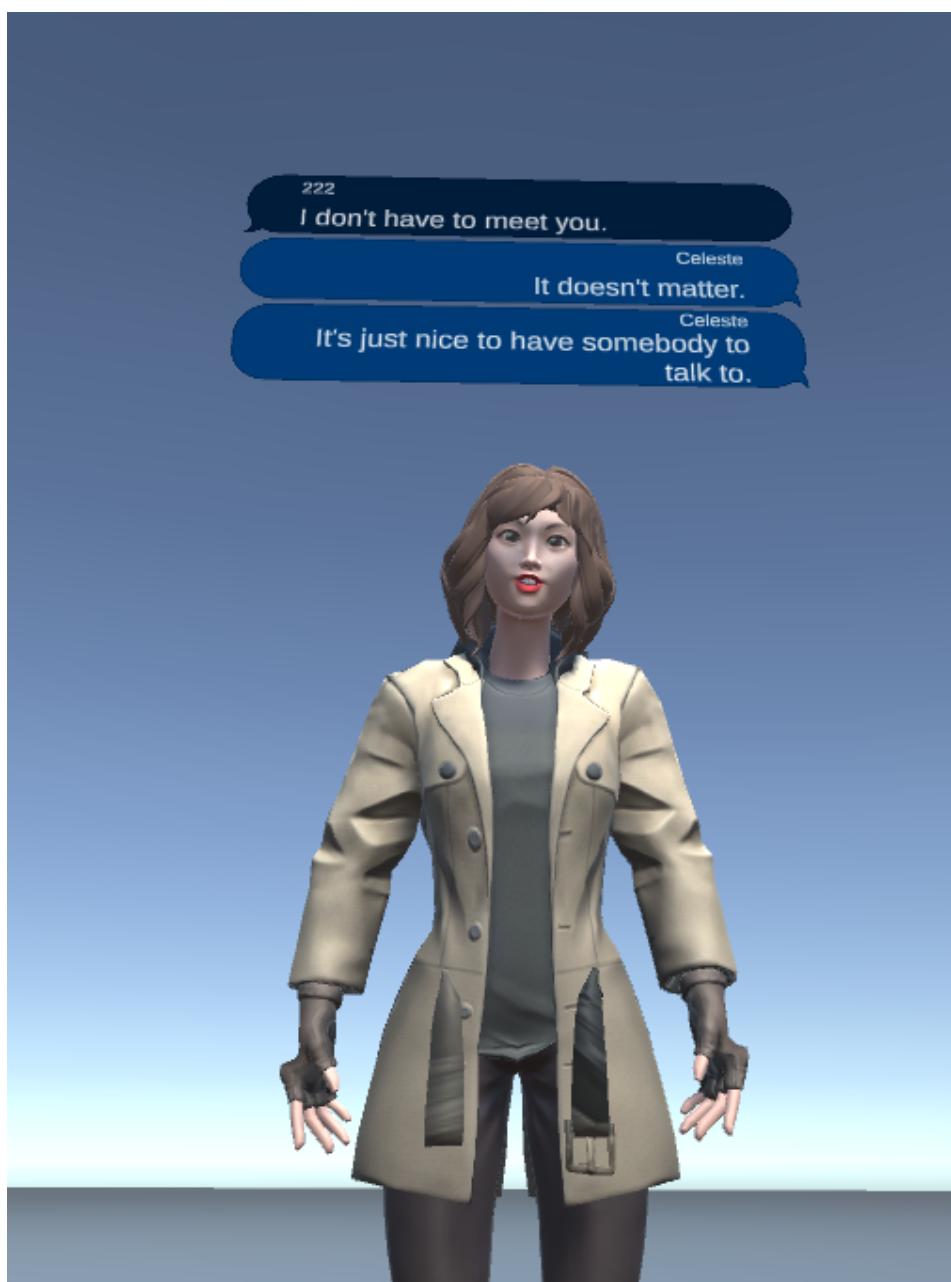
I don't have to meet you.

Celeste

It doesn't matter.

Celeste

It's just nice to have somebody to  
talk to.



Each kind of bubble has left and right directions for identifying messages sent by the player or the character. You can check [this](#) page for further information.



> tutorial-integrations

> Unity

> package-resources

> api-references

# api-references

## API References

Please review the following pages for a summary of our **API References**.

On this page



# AudioCapture

## AudioCapture

This is a global Audio Capture controller. For each separate **InworldCharacter**, we use class **AudioInteraction** to handle **AudioClips**.

## Inspector Variables

Variable	Description
Audio Rate	The sample rate of Audio Clip
Buffer Seconds	How long in seconds to record and generate a sample that would be sent to server.

## Properties

Property	Type	Description
IsCapturing	Bool	Get/Set the microphone to record voices.

## API

Function	Return Type	Description
StartRecording	void	Let microphone to start recording
StopRecordidng	void	Let microphone to stop recording

On this page



# AudioInteraction

## AudioInteraction

This component is used to send and receive audio from the server.

## Inspector Variables

Variable	Description
Character	The <b>InworldCharacter</b> that it is attached to
Playback Source	The <b> AudioSource</b> that it is referred to

## Properties

Property	Type	Description
Character	<b>InworldCharacter</b>	Gets or sets an attached Inworld Character
PlaybackSource	<b> AudioSource</b>	Gets or sets the audio source for playback
IsPlaying	<b> Bool</b>	Get if the playback source is playing
CurrentChunk	<b> AudioChunk</b>	Get the current playing audio chunk

## Events

Event	Type	Description

Event	Type	Description
OnAudioStarted	Action<PacketId>	Triggered when a piece of audio source started
OnAudioFinished	Action<InworldPacket>	Triggered when a piece of audio source finished playing

## API

Function	Return Type	Description	Parameters
Clear	void	Call this function to clean up the cached queue	N/A

On this page



# BodyAnimation

## BodyAnimation

This component is used to receive gesture events and/or emotion events from the server. It plays animations on that character.

## Properties

Property	Type	Description
Animator	Animator	Gets or sets the animator this component is attached to
Character	InworldCharacter	Gets or sets the character this component uses

## API

Function	Return Type	Description	Parameters
HandleMainStatus	void	Handle the main status of the character: <code>Idle</code> , <code>Talking</code> , <code>Walking</code> , etc.	<b>status:</b> incoming status
HandleEmotion	void	Plays an animation according to the target emotion	<b>SpaffCode:</b> An enum of emotion
HandleGesture	void	Plays the target gesture's animations	<b>gesture:</b> An enum of target gesture

On this page



# ChatBubble

## Chat Bubble

This class is for each detailed chat bubble.

## Inspector Variables

Variable	Description
Text Field	The text component of the bubble
Icon	The thumbnail in the bubble
Character Name	The name of the speaker

## Properties

Property	Type	Description
Text	string	Gets or sets the bubble's main content
Height	float	Gets the bubble's height
CharacterName	string	Gets or sets the speaker's name
Icon	RawImage	Gets or sets the thumbnail of the bubble

## API

Function	Return Type	Description	Parameters
SetBubble	void	Set the bubble's property	<b>charName:</b> The name of the owner of the bubble <b>thumbnail:</b> The thumbnail of the owner of the bubble <b>text:</b> The content of the bubble

On this page



# ChatPanel3D

## 3D Chat Bubble

This class is used to show and hide the characters' floating text bubbles.

