Virtual Agent Interaction Framework

VAIF Quick Help Guide

Git Version 3.2

ADVANCED AGENT ENGAGEMENT TEAM

Table of Contents

1. Overview
    1.1. What is VAIF?
    1.2. What is an AGENT?
    1.3. Features of VAIF
        1.3.1. Current Features
        1.3.2. Soon-To-Come Features
        1.3.3. Future Work
    1.4. History of Releases
        1.4.1. Current: VAIF v3.2
        1.4.2. VAIF v3.1
        1.4.3. VAIF v3.0
        1.4.4. VAIF v2.0
        1.4.5. VAIF v1.0

2. Required: Install Unity (2017.1 or later)

3. Downloading the VAIF Package

4. Import To Unity

5. The Example Scene

6. Create a Timeline
    6.1. Create a Conversation
    6.2. Types of Events
    6.3. Adding Events to a Conversation
        6.3.1. Exceptions: General Events
            6.3.1.1. Editing an Event's "Next Event" Field
            6.3.1.2. Editing an Event's "Agent" Field
            6.3.1.3. Editing an Event's "Want In Range" Field
            6.3.1.4. Editing an Event's "Want Looked At" Field
        6.3.2. Exceptions: Specific Events
            6.3.2.1. Response Events
            6.3.2.2. Wildcard Events

7. Creating Your Agents
    7.0. Import the Agent
    7.1. Using a VAIF-provided Agent (prefab)

7.2. Adding the Managers to an Agent (non-prefab)
7.2.1. Add the Agent Status Manager
7.2.2. Add the Move Manager
7.2.3. Add the Emote Manager (Future Implementation)
7.2.4. Add the Dialog Manager
7.2.5. Add the Animation Manager
7.2.6. Add the Look At Collision Script
7.2.7. Add the Personalities Script (Future Implementation)
7.2.8. Add the Emotions Script (Future Implementation)
7.2.9. Add the Gaze Manager (Future Implementation)
7.2.10. Add the Rigidbody Component
7.2.11. Capsule Collider
7.2.12. Nav Mesh Obstacle
7.2.13. Adding the Managers to the Agent's Head
7.2.13.1. Add the LipSyncInterface Prefab
7.2.13.2. Add the OVRLipSyncContext Script
7.2.13.3. Add the OVRLipSyncContext Morph Target Script
7.3. Integrating the Vive VR Headset
8. Troubleshooting

1. Overview

1.1. What is VAIF?

The Virtual Agent Interaction Framework (VAIF) is a Unity package that allows users to create intelligent agents with minimal effort.

1.2. What is an AGENT?

An Embodied Conversational Agent (ECA) is a virtual character who is able to mimic human-like characteristics, ranging from speech to gestures.

1.3. Features of VAIF

1.3.1. Current Features

1. Unity GUI Usability: hover over fields in the script components of the Unity GUI (see Section 1.3.1); e.g, hovering over the Want Looked At field:

2. Multi-Agent interaction: navigating between Conversations to interact with various agent(s). A user can leave a conversation, and return to it to replay the previous event.
3. Wildcards: handle general verbal inputs (no need for recognition)
4. Responses: handle specific responses (recognition by keyword phrases)

1.3.2. Soon-To-Come Features

1. GazeManager: control agent eye-gaze and head movement -- requires blendshape (facial expressions)
2. MemoryManager: save and use events with each agent/user combination
3. GestureManager: recognize gestures from users during interaction

1.3.3. Future Work

1. Network Manager: add multiplayer support
2. EmotionCheckManager: based on memory (state of mind) -- requires blendshape (facial expressions)

1.4. History of Releases

1.4.1. Current: VAIF v3.2

Improved usability and organization

1.4.2. VAIF v3.1

Refactored Conversation code to working order.

1.4.3. VAIF v3.0

Added Conversations. Needs major debugging.

1.4.4. VAIF v2.0

The tool now allows for generalized state management (AgentStatusManager and ESV), rather than relying on only listening/speaking.

1. Responses: Fixed to recognize inputs.
2. JumpManager handles the sequence of events within a conversation.

1.4.5. VAIF v1.0

This previous version tracks when EventIM type instances (Response, Dialog, Animation, etc.) are completed. No more running through a list of events!

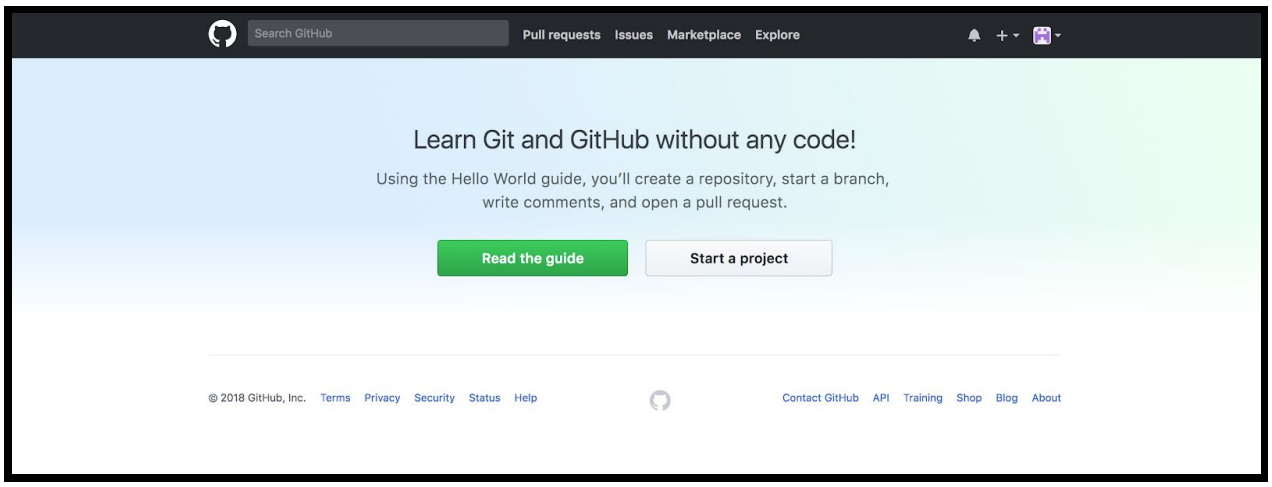
2. Required: Install Unity (2017.1 or later)

1. Install Unity 2017.1 or later at this page: https://unity3d.com/get-unity/download
2. Expect this download to take a while as Unity is a large program.
3. Visit the VAIF package from the GitHub repository (available at https://github.com/isguser/VAIF).
4. Download the VAIF package. (*see Section 3)*