## Inspector Variables

Variable	Description
Left Bubble	<b>ChatBubble</b> aligned on the left
Right Bubble	<b>ChatBubble</b> aligned on the right
Panel Anchor	The anchor for displaying bubbles
Owner	The <b>InworldCharacter</b> who owns the bubbles

On this page



# DemoCanvas

## Demo Canvas

This root class is for the UI Component in sample scenes.

## Inspector Variables

Variable	Description
Title	The title showcased on the monitor
Content	The detailed contents

On this page



# DummyAvatarLoader

## DummyAvatarLoader

### Dummy Avatar Loader

If you do not want to have a 3D model, drag the **Dummy Avatar Loader Prefab** into the "Avatar Loader" field.

## Events

Event	Type	Description
AvatarLoaded	Action< <b>InworldCharacter</b> >	Triggered immediately in <code>ConfigureModel()</code>

## API

Function	Return Type	Description	Parameters
ConfigureModel	void	Implements interface functions by invoking <code>AvatarLoaded</code>	<b>character:</b> the <b>InworldCharacterData</b> to load, <b>model:</b> the model to attach
Import	IEnumerator	Implements interface functions	<b>url:</b> a string of URLs to download avatars
LoadData	<code>gameObject</code>	Implements interface functions	<b>content:</b> data to load

Function	Return Type	Description	Parameters
LoadData	gameObject	Implements interface functions	<b>fileName:</b> file to load

On this page



# EmotionCanvas

## Emotion Canvas

This class is for the UI Component in sample scenes.

## Properties

Property	Type	Description
Emotion	SpaffCode	The current Emotion of the character
Gesture	GestureType	The current Gesture of the character

## API

Function	Return Type	Description	Parameters
SendEmotion	void	Set the emotion of the character	<b>emotion:</b> the emotion to send
SendGesture	void	Set the gesture of the character	<b>gesture:</b> the gesture to send
SetMainStatus	void	Set the main status of the character	<b>mainStatus:</b> the main status to send

On this page



# GLTFAvatarLoader

## GLTFAvatarLoader

This is the default avatar loader. Use **GLTFUtility** to download and import **Ready Player Me** data, such as `.glb` files.

## Inspector Variables

Variable	Description
Controller	The run-time animation controller that would be installed on the avatar.
Avatar	The avatar being processed
Lip Animations	The prefab for lip syncing, which by default is <code>null</code> . It can be added by following <b>these steps</b> .
Head Anim Loader	The prefab for loading head and eye animations, which by default is <code>null</code> . It can be added by following <b>these steps</b> .

## Events

Event	Type	Description
AvatarLoaded	Action< <b>InworldCharacter</b> >	Triggered once <code>ConfigureModel()</code> is completed

## API

Function	Return Type	Description	Parameters
ConfigureModel	void	bind <b>InworldCharacterData</b> to target <code>.glb</code> model	<b>character</b> : the <b>InworldCharacterData</b> to load, <b>model</b> : the model to attach to
Import	IEnumerator	Implements interface functions	<b>url</b> : a string of URLs to download avatars.
LoadData	<code>gameObject</code>	Loads data from memory (bytestring) to generate a <code>gameObject</code>	<b>content</b> : data to load
LoadData	<code>gameObject</code>	Loads data from file to generate a <code>gameObject</code>	<b>fileName</b> : file to load

On this page



# HeadAnimation

## Head Animation

This class is the basic class for displaying head animations. It currently supports looking at players, and display facial expressions for emotions.

If you want to use detailed head-eye movements, please do the following:

1. Purchase and download page [Realistic Eye Movements](#)
2. Add `LookTargetController` and `EyeAndHeadAnimator` components to your `InworldCharacters`.
3. Implement `SetupHeadMovement` by, a. Calling `Resources.Load<TextAsset>(m_HeadEyeAsset);`  
b. Calling `EyeAndHeadAnimator::ImportFromJson()`, with the data of the `TextAsset` that you loaded

## Inspector Variables

Variable	Description
Head Eye Asset	The JSON file name for loading head eye movements
Face Data	The scriptable object that stores morphed data
Morph Time	how long in seconds will the character do the emotion transitions

## Properties

Property	Type	Description
Animator	Animator	Gets or sets the animator this component is attached to
Character	<code>InworldCharacter</code>	Gets or sets the character this component uses

# API

Function	Return Type	Description	Parameters
HandleMainStatus	void	Handle the main status of the character: <code>Idle</code> , <code>Talking</code> , <code>Walking</code> , etc.	<b>status</b> : incoming status
HandleEmotion	void	Plays an animation according to the target emotion	<b>SpaffCode</b> : An enum for emotion
HandleGesture	void	Plays the target gesture's animations	<b>gesture</b> : An enum for target gestures

On this page



# HistoryItem

## HistoryItem

**HistoryItem** is the data class received from the server.

## Properties

Property	Type	Description
Utteranceld	string	Gets or sets the Utteranceld.
InteractionId	string	Gets or sets the InteractionId.
IsAgent	bool	Returns <code>True</code> if the owner is <b>InworldCharacter</b>
Event	Packets.TextEvent	This is triggered whenever the event is changed
Final	bool	Return <code>True</code> if the <b>HistoryItem</b> has been finished

On this page



# IAvatarLoader

## IAvatarLoader

This is the interface for loading avatars. It is implemented by different providers. If you would like to use your customized avatar loader, then you will need to: (1) inherit this interface; (2) create a prefab with your own inherited avatar loader; (3) set it as the `Avatar Loader` for `InworldAI`.

## API

Function	Return Type	Description	Parameters
ConfigureModel	void	For binding <code>InworldCharacterData</code> to avatar <code>gameObjects</code>	<code>character</code> : the <code>InworldCharacterData</code> to load, <code>model</code> : the model to attach
Import	IEnumerator	For loading avatars from the stream of URLs	<code>url</code> : a string of URLs to download avatars
LoadData	gameObject	For loading avatar from memory or bytestring	<code>content</code> : data to load
LoadData	gameObject	For loading avatar from local files	<code>fileName</code> : file to load

On this page



# IEyeHeadAnimLoader

## IEyeHeadAnimLoader

This is the interface for loading eye and head animations. It is currently `null` and only supports the basic feature of looking at the player. We do have default support for **Realistic Eye Movements**

To implement this, you can do the following:

1. Purchase and download the package
2. Add `LookTargetController` and `EyeAndHeadAnimator` to the **Inworld Character Prefab**
3. Create a prefab with the script inherited from the interface `GetComponent<EyeAndHeadAnimator>()`.
4. Allow the script to call `ImportFromJson()` with the data stored at `Resources/Animations/`
5. Put the prefab inside the head `anim` loader for `GLTFAvatarLoader`

If you would like to implement your own head animation, you can also create a prefab with a script that inherits this interface. Put that prefab inside the `GLTFAvatarLoader`'s `HeadAnimLoader`.

## API

Function	Return Type	Description	Parameters
<code>SetupHeadMovement</code>	<code>void</code>	For setting up head and eye movements	<code>avatar</code> : the model to attach

On this page



# ILipAnimations

## ILipAnimations

This is the interface for lip-syncing, and it is by default `null`. Our model also supports Oculus Lip Sync. To implement it, you need to fetch and download the [Oculus VR Unity Package](#)

Create a prefab with a script inheriting **ILipAnimation**, as well as `OvrLipSync`, `OvrLipSyncContent`, and  `AudioSource`.

Note that the viseme of `SkinnedMeshRender` for **Ready Player Me** characters starts at index 57 and ends in 74. The data values range from 0 to 1, while OVR's original demo ranged from 0 to 100.

## API

Function	Return Type	Description	Parameters
ConfigureModel	void	Let the model attach <b>LipSyncing</b>	<b>model</b> : the model to attach
StartLipSync	void	For the component to start lip-syncing	N/A
StopLipSync	void	For the component to stop lip-syncing	N/A

On this page



# IStudioDataHandler

## Inworld Studio

The Studio Data Handler interface for implementing studio connection APIs. In the editor, it is inherited as **InworldEditorStudio**. At run-time, it is inherited as **RuntimeInworldStudio**.

## API

Function	Return Type	Description	Parameters
CreateWorkspaces	void	For instantiating <b>InworldWorkspaceData</b>	<b>workspaces</b> : data returned from the server
CreateScenes	void	For instantiating <b>InworldSceneData</b>	<b>workspace</b> : target <b>InworldWorkspaceData</b> to add scenes. <b>scenes</b> : data returned from the server
CreateCharacters	void	For instantiating <b>InworldCharacterData</b>	<b>workspace</b> : target <b>InworldWorkspaceData</b> to add characters. <b>characters</b> : data returned from the server
CreateIntegrations	void	For instantiating <b>InworldKeySecret</b>	<b>workspace</b> : target <b>InworldWorkspaceData</b> to add key/secret pairs. <b>apiKeys</b> : data returned from the server
OnStudioError	void	For handling error messages returning from the studio server	<b>studioStatus</b> : the error message type of the studio server <b>msg</b> : the returning string of the message

Function	Return Type	Description	Parameters
OnUserTokenCompleted	void	To be invoked when the studio access token is generated	N/A



On this page



# InitInworld

## Init Inworld

Set all the data you want to load in this Unity Scene.

## Inspector Variables

Variable	Description
WS Data	The <b>InworldWorkspaceData</b> you want to load.
Inworld Scene Data	The <b>InworldSceneData</b> you want to load.
Char Data	The <b>InworldCharacterData</b> you want to load.
Key Secret	The <b>InworldKeySecret</b> you want to load.

On this page



# InworldAnimation

## InworldAnimation

Interface for handling emotion and gesture events from our server.

## Properties

Property	Type	Description
Animator	Animator	The animator that is attaching
Character	<a href="#">InworldCharacter</a>	Gets or sets the <a href="#">InworldCharacter</a> that is attaching

## API

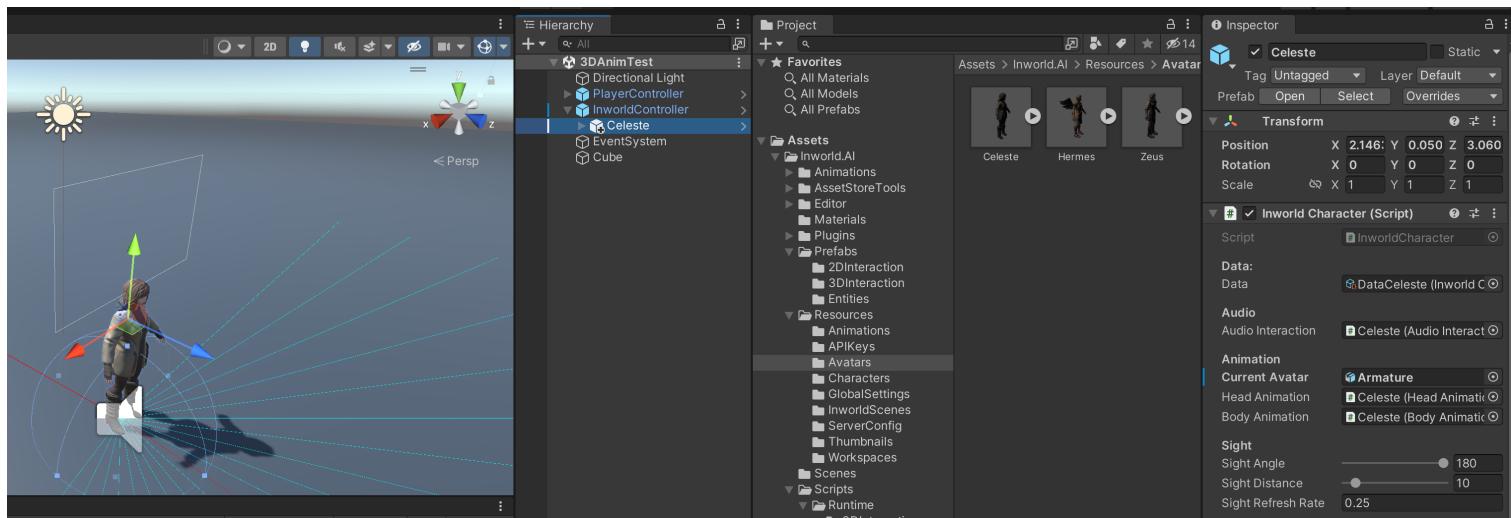
Function	Return Type	Description	Parameters
HandleMainStatus	void	This is for handling the main status of a character, i.e., <a href="#">Idle</a> , <a href="#">Talking</a> , <a href="#">Walking</a> , etc.	<b>status:</b> incoming status
HandleEmotion	void	For playing animation according to the target emotion. Please implement this function to select and play your customized animations.	<b>SpaffCode:</b> An enum of emotion
HandleGesture	void	For playing the target gesture's animations. Please implement this function to select and play your customized animations.	<b>gesture:</b> An enum of target gesture

On this page



# InworldCharacter

This script implements the **Inworld Character** that players can communicate with.



## Inspector Variables

Variable	Description
Data	The <b>InworldCharacterData</b> for this character
Audio Interaction	The character's <b>AudioInteraction</b>
Current Avatar	The character's Animator
Sight Angle	The character's field of view, displayed in Gizmos
Sight Distance	The character's sight distance, displayed in Gizmos
Sight Refresh Rate	How often the characters are calculating priority

## Properties

Property	Type	Description
Data	InworldCharacterData	Returns InworldCharacterData
ID	string	Returns the ID for the character. The ID or CharacterID field will be generated only after LoadingScene() is successful
CharacterName	string	The display name for the character. Note that CharacterName may not be unique
BrainName	string	The BrainName for the character. Note that BrainName is actually the character's full name, formatted like workspace/xxx/characters/xxx. It is unique
IsValid	bool	Returns True if the character has InworldCharacterData
HasRegisteredLiveSession	bool	Checks if this InworldCharacter has been registered in the live session
Priority	float	Returns the priority of the character. The higher the priority is, the more likely the character is to respond to the player
CurrentAudioRemainingTime	float	Returns the remaining time for the character's audioClip. This data will be modified by AudioInteraction
Event	InteractionEvent	Returns the Unity Event of Interaction
Audio	AudioInteraction	Gets or sets the character's AudioInteraction.
CurrentAvatar	gameObject	Gets or sets the model it is bound to
Emotion	string	Get the current emotion (last emotion received from server) of the character.

Property	Type	Description
Gesture	string	Get the current gesture (last gesture received from server) of the character.