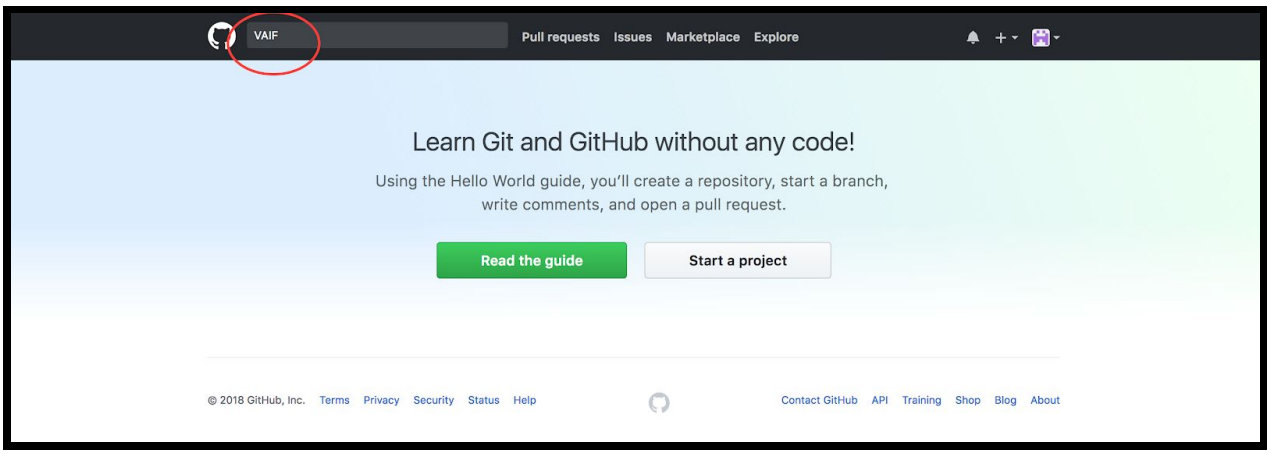
Tutorial Video: VAIF 1

3. Downloading the VAIF Package

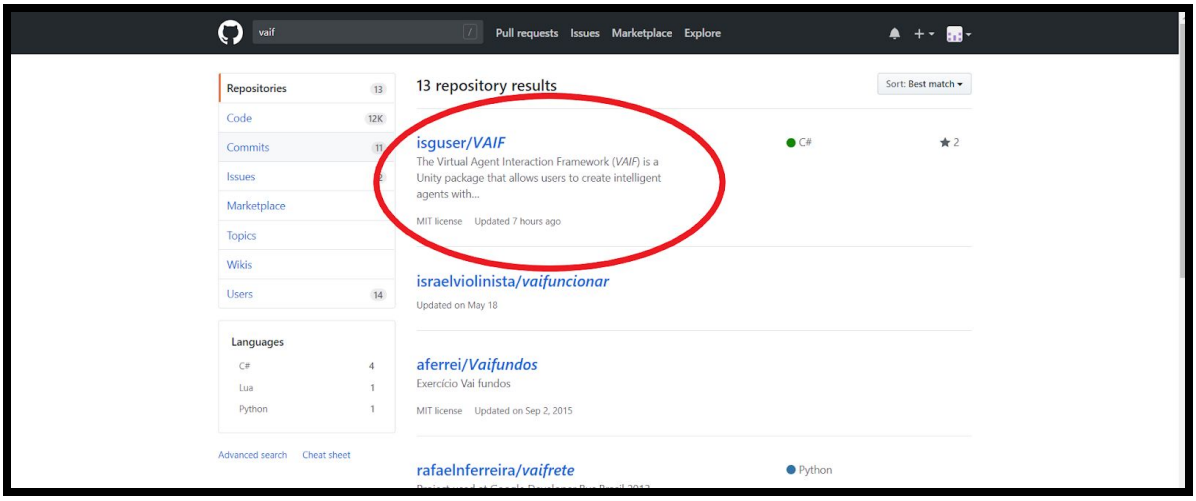
1) Visit the Github website and you should see a screen as follows:



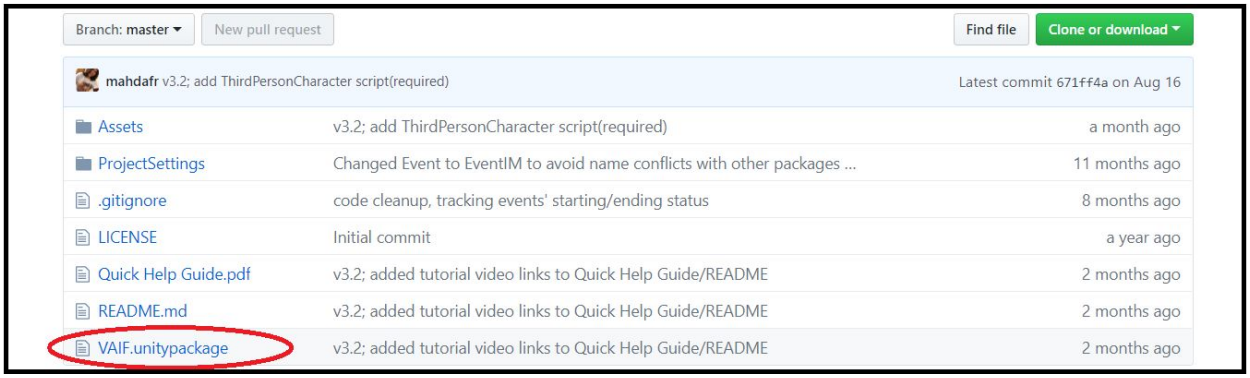
2) Type **VAIF** (all caps) into the '*Search Github*' bar:



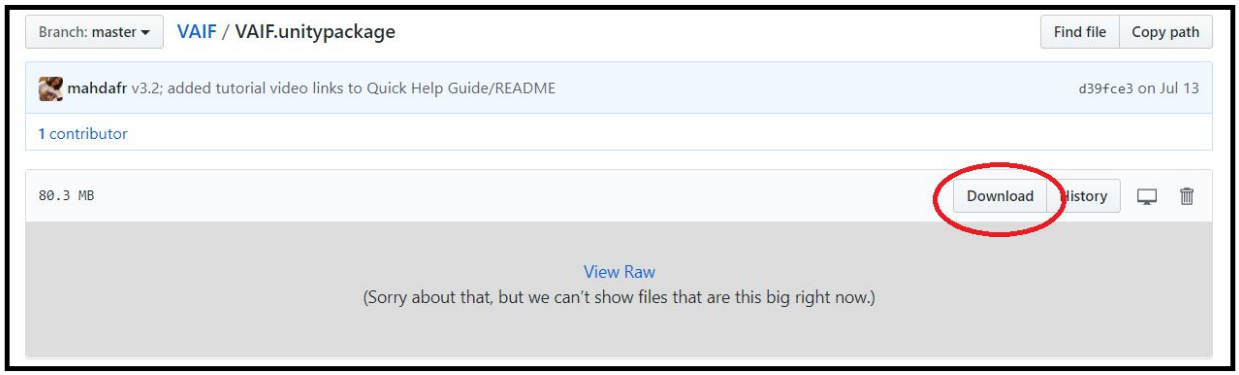
3) Click the link to the project **isguser/VAIF**:



4) Click the **VAIF.unitypackage** file in the repository:



5) Click the **Download** option (select any directory or location you prefer) and the wait for the file to finish downloading to your computer:

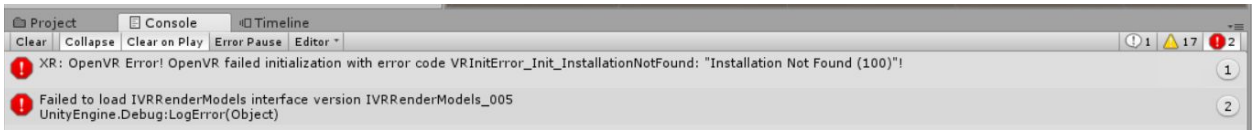


Tutorial Video: [VAIF 1]

4. Import To Unity

1. In Unity, create a new Unity Project. You can also import VAIF to a saved Unity project.
2. Select “Assets” > “Import Package” > “Custom Package”
3. Navigate to the directory where you downloaded the VAIF Unity Package (from Section 3 Downloading the VAIF Package), and select the VAIF.unitypackage file.
4. Click “Open”.
5. When the “**Import Unity Package**” window opens in Unity, click “Import” on the bottom right of the small window.
6. A small window with a progress bar will open and close repeatedly. Do not worry about this, Unity is importing the VAIF package and its contents to the project.
7. When the “**SteamVR_Settings**” window opens in Unity, click “**Accept All**” on the bottom left of the small window.
8. A new dialog box will give you the message that “**You made the right choice!**”. Click “Ok”.
9. Begin designing your project (proceed to Section 6: Create a Timeline)

NOTE: If you do not have a VR headset connected to your system, every time you run the scene in Unity, this error will come up in the Console tab:



Do not worry about this error. You can still test your scene in Unity without having any VR Headset connected. The error will not appear when you plug in your SteamVR Headset.
Tutorial Video: VAIF 2

5. The Example Scene

Included in the downloadable package on Unity is an example scene in Unity called ExampleScene. This scene contains an example of the different types of events to create. You may have to convert this project to a different version of Unity, since it is set for Unity 2017.3.1f1.

1. The scene is located in Assets > Resources > _Scenes > ExampleScene.unity
2. To download it from the GitHub, follow the instructions in Section 3 Downloading the VAIF Package.
3. To open the scene in Unity, click on File > Open Scene (or, CTRL + O), and navigate to Assets > Resources > _Scenes. Double click on ExampleScene.unity, and wait for Unity to load the Scene.

6. Create a Timeline

Before starting this section, it is recommended to begin designing your agents (see Section 7 Create Your Agent). A timeline consists of one or more conversations and represents the sequence of events in your scene and when they will be played.