# API

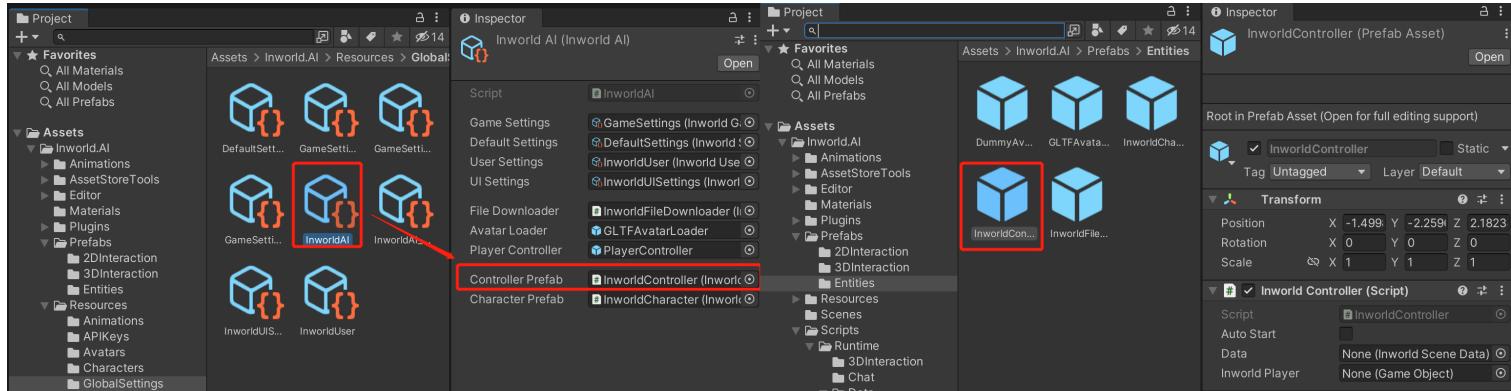
Function	Return Type	Description	Parameters
LoadCharacter	void	Lets an <b>InworldCharacter</b> bind data. This model can be of any kind (e.g., even a cube). If the input model is <code>null</code> , then it will download a model and attach to it whenever it has a valuable <code>modelUri</code> . Otherwise, it will attach to the default avatar.	<b>incomingData:</b> The <code>InworldCharacterData</code> to bind to <b>model:</b> The model to bind to
ResetCharacter	void	Resets the history items for a character and performs audio cache clean-up	N/A
SendText	void	Sends text to the character via a <b>InworldPacket</b>	<b>text:</b> the text to send
SendTrigger	void	Send target character's trigger via <b>InworldPacket</b> .	<b>triggerName:</b> The trigger to send. Both formats are acceptable. You could send either whole string from <code>CharacterData.trigger</code> , or the trigger's <code>shortName</code> .
SendEventToAgent	void	Sends general events to the character	<b>packet:</b> The <code>InworldPacket</code> to send

On this page



# InworldController

The **InworldController** is the main controller. It is a singleton `gameObject`. Please ensure that there is only one **InworldController** in the scene.



## Inspector Variables

Variable	Description
Auto Start	Set to <code>True</code> if you want to immediately load the scene once a run-time token is acquired
Data	The <b>InworldSceneData</b> that is being loaded
Inworld Player	The <b>Player Controller</b> that the <b>InworldCharacters</b> are communicating with

## Events

Event	Type	Description
OnStateChanged	Action< <b>ControllerStates</b> >	Triggered on a status change
OnPacketReceived	Action< <b>InworldPacket</b> >	Triggered on the receival of <b>InworldPackets</b>

Event	Type	Description
OnCharacterChanged	Action<InworldCharacter, InworldCharacter>	Triggered whenever the highest priority InworldCharacter is changed

## Properties

Property	Type	Description
IsCapturing	bool	Get if it's capturing player's audio from microphone.
AutoStart	bool	Gets or sets at AutoStart. The <b>Inworld Controller</b> would immediately start the session once the token is received if <b>Auto Start</b> is checked
CurrentScene	InworldSceneData	Gets or sets the current scene data
Player	gameObject	Gets or sets the current <b>Player Controller</b>
CurrentCharacter	InworldCharacter	Gets or sets the current <b>InworldCharacter</b> . Usually, it is set by <b>CheckPriority()</b>
Characters	List<InworldCharacter>	Gets or sets all the characters that the <b>InworldScene</b> contains
HasInit	bool	Checks if the run-time session token has been received and if the client has been initialized
State	ControllerStates	Gets or sets the controller's current state. Once the state has been set, its <b>OnStateChanged</b> event will be invoked
IsValid	bool	Checks if all the data is correct

## API

Function	Return Type	Description	Parameters
----------	-------------	-------------	------------

Function	Return Type	Description	Parameters
Init	void	Initializes the SDK. Ensure that there is a valid ServerConfig (i.e., it has a valid URI for both the <b>RuntimeServer</b> and <b>StudioServer</b> ) and a valid API Key and API Secret	N/A
LoadCharacter	System.Threading.Tasks.Task	Starts a session with the target <b>InworldCharacter</b> . If the session is successful, then this will create a default session for that character with a valid <b>SessionKey</b> and <b>AgentID</b> .	<b>character:</b> A <code>gameObject</code> or prefab that has an <b>InworldCharacter</b> Component.
LoadScene	System.Threading.Tasks.Task	Starts a session with the target <b>InworldSceneData</b> . If the session is successful, then this sets the current <b>InworldSceneData</b> to the new one, with a valid <b>SessionKey</b> and a list of its <b>InworldCharacter</b> with valid <b>AgentIDs</b> .	<b>inworldSceneData:</b> The InworldScene to load.
Disconnect	System.Threading.Tasks.Task	Disconnects and ends any server-based NPC interactions	N/A
SendEvent	void	Sends an <b>InworldPacket</b> to our server	<b>packet:</b> <code>InworldPacket</code> to send

Function	Return Type	Description	Parameters
StartAudioCapture	void	Starts communicating with the target character via audio	<b>characterID:</b> A string representing the character ID. It is generated after the InworldScene is loaded and the session is started
EndAudioCapture	void	Stop communicating with the target character via audio	<b>characterID:</b> A string representing the character ID. It is generated after the InworldScene is loaded and the session is started
RegisterCharacter	void	Called by <b>InworldCharacter</b> to register them into <b>InworldController's</b> character list	<b>character:</b> The <b>InworldCharacter</b> to add
GetFirstChild	<b>InworldCharacter</b>	Get the first attached <b>InworldCharacter</b>	<b>isActive:</b> if it's true, returns first active <b>InworldCharacter</b> , else return the first inactive <b>InworldCharacter</b>

On this page



# InworldEditor

## InworldEditor

### Properties

Property	Type	Description
Instance	<b>InworldEditor</b>	Gets an instance of the <b>InworldEditor</b> . It will create an <b>Inworld Studio Panel</b> if the panel has not opened
Progress	Dictionary< <b>InworldWorkspaceData</b> , <b>WorkspaceFetchingProgress</b> >	Gets the current data fetching progress across all the workspaces
Status	<b>InworldEditorStatus</b>	Gets or sets the current Inworld Editor Status. The old status will call <code>OnExit()</code> and the new status will call <code>OnEnter()</code>
CurrentProgress	<b>InworldUISettings</b>	Gets the current workspace's data fetching progress
IsValidData	bool	Checks if all the data in the current data of the <b>Inworld Game Setting</b> is true

### API

Function	Return Type	Description	Parameters

Function	Return Type	Description	Parameters
ShowPanel	void	Opens the <b>Inworld Studio Panel</b> . This will detect and pop the import window if you do not have TMP imported	N/A
SetupInworldCharacter	void	Sets a <code>gameObject</code> in the scene with <b>InworldCharacterData</b> . If the <b>InworldCharacterData</b> belongs to an <code>InworldAI.Game.CurrentScene</code> , then the data and its related components (e.g., animation, player detecting, etc.) will be added to the <code>gameObject</code> . All the characters that have associated <b>Inworld Character Data</b> but that are not in the current <b>Inworld Scene</b> will be deleted	<b>avatar:</b> That <code>gameObject</code> that you want to bind an <b>InworldCharacter</b> to. <b>selectedCharacter:</b> The <b>InworldCharacterData</b> to add to the <code>gameObject</code> . Note that the <b>InworldCharacterData</b> should be in <code>InworldAI.Game.CurrentScene</code>
LoadPlayerController	void	Adds a player controller into the current scene. Note that: 1. Player controller is mandatory for the characters to communicate. 2. If you call this function, then the main camera in your current scene will be deleted. If you have your own player control that allows characters to communicate, then add <code>InworldController.Player</code> to your customized controller object	N/A

# InworldEnums

## InworldEnums

Stores all the **Enums** declared and used in this package.

Enum Type	Description
StudioStatus	The status of <b>InworldStudio</b> .
InteractionStatus	The status of conversation of <b>InworldCharacters</b>
RuntimeStatus	The status of the run-time server
Ownership	Indicates if the <b>InworldCharacterData</b> was created by default or by the user
InworldEditorStatus	The status of <b>InworldEditor</b>
ControllerStates	The status of <b>InworldController</b>
InworldSceneStatus	The status of an InworldScene at run-time
AnimMainStatus	The main status of the animator, i.e., <code>Idle</code> , <code>Talking</code> , etc.
Gesture	The enum for the gesture returned by the server
Emotion	the enum for the emotions returned by the server

On this page



# InworldFileDownloader

## InworldFileDownloader

### CharacterFetchingProgress

This class is used to get data fetching progress for an **InworldCharacter**. It stores all the enums declared and used in this package.

### Properties

Property	Type	Description
thumbnailProgress	UnityWebRequestAsyncOperation	Gets the progress of the thumbnail download
avatarProgress	UnityWebRequestAsyncOperation	Gets the progress of the avatar download
Progress	float	Gets the progress of the download as a percentage (0 to 100)

## InworldFileDownloader

### Inspector Variables

Variable	Description
Download Thumbnail	Describes if it downloads thumbnails
Download Avatar	Describes if it downloads avatars

## Events

Event	Type	Description
OnAvatarDownloaded	Action<InworldCharacterData>	Triggered when .g1b model downloaded
OnThumbnailDownloaded	Action<InworldCharacterData>	Triggered when thumbnails downloaded
OnAvatarFailed	Action<InworldCharacterData>	Triggered when .g1b model downloading failed
OnThumbnailFailed	Action<InworldCharacterData>	Triggered when thumbnail downloading failed

## Properties

Property	Type	Description
Progress	float	Gets the progress of the download as a percentage (0 to 100)

## API

Function	Return Type	Description	Parameters
DownloadCharacterData	void	Downloads thumbnail and avatar of InworldCharacterData.	<b>charData:</b> target InworldCharacterData
Init	void	Clears all the current downloading requests	N/A
DownloadThumbnail	void	Downloads thumbnail of InworldCharacterData	<b>charData:</b> target InworldCharacterData
DownloadAvatar	void	Downloads avatar of InworldCharacterData	<b>charData:</b> target InworldCharacterData

DownloadAvatar

On this page



# InworldLipAnimation

## Inworld Lip Animation

This class is used to receive phone data from server, and morph the face to do the lip syncing.

## Inspector Variables

Variable	Description
Face Anim Data	The scriptable object that stores morphed data
Viseme Sil	The name for the first viseme index in <b>SkinnedMeshRenderer</b>
Lip Expression	How drastic the lip moves. Range from 0 to 1.

## Properties

Property	Type	Description
Character	<b>InworldCharacter</b>	Gets or sets the character this component uses

## API

Function	Return Type	Description
Init	void	Start the lip sync

On this page



# InworldPlayer

## Chat Bubble

This is the class for global text management. It is originally added in **Player Controller**, where it would be called by `KeyCode.Backquote`.

## Inspector Variables

Variable	Description
Camera Controller	the camera controller for controlling the main player
Global Chat Canvas	The canvas in runtime for displaying history logs and support typing/recording. Could be <b>NULL</b>
Trigger Canvas	The canvas in runtime to display triggers. mostly is <b>NULL</b> . Only used in some sample cases.
Record Button	The button for holding record in global panel
Content RT	The RectTransform of the Content
Bubble Left	The <b>ChatBubble</b> aligned to the left
Bubble Right	The <b>ChatBubble</b> aligned to the right
Input Field	The Input Field
RT Canvas	The real time canvas, mostly is <b>NULL</b> . Only used in some sample cases.
Init Position	It's initial position, used to reset. Used in some sample cases.

Variable	Description
Init Rotation	It's initial rotation, used to reset. Used in some sample cases.

## API

Function	Return Type	Description	Parameters
SendText	void	UI Functions. Called by the <code>Send</code> button or <code>KeyCode.Return</code>	N/A
BackToLobby	void	UI Functions. Only used in some sample cases, used to get back to lobby.	

On this page



# InworldStudio

## Inworld Studio

**InworldStudio** is a data processing class for communicating with the GRPC server. Its response data would be stored in **InworldUserSettings**.

## API

Function	Return Type	Description	Parameters
InworldStudio	InworldStudio	Constructor	<b>owner:</b> The StudioDataHandler Interface: In Editor, it is InworldEditorStudio. In Runtime, it is RuntimeInworldStudio.
GetUserToken	System.Threading.Tasks.Task	Gets the user token (i.e., studio access token). If returned, data would be overwritten in InworldAI.User. The studio handler would then invoke <code>OnUserTokenCompleted</code> . If failed, the studio handler will invoke <code>OnStudioError</code> .	<b>tokenForExchange:</b> ID Token used to exchange the user token (i.e., studio access token), or {OculusNonce and OculusID} if you are using Oculus. <b>authType:</b> authType, by default this is firebase.

Function	Return Type	Description	Parameters
ListWorkspace	System.Threading.Tasks.Task	<p>Lists the workspaces. If returned, the handler for the studio will invoke <code>CreateWorkspaces</code>. Otherwise, the handler will invoke <code>OnStudioError</code>.</p>	N/A
ListScenes	System.Threading.Tasks.Task	<p>Lists the scenes. If returned, the handler of Inworld Studio will invoke <code>CreateScenes</code>. Otherwise, the handler will invoke <code>OnStudioError</code>.</p>	<b>workspace</b> : list InworldScenes in <b>InworldWorkspaceData</b>
ListCharacters	System.Threading.Tasks.Task	<p>Lists the characters. If returned, the handler of the studio will invoke <code>CreateCharacters</code>. Otherwise, the handler will invoke <code>OnStudioError</code>.</p>	<b>workspace</b> : lists the characters in <b>InworldWorkspaceData</b>
ListSharedCharacters	System.Threading.Tasks.Task	<p>Lists shared characters. This function works in Oculus only. It needs Oculus Nonce and an ID instead of an ID Token to get shared characters. If returned, the handler of the studio will invoke <code>CreateCharacters</code>. Otherwise, the handler will invoke <code>OnStudioError</code>.</p>	<b>sharedWorkspace</b> : the <b>InworldWorkspaceData</b> for list sharing characters. <b>oculusNonce</b> : data obtained from Oculus. <b>oculusID</b> : data obtained from Oculus

Function	Return Type	Description	Parameters
ListAPIKey	System.Threading.Tasks.Task	<p>Lists API Keys. If returned, the handler of the studio will invoke <code>CreateIntegrations</code>, Otherwise, the handler will invoke <code>OnStudioError</code>.</p>	<p><b>workspaceData</b>: list of API Keys and Secrets in <b>InworldWorkspaceData</b></p>

On this page



# RecordButton

## RecordButton

This class is used for the Record Button in the global chat panel.