1. Drag and drop the Timeline prefab (found in Assets > Resources > _Prefabs > Timeline.prefab) to the Hierarchy as its own parent GameObject:

2. Now that you've created a Timeline, you can add Conversations to it. You must create a Conversation to add Events to play a scene.
3. Be sure that you only have ONE Timeline Gameobject in the Hierarchy window.

Tutorial Video: VAIF 3

6.1. Create a Conversation

1. You need to create a Conversation in order to add Events to play. Conversations are groups of events where players (or users) will interact with ECAs or agents.
2. Click on the Timeline GameObject in the hierarchy.
3. In the Conversation IM (Script) component, click on the "**Add Conversation**" button. This will add the conversation and the appropriate scripts needed for it. The conversation will be added to the hierarchy of Timeline.

   Developers are working on a feature of different types of Conversations.
4. Now that you've created a Conversation, you can add Events to it.
5. To add agents, see Section 7 Creating Your Agent.

Tutorial Video: VAIF 4

6.2. Types of Events

Before adding Events to a Conversation, scene designers must understand the different types of Events. Each list contains different requirements in order to run.

General Events - These events do not require to specify the Next Event to play in the Timeline:

1. Animation
2. Dialog
3. Move
4. RotateTo
5. Wait

Specific Events - These events require different “Next Event” specifications in order to play in the Timeline. These are helpful for branching out inside the interaction that are conditional to user’s responses:

1. Response
2. Wildcard

Tutorial Video: VAIF 5

6.3. Adding Events to a Conversation

1. You need to create a Conversation in order to run Events. See Section 6.1 Create a Conversation.
2. Click on the Conversation GameObject in the hierarchy to which you want to add events.
3. In the Inspector window, look at the “Event IM (Script) component”.
4. You **MUST** select the Agent that the Event will be attached to. For example, in an Animation event, choose the Agent that will perform this event. Do this by dragging and dropping the Agent (from the Hierarchy) to the Agent field in the EventIM component.
5. Click “**Add Event**” by choosing the type of event to add in the “Event IM (Script) component”.
6. Clicking "Add Event" will add the event and the appropriate scripts needed for that event. The event will be added to the hierarchy of Conversation.

NOTE: Unity does **NOT** change windows to see this added event; you must click on the event in the hierarchy of the Conversation in order to view/edit the event.

7. Every type of event requires Settings and GameObjects to be assigned **before** run/play of a scene. See the Exceptions section for additional details on these additional settings:
    a. Next Event,
    b. Agent,
    c. Want In Range, and
    d. Want Looked At.

Tutorial Video: VAIF 7

6.3.1. Exceptions: General Events

1. If the event is an Animation, you **must** define the animation to use (this is from the list of animations in that Agent's “AnimationManager”; it must be in the _resources folder).
2. If the event is a Dialog, you **must** define the dialog to use (this is from the list of dialogs that must be in the _resources folder).
3. If the event is the last event of the Conversation, it's “Next Event” can be empty, but you must check the “Is Last Event” field.

6.3.1.1. Editing an Event’s “Next Event” Field

The Next Event is saved as a GameObject. This means you will have to drag and drop into this field the event you want to play after the event you are editing completes.

1. Find the next event to play (after this one) in the hierarchy of the Conversation.
2. Drag and drop the event to the field of Next Event.
3. Click on the box; it should highlight in yellow (in the hierarchy) which event is this event's Next Event (the one you dragged).

6.3.1.2. Editing an Event's "Agent" Field

The Agent is saved as a GameObject. This means you will have to drag and drop into this field the agent which will be targeted for this event.

1. Find the agent in the hierarchy window.
2. Drag and drop the agent to the field of Agent.
3. Click on the box; it should highlight in yellow (in the hierarchy window) which agent is this event's Agent (the one you dragged).

6.3.1.3. Editing an Event's "Want In Range" Field

An event's "Want In Range" is a required setting. This setting, like the Want Looked At, has three possibilities that you must choose from as a condition to starting this event (in combination with Want Looked At):

- TRUE (the user **must** be near the agent referenced in Agent),
- FALSE (the user **must not** be near the agent referenced in Agent), or
- DONTCARE (the event will play regardless of the user's location).

6.3.1.4. Editing an Event's "Want Looked At" Field

An event's Want Looked At is a required setting. This setting, like the Want In Range, has three possibilities that you must choose from as a condition to starting this event (in combination with Want In Range):

- TRUE (the user must look at the agent referenced in Agent),
- FALSE (the user **must not** look at the agent referenced in Agent), or
- DONTCARE (the event will play regardless of the user's gaze direction).

Tutorial Video: VAIF 6

6.3.2. Exceptions: Specific Events

6.3.2.1. Response Events

The settings for a "Response" event are unique. The settings for the following fields are the same as General Events:

- Agent
- Want In Range
- Want Looked At
- Is Last Event

This event **does not** fill the "Next Event" field until run time. The following fields require additional settings:

1. Response Items - A list of the phrases which will be recognized as a response to a previously played Dialog event.
    a. Adjust the size. The Size of Response Items must match the Size of Jump IDs.
    b. Fill each of the Elements with a phrase.
    c. For each Response Item Element, you must set a Jump ID (see below).
    d. For example, a Response Item like 'no' at Element 0 may jump to some Animation event. This means that you will drag and drop the Animation event to Element 0 of the Jump ID list.
    NOTE:
        i. Adding a 'yes', adds the following phrases as similar recognitions:
            1. yeah / uh-huh
            2. yup/yep / okay / sure
            3. sounds good / that sounds good
            4. cool / alright
        ii. Adding a 'no', adds the following phrases as similar recognitions:
            1. nay / nah / nope
            2. no way
            3. no, thank you no, thanks
            4. i don't think so
        iii. Adding an 'I don't know', adds the following phrases as similar recognitions:
            1. I'm unsure / I'm not sure
            2. I dunno / maybe / I guess
            3. I have no idea / I have no clue
        iv. If the user asks a question requesting a repeat, such as any of the phrases listed below, the system will automatically replay the previous Dialog event:
            1. what / huh
            2. I don't know what you said
            3. what was that / what did you say
            4. say that again / say it again / come again
            5. can you repeat that / can you repeat what you said / can you repeat what you just said / can you repeat the question
            6. can you say that again / can you say it again
            7. repeat it / repeat that / repeat yourself
            8. repeat please / repeat what you said / repeat what you just said / repeat the question
2. Jump IDs - A list of events to choose which event plays next when a Response Item is uttered by the user (and recognized by the system).
    a. Adjust the size. The Size of Jump IDs must match the Size of Response Items.
    b. Drag and drop the event from the hierarchy window to this field to play this event when its matching Response Item is uttered and recognized.
    c. Each Jump ID must be set for a Response Item (see above).

d. Like in the above example, you will drag and drop an Animation event to Element 0 of the Jump ID list to be the Next Event when 'no' (stored in Element 0 of the Response Items list) is uttered and recognized.

3. Timeout - The time (in seconds) to wait for an utterance by the user before jumping to the "Timeout Jump ID" event.
4. Timeout Jump ID - The event to play when the time in "Timeout" is exceeded.
5. Misrecognitions

   Developers are working on this feature.
6. Base Case Jump ID

   Developers are working on this feature.

Tutorial Video: VAIF 9

6.3.2.2. Wildcard Events

The settings for a "Wildcard" event are unique. The settings for the following fields are the same as General Events:

- Agent
- Next Event
- Want In Range
- Want Looked At
- Is Last Event

This event **does** fill the "Next Event" field. The following fields require additional settings:

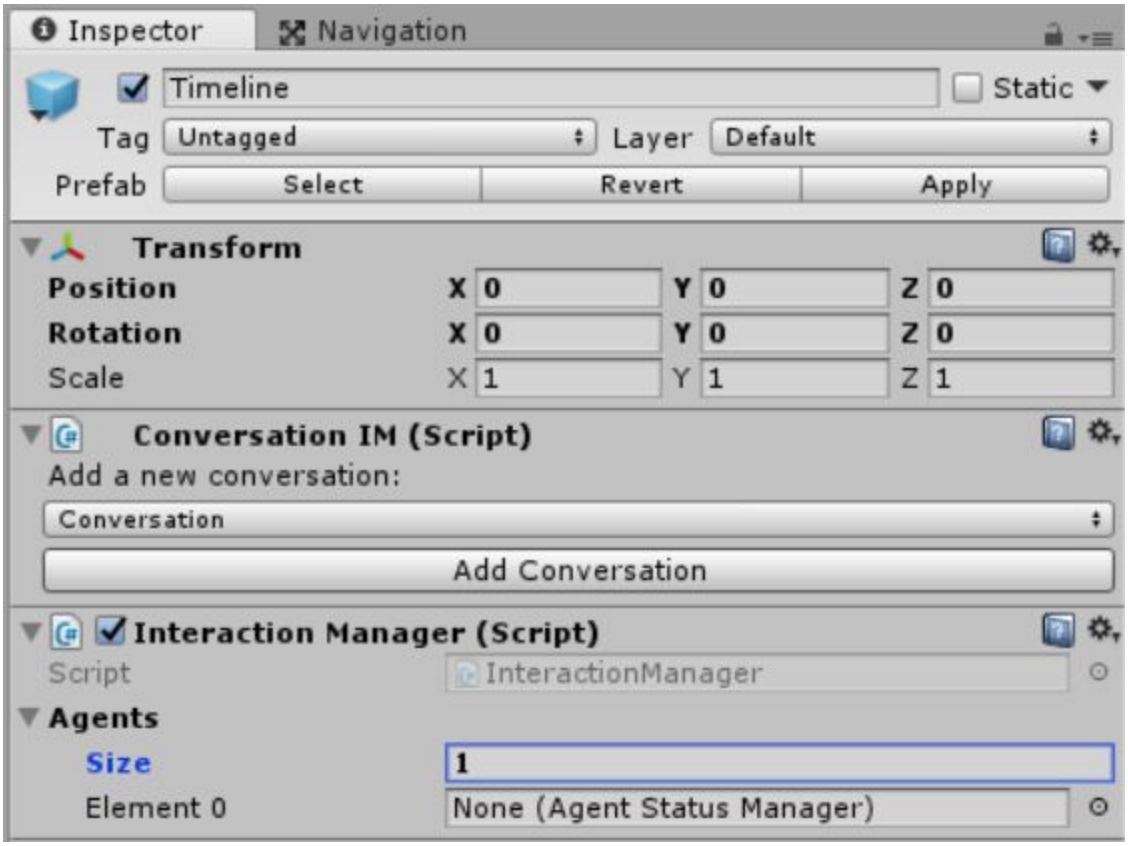
1. Timeout - The time (in seconds) to wait for an utterance by the user before jumping to the "Next Event".
2. Annotation

   Developers are working on this feature.

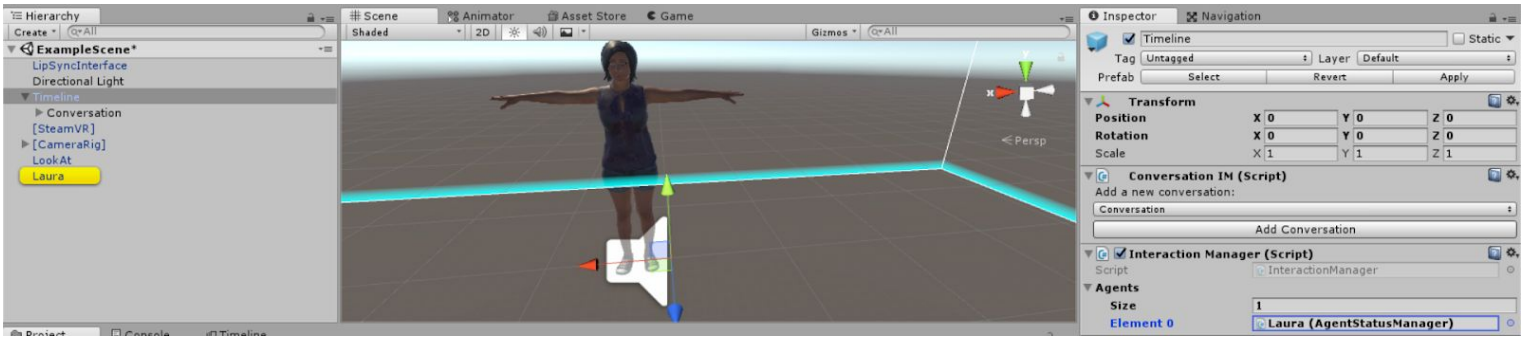
Tutorial Video: VAIF 8

7. Creating Your Agents

NOTE: After adding all your agents, you MUST drag and drop each of the Agents in the Hierarchy into the Interaction Manager (Script) component of the Timeline GameObject. First, you must modify the size field to the number of Agents in the entire Timeline:



The result will look something like this (depending on how many Agents you have):



7.0. Import the Agent

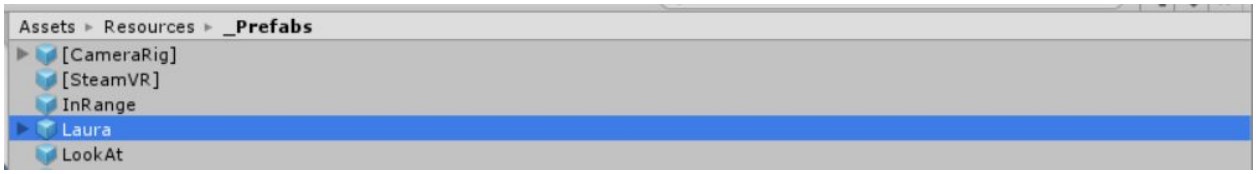
Two Agents are included in the VAIF asset. To customize any of the agents, you can modify the materials of the model to change the agent's appearance in hair or clothes.

- There are prefabs of these agents (equipped with all of the agents' scripts and components) that can be found in **Assets > Resources > _Prefabs**. The two Agents available are "Laura" and "Taema". See 7.1. Using a VAIF-provided Agent (prefab)
- Also available are the agents' 3D Object files (.fbx) can also be found in the **Assets > Resources > _Agents**. See 7.2. Adding the Managers to an Agent (non-prefab) to add all of the components and managers to an Agent GameObject.

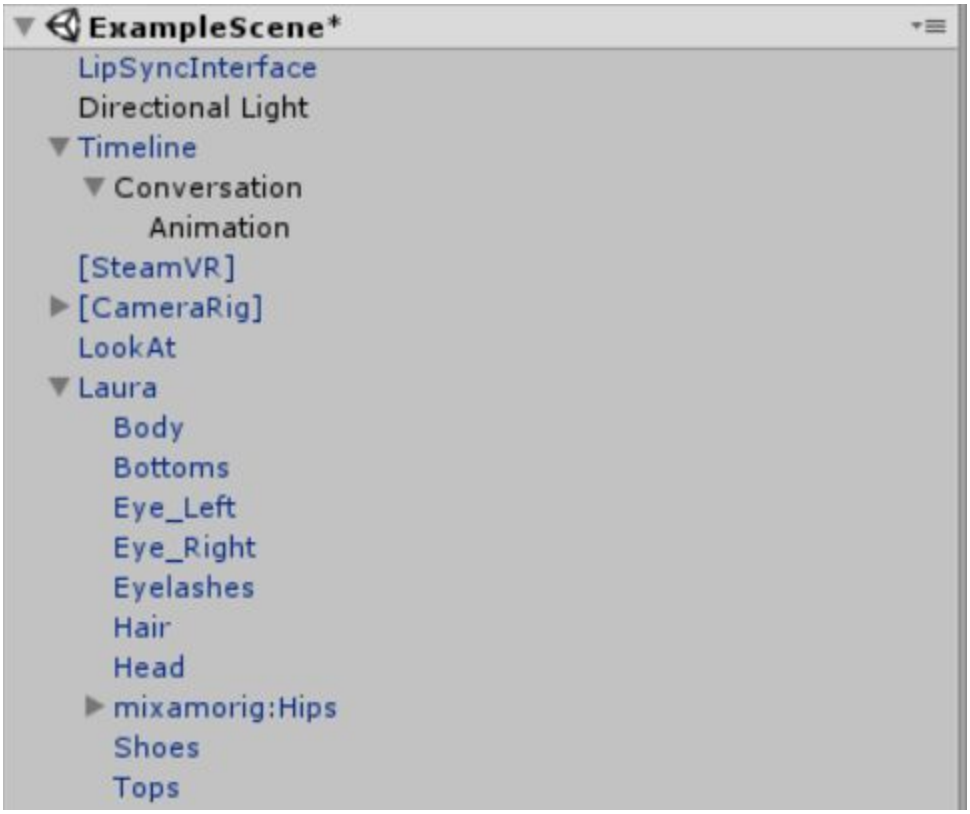
7.1. Using a VAIF-provided Agent (prefab)

You can use one of the VAIF-provided Agents, available as prefabs for two agents: Laura and Taema.