## Properties

Property	Type	Description
IsRecording	bool	Returns if the button is holding

## API

Function	Return Type	Description	Parameters
OnPointerDown	void	Interface Functions. Usually called when this button is pressed.	<b>eventData:</b> PointerEventData for current Input data.
OnPointerUp	void	Interface Functions. Usually called when this button is released.	<b>eventData:</b> PointerEventData for current Input data.

On this page



# ResetNeutral

## Reset Neutral

Set the animation status to neutral. Called in **Animator Inworld Anim Ctrl**.

## Inspector Variables

Variable	Description
Face Anim Data	The scriptable object that stores morphed data
Viseme Sil	The name for the first viseme index in <b>SkinnedMeshRenderer</b>
Lip Expression	How drastic the lip moves. Range from 0 to 1.

## Properties

Property	Type	Description
Character	<b>InworldCharacter</b>	Gets or sets the character this component uses

## API

Function	Return Type	Description
OnStateEnter	void	Interface function, called in Animator.

On this page



# RuntimelnworldStudio

## RuntimelnworldStudio

This class is used to acquire studio access tokens, connect to the studio server, and fetch data at run-time.

## API

Function	Return Type	Description	Parameters
Init	void	Used for acquiring studio access tokens	<b>tokenForExchange</b> : id token to exchange.
Init	void	Used for acquiring studio access tokens via an Oculus account	<b>oculusNonce</b> : random string generated from Oculus <b>oculusID</b> : Oculus ID number
CreateWorkspaces	void	Instantiating <b>InworldWorkspaceData</b>	<b>workspaces</b> : data returned from the server
CreateScenes	void	Instantiating <b>InworldSceneData</b>	<b>workspace</b> : target <b>InworldWorkspaceData</b> to add scenes <b>scenes</b> : data returned from the server
CreateCharacters	void	Instantiating <b>InworldCharacterData</b>	<b>workspace</b> : target <b>InworldWorkspaceData</b> to add characters <b>characters</b> : data returned from the server

Function	Return Type	Description	Parameters
CreateIntegrations	void	Instantiating <b>InworldKeySecret</b>	<b>workspace</b> : target <b>InworldWorkspaceData</b> to add key/secret pairs <b>apiKeys</b> : data returned from the server
OnStudioError	void	Handling error messages returning from the studio server.	<b>studioStatus</b> : the error message type for the studio server <b>msg</b> : the returning string of the message
OnUserTokenCompleted	void	This is invoked when the studio access token has been generated	N/A

On this page



# StudioDataClasses

The **StudioDataClasses** store all the data classes used in editor mode.

## API

Class	Description
FBTokenResponse	The response by the firebase server, returning the ID token that can be used to exchange the <b>studio access token</b> .
WorkspaceFetchingProgress	Data class to calculate the progress of fetching all the data in <b>InworldWorkspaceData</b>

On this page



# TransformCanvas

## Transform Canvas

This class is for the UI Component in sample scenes.

## Inspector Variables

Variable	Description
Stone	The <b>GameObject</b> of the stone
Avatar	The <b>GameObject</b> of the avatar
Char Data	The <b>InworldCharacterData</b> of the character
Lip Animation	The <b>InworldLipAnimation</b> of that character

On this page



# legacy

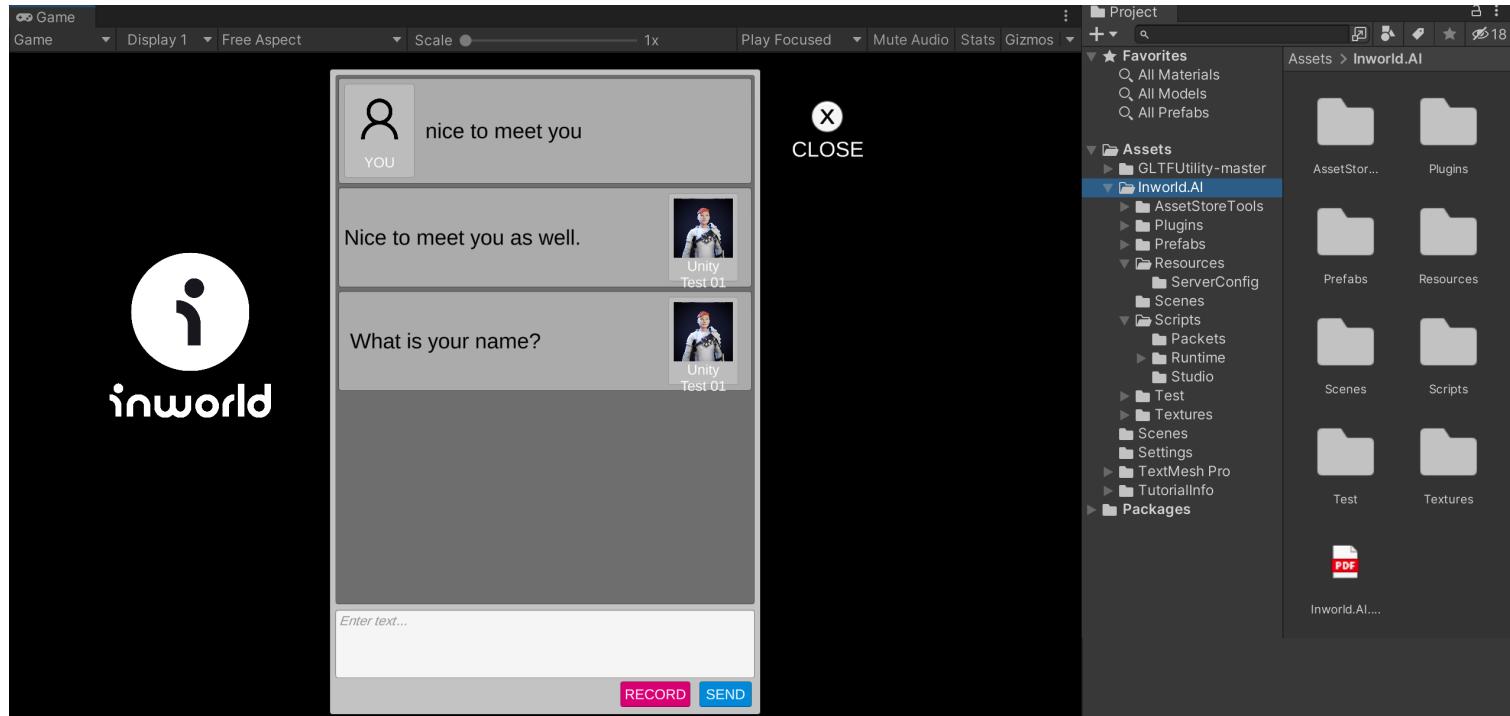
## Legacy Integrations

The **Legacy Inworld AI Unity SDK** is cross-platform virtual character integration plugin for Unity. It supports communication in 2D.