1. Both of these Agents' prefabs can be found in Assets > Resources > _Prefabs.



2. Choose either Laura or Taema (or both) to drag and drop to your scene's Hierarchy.

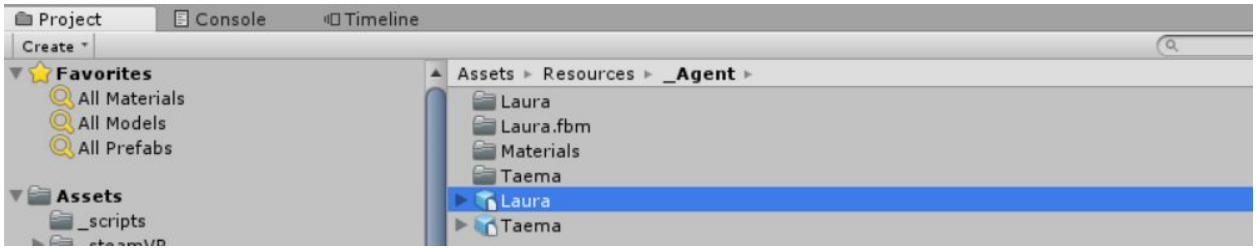


3. Verify that Laura (or Taema) has the correct components and properties matching all those found in Section 7.2. Sometimes, Unity may not find the correct script, tag, or the Agent may not be rigged as a Humanoid.
4. Proceed to 7.3 Integrating the VR Headset.

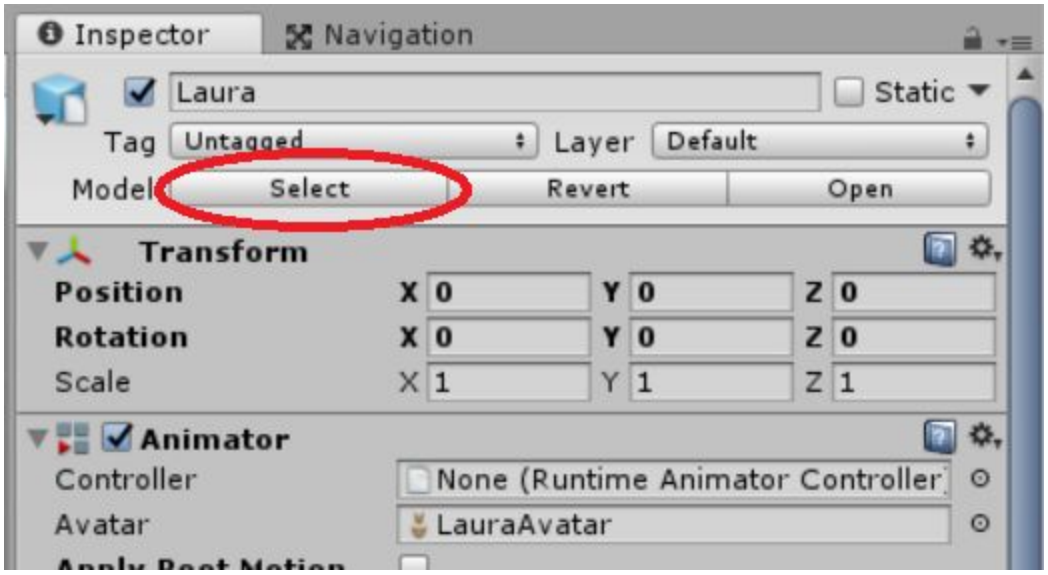
7.2. Adding the Managers to an Agent (non-prefab)

Your **Agent must be rigged** as a humanoid 3D Object (.fbx). You can practice this section on one of the VAIF-provided Agents 3D Object Files (found in Assets > Resources > _Agent, and you can choose Laura.fbx or Taema.fbx).

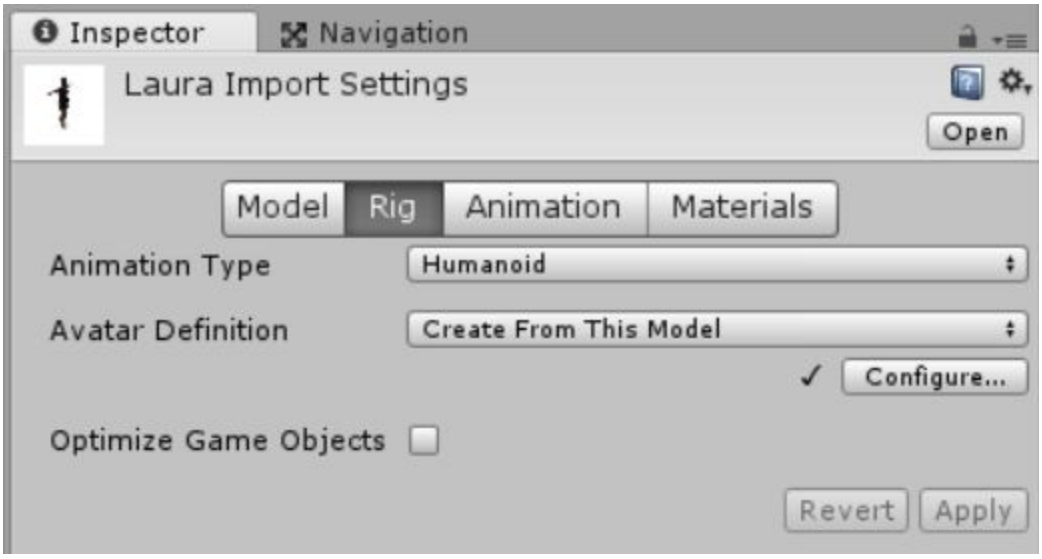
1. Drag and drop the Agent from the Resources > _Agent folder in the Console (see image below) to the Hierarchy window; e.g, drag and drop the Laura Object (.fbx) file to the Hierarchy:



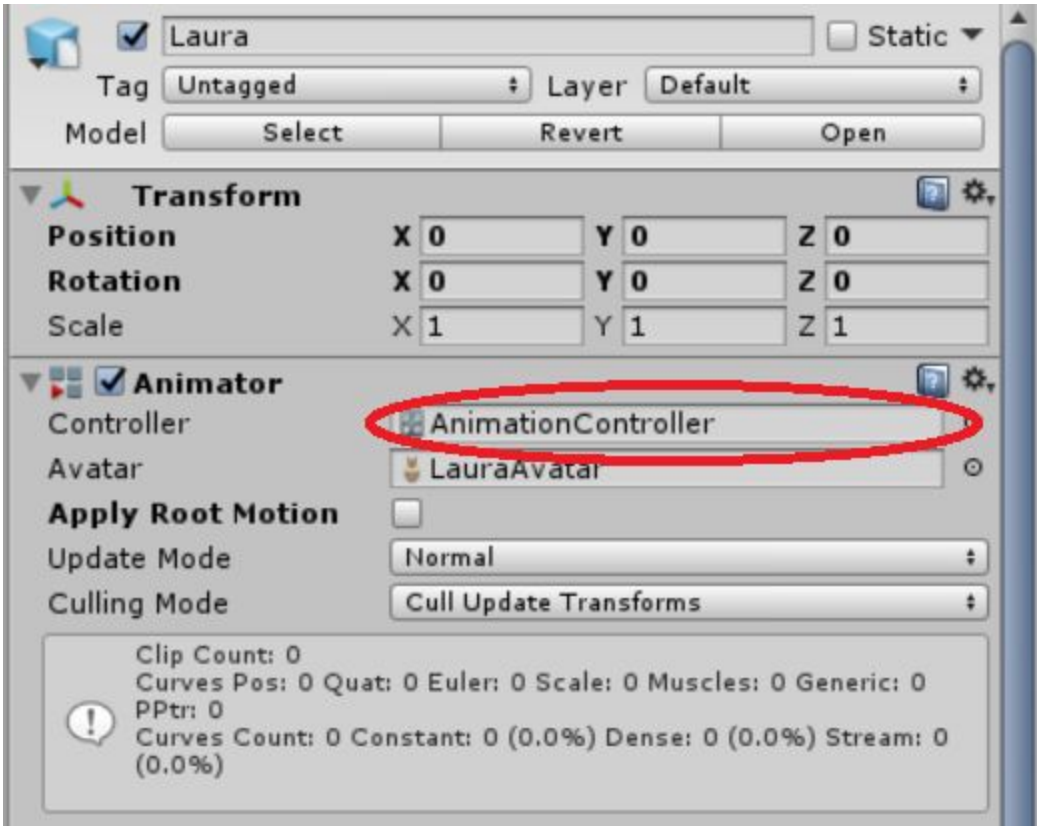
2. Click on the Agent in the Hierarchy window. In the Inspector window with this Agent selected, click on Select (see image).



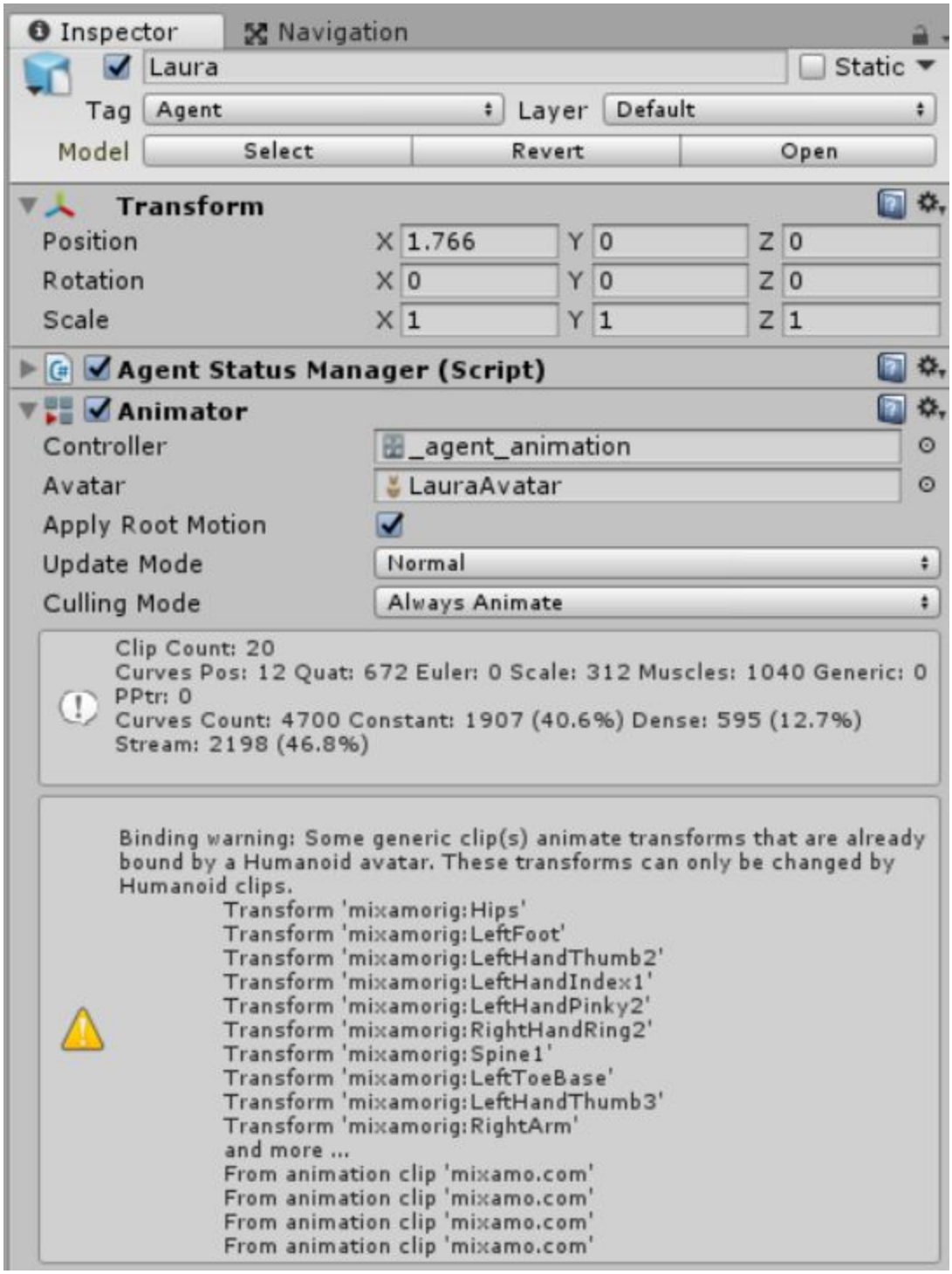
3. Click on “Rig”, and in the “Animation Type”, make sure “Humanoid” is selected. Click “Apply” if you made the change to Humanoid, and wait for Unity to look the changes.



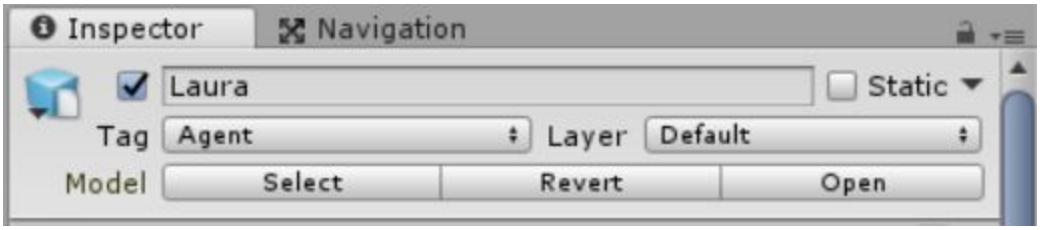
4. For every Agent you add, drag and drop the _agent_animation (found in Assets > Resources > _Animations) to the Controller of the Animator component in the agent’s Inspector window:



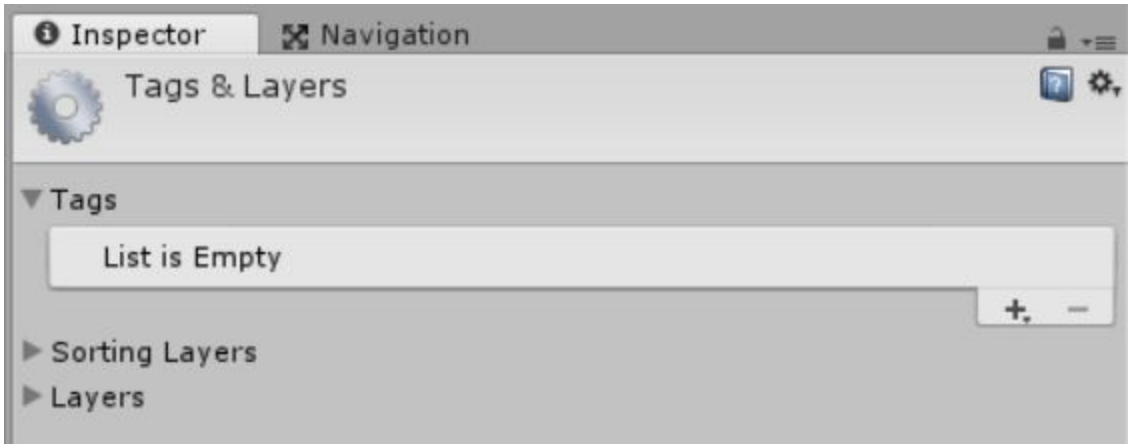
***NOTE**:* You may get the following list of warnings from Mixamo's animation clips, it is okay to ignore these warnings:



5. Check the “Apply Root Motion” checkbox.
6. For each Agent you add, the Agent must be Tagged as “Agent”. If the “Agent” tag does not exist in the drop-down menu, you must choose “Add tag” at the bottom of the list of options.

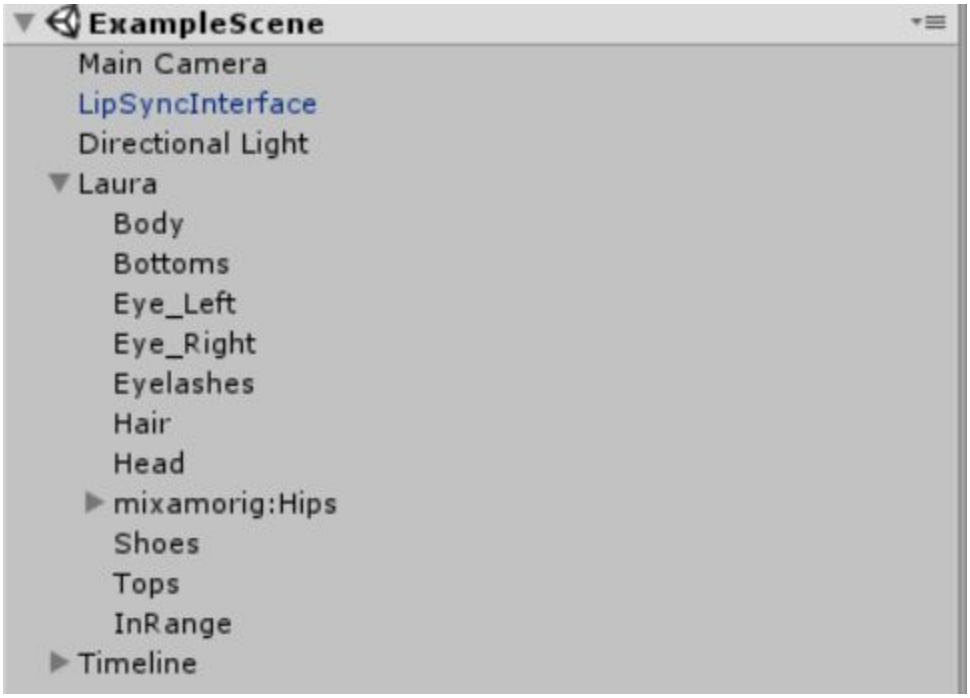


7. When you select “Add tag”, this window opens:

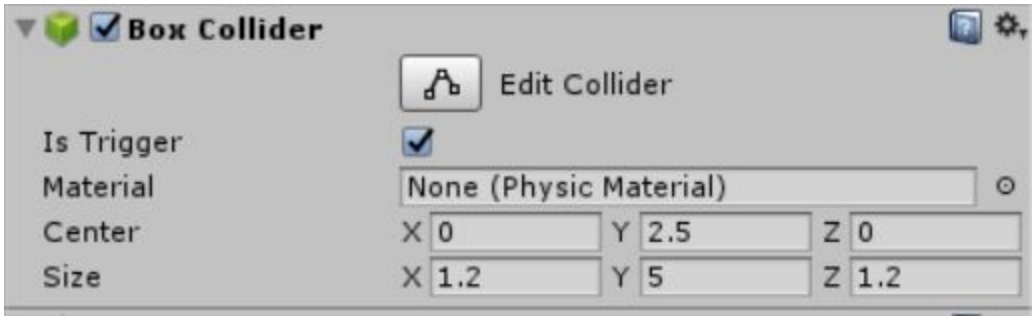


8. Next, you can add the following Tags to the list of tags:

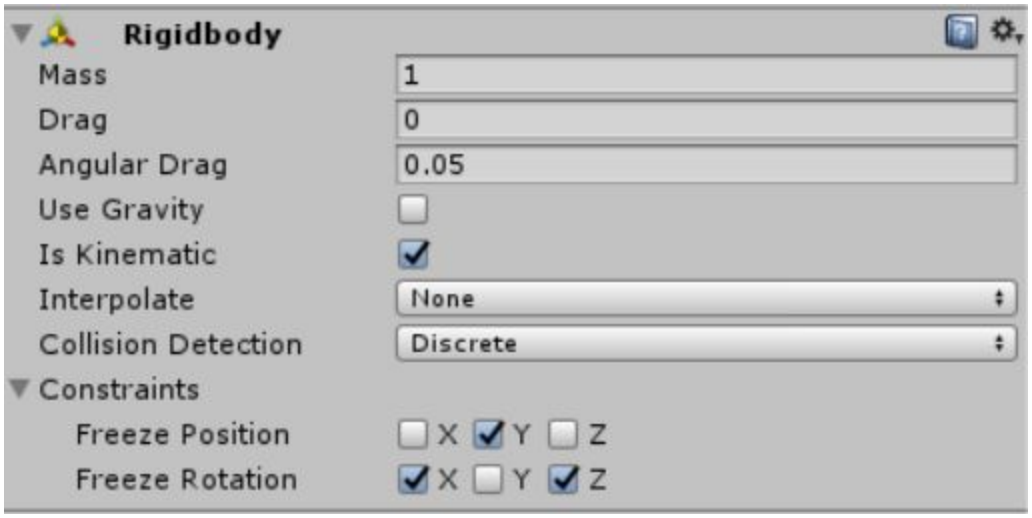
9. For the Agent, select the Agent tag. You will use the other tags later for other components or GameObjects.
10. To the Agent GameObject in the Hierarchy, drag and drop the InRange Prefab (found in Assets > Resources > _Prefabs) to **each of your Agents**. The resulting Hierarchy of the Agent will look like this:



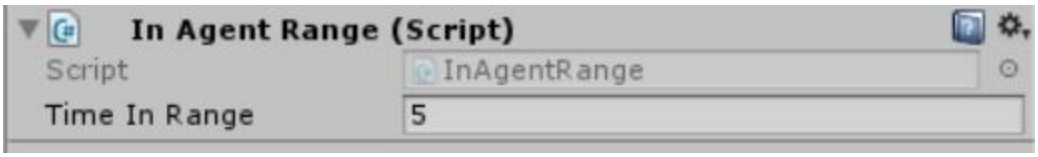
11. Verify that the InRange GameObject matches these settings:
    a. Box Collider; modify the component to this data:



    b. Rigidbody (not the 2D version); modify the component to:



c. In Agent Range (Script). This script will trigger the "In Range" flag when the user (or VR headset) is within the agent's box collider for some Time In Range. Modify the Time In Range to be 5 seconds.



NOTE: Your Agent will be composed of several managers. You will add all of these managers in Sections 7.2.1 - 7.2.12 to the Agent GameObject as components of the GameObject. For more experienced users, depending on the goal of your application, you may want to add all managers, or omit some managers depending on which features you would like to include. The order of these managers are not important.

7.2.1. Add the Agent Status Manager

This keeps track of what the agent is currently doing. Below are the states and when they are marked true:

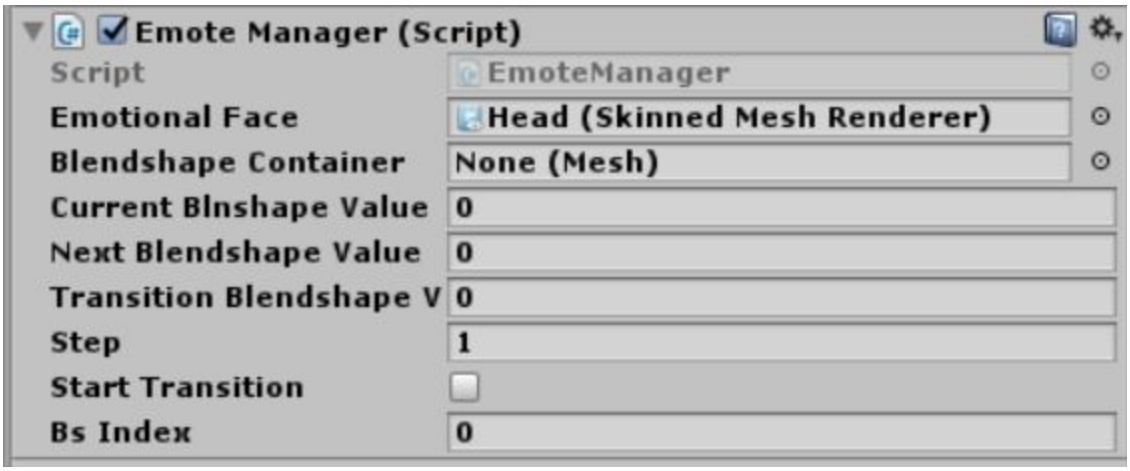
- *Speaking* - The agent is playing a dialog.
- *Listening* - The agent is listening to the user
- *Waiting* - The agent is waiting until the next event
- *In Range* - The user is close enough to the Agent
- *Moving* - The agent is currently moving towards a given target location
- *Looked At* - The agent is being looked at by the user

7.2.2. Add the Move Manager

Dictates to the agent where to move next and calculates which animation to play with.

7.2.3. Add the Emote Manager (Future Implementation)

With facial blendshapes, your agent will be able to display specific facial expressions as controlled by the emote event. You **MUST** still add this Script component to the Agent GameObject. You will also need to **drag and drop** the Agent's Head from the Hierarchy to the Emotional Face field of the Emote Manager Component:



7.2.4. Add the Dialog Manager

Grabs the audio file from the dialog event and plays the audio file with the correct phonemes to this agent.

7.2.5. Add the Animation Manager

This manager grabs all the animations possible from the animator attached to the agent in alphabetical order. So when an Animation event occurs, this manager plays the correct animation.

NOTE: You must add all animation clips (rigged as Humanoid, see below) to the Assets > Resources > _Animations folder in order to use them in Animation Events.

7.2.6. Add the Look At Collision Script

A box collider that is attached to the Player to keep track if the Player is looking at the Agent. **Modify** the "Time on Target" field of this component to the number of seconds you want to trigger the "lookedAt" boolean value. Normally, 1.5 is a long enough time to be looking at the agent.

7.2.7. Add the Personalities Script (Future Implementation)

Eventually, Agents will be able to represent themselves based on the type of personality you instruct. You MUST still add this Script component to the Agent GameObject.

7.2.8. Add the Emotions Script (Future Implementation)

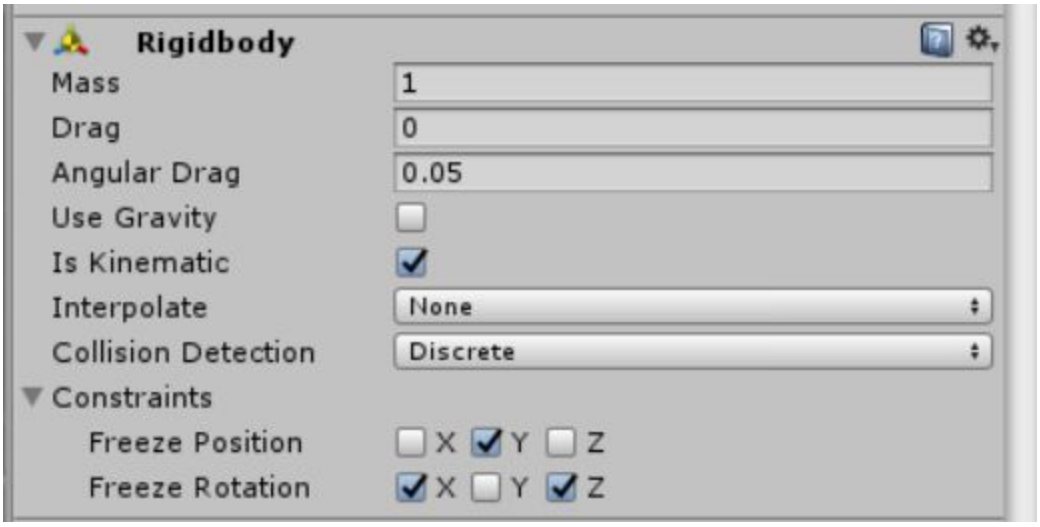
This manager keeps track of display of emotions for the agent. You MUST still add this Script component to the Agent GameObject.

7.2.9. Add the Gaze Manager (Future Implementation)

Make the agent look at camera (participant inside unity scene) and other specific objects inside the unity scene. **DO NOT ADD** this Script component to the Agent GameObject.

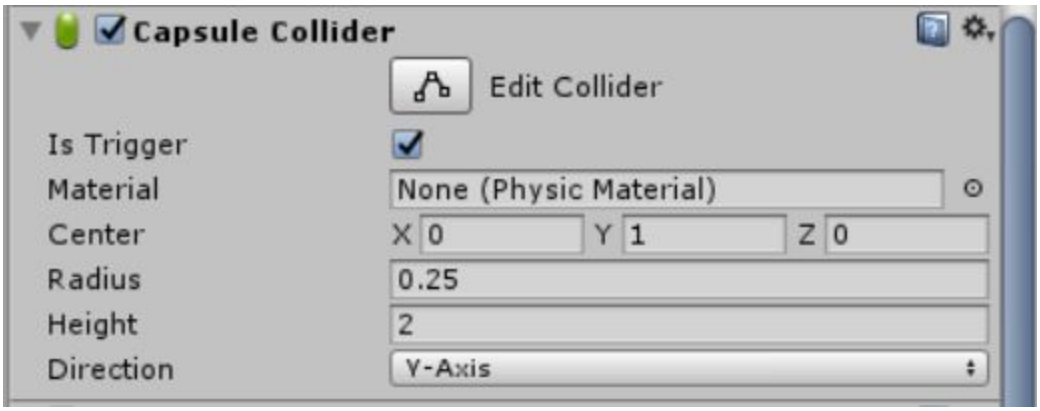
7.2.10. Add the Rigidbody Component

Make sure you don't pick the 2D version. This component gives weight to the agent in the scene's physical space. Modify this component to look like the following:



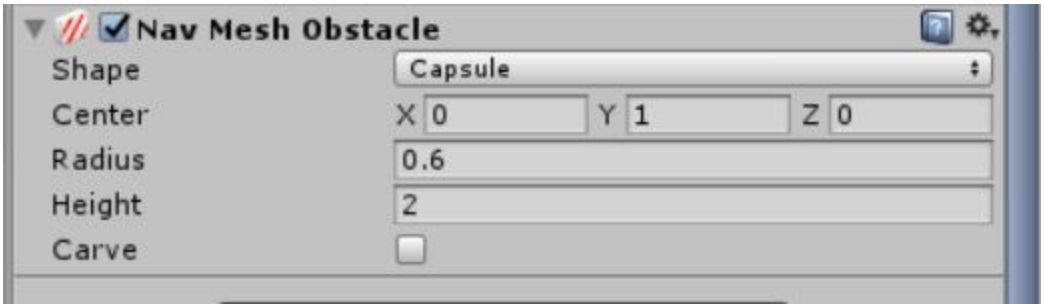
7.2.11. Capsule Collider

Allows this Agent to detect when the user (VR headset) is in range. Modify this component to look like the following:



7.2.12. Nav Mesh Obstacle

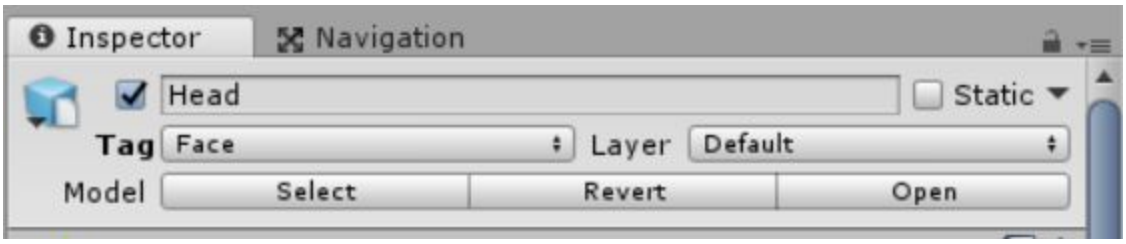
Allows this Agent to detect which obstacles to avoid. Modify this component to look like the following:



7.2.13. Adding the Managers to the Agent's Head

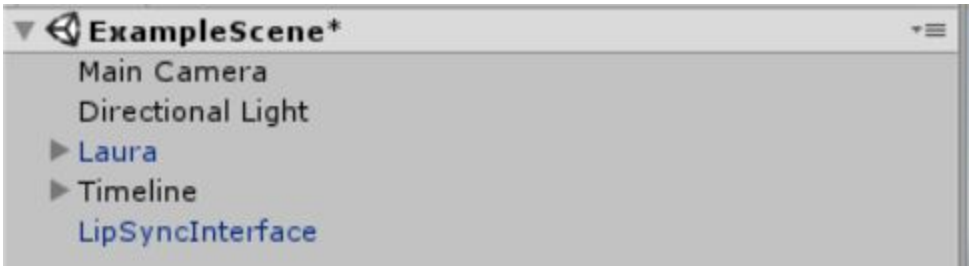
Your Agent's face will be composed of several managers in order to make the agent speak. Modify the Head component of the Agent by expanding the Agent in the Hierarchy, and clicking on Head:

Tag the Head with the Face tag you created:



7.2.13.1. Add the LipSyncInterface Prefab

1. Find it in the folder in Assets > Prefabs
2. Drag the prefab (the blue