 **Note:** The API for the legacy package may be different from the current one.

## Download

You can download our Unity Integration package [here](#). Before getting you started, this tutorial series will begin with an overview of compatibility, assets, and API references.



## Unity Package Structure

1. `Plugins/` - contains all native plugin files. Do not modify or remove these.
2. `Prefabs/` - contains the `ChatScreen/` and `MainScreen/`

3. `Resources/` contains following:

- **Inworld Console**, a sample object for showcasing how to use the package
- **Inworld Controller**, the main singleton object with which you will call the API. Ensure that there is only one InworldController in the scene
- **Inworld Character**, the AI character instance for you to communicate with
- **Inworld Scene**, a scenario placeholder to list characters
- `ChatScreen/` and `MainScreen/` folders for use in demo scenes

4. `Scenes/` contains a sample scene to showcase our Unity integration

5. `Scripts/` contains:

- `ForceInitialize.cs`, the file used for solving linking issues in IL2CPP. This file cannot be called, but you should not remove it if your target platform is building with IL2CPP.
- `Packets/`, the folder containing files of packets to communicate with our server
- `Audio/`, the folder that captures audio-related files
- `Auth/`, the folder that is used for authentication and logging into our server
- `Chat/`, a folder for demo use only
- `Data/`, a folder for data implementation (e.g., character or server properties)
- `Entities/`, the implementation of SDK related `gameObjects`
- `Util/`, an aggregation of enums, events, tools, and files used

6. `Textures` contains demo-related files

On this page



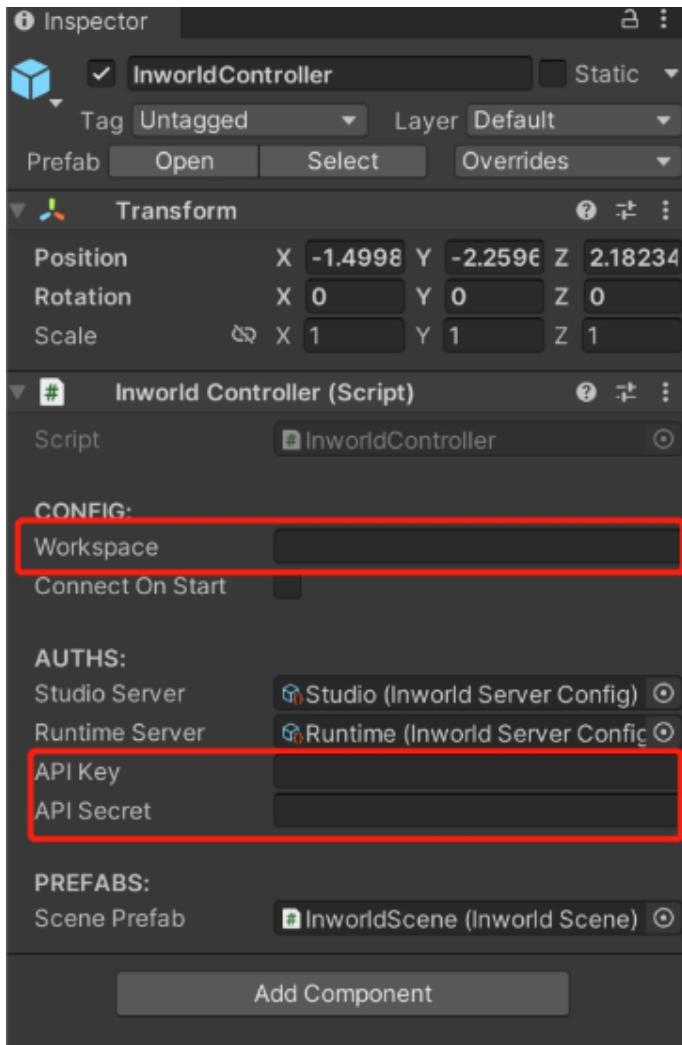
# quick start

## Getting Started

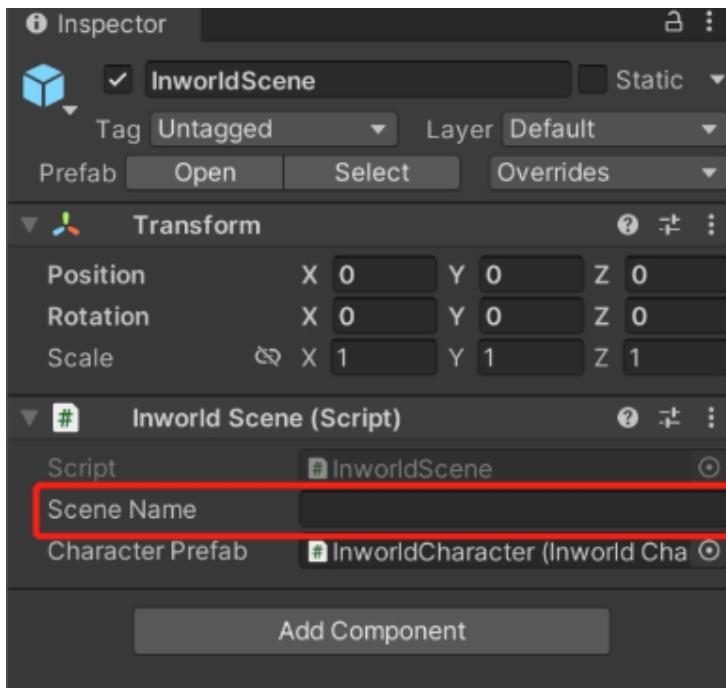
## Your Unity Environment

Our Unity Integration package comes with an exemplar **Unity Scene** that connects to your **InworldScene** and enables you to talk to your characters. Select “Yes” to any update prompts as you do the following:

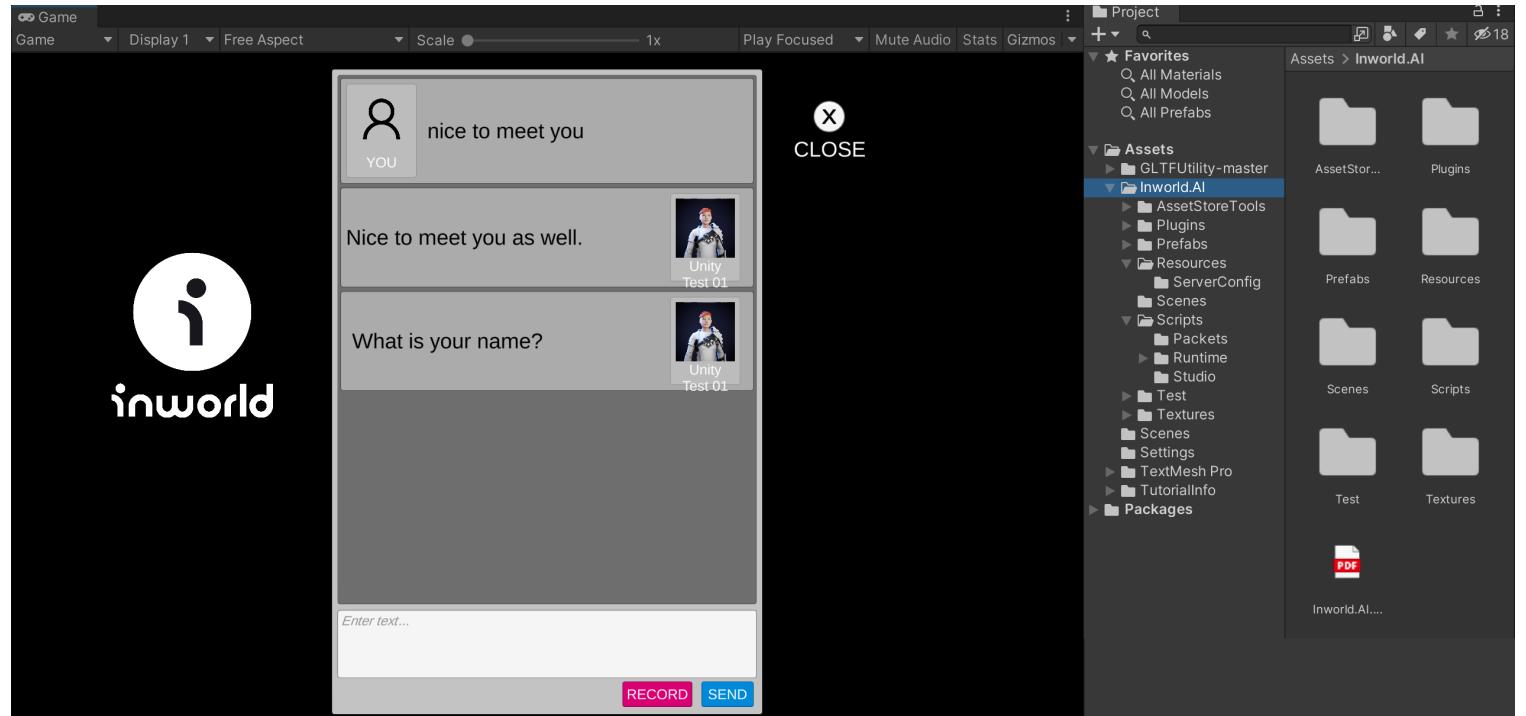
1. Open your project in Unity and import the Asset Package file at `Assets` > `Import Package` > `Custom Package`
2. Select the **Demo Scene** from `Assets/Inworld.AI/Scenes`
3. With the **Demo Scene** open, select `InworldController` and input your **Workspace Name**, **API Key**, and **API Secret**



#### 4. Enter your **Scene Name** in **InworldScene**.



Now, you can run the scene! Press **Login** and then **Load Inworld Scene** to view a list of all characters in that scene. Afterwards, you should be able to interact with the characters in Unity.



On this page



# compatibility

## Compatibility

### Unity Version

Unity Version	Tested Version	API Level	Note
2019.4	2019.4.38f1	.NET 4.x Only	In <code>Project Settings</code> > <code>Player</code> > <code>Other Settings</code> , uncheck "Assembly Version Validation"
2020.3	2020.3.34f1	.NET 4.x Only	
2021.1	2021.1.21f1	.NET 4.x Only	
2021.2	2021.2.0f1	.NET Standard 2.1 or .NET Framework	
2021.3	2021.3.2f1	.NET Standard 2.1 or .NET Framework	

## Configuration

Scripting Backend

Mono

Api Compatibility Level\*

.NET Standard 2.1

C++ Compiler Configuration

Release

Use incremental GC



Assembly Version Validation



Active Input Handling\*

Input Manager (Old)

## Platform

Other platforms will be tested and updated shortly.

Tested Platform	Scripting Backend	API Level
Windows	MONO	.NET Standard 2.1 or .NET 4.x+
Android	IL2CPP	.NET 4.x
Oculus	IL2CPP	.NET 4.x

## TextMeshPro

If you would like to try our provided example, please download **TextMeshPro** from Unity Package Manager.

Package Manager

+ Packages: In Project Sort: Name ↓

Packages - Unity

TextMeshPro 3.0.6 ✓

**TextMeshPro** Release

Unity Technologies

Version 3.0.6 - April 22, 2021

Registry Unity

[View documentation](#) • [View changelog](#) • [View licenses](#)

TextMeshPro is the ultimate text solution for Unity. It's the perfect replacement for Unity's UI Text and the legacy Text Mesh.

Powerful and easy to use, TextMeshPro (also known as TMP) uses Advanced Text Rendering techniques along with a set of custom shaders; delivering substantial visual quality improvements while giving users incredible flexibility when it comes to text styling and texturing.

TextMeshPro provides Improved Control over text formatting and layout with features like character, word, line and paragraph spacing, kerning, justified text, Links, over 30 Rich Text Tags available, support for Multi Font & Sprites, Custom Styles and more.

Great performance. Since the geometry created by TextMeshPro uses two triangles per character just like Unity's text components, this improved visual quality and flexibility comes at no additional performance cost.

Last update May 25, 13:58

C | ▾

Remove



On this page



# Prerequisites

## Prerequisites

This section will cover how to set up the demo scene. Please review the prerequisites that follow.

## Studio Requirements

To integrate **Workspaces**, **Characters** and **Scenes** into your Unity project, you will need to identify their alphanumeric MR Codes.

1. Identify your workspace MR Code, e.g., `workspaces/my-new-workspace`
2. Ensure that your characters have been added to at least one scene
3. Identify the MR Code for your scene, e.g., `workspaces/my-new-workspace/scenes/my-new-scene`
4. Identify the MR Codes for the characters that you want to integrate into your project
5. Generate an API Key and Secret from the Integrations Tab at `API Keys` > `+ Generate New Key`

## MR Codes

You can find each required MR Code in the Studio UI:

### Workspace MR Code

The screenshot shows the 'Workspaces' section of the Inworld Studio interface. At the top, there's a header with the 'inworld' logo and a 'Create new workspace' button. Below the header, there's a table-like list of workspaces. The first row contains 'Inworld Sandbox' with the URL 'workspaces/default-fymukhvtzx5ce4x3rfhnw'. The second row contains 'Wonderland' with the URL 'workspaces/wonderland'. Each row has three icons on the right: a person icon, a pencil icon, and a trash bin icon. The entire list is enclosed in a light gray border.

Workspace	URL
Inworld Sandbox	<a href="http://workspaces/default-fymukhvtzx5ce4x3rfhnw">workspaces/default-fymukhvtzx5ce4x3rfhnw</a>
Wonderland	<a href="http://workspaces/wonderland">workspaces/wonderland</a>

### Scene MR Code

[Characters](#)[Scenes](#)[Common Knowledge](#)[Integrations](#)[Discord](#)[Documentation](#)[Contact support](#)[Terms of Use | Privacy policy](#)

## Scenes

[+ Create new scene](#)

Name	Description	Scene triggers
Greek Mythology	My scene	

[Copy machine readable id for integration](#)

## Character MR Code

Workspace  
Inworld Sandbox[Change](#)

J

[Characters](#)[Scenes](#)[Common Knowledge](#)[Integrations](#)

Celeste



Hermes



Morgana



Zeus

- [Delete character](#)
- [Copy machine-readable id for integration](#)
- [Share character](#)

If you have not yet created your **Workspace**, **Character** and **Scene**, please check out our **Studio Basics Tutorial Series**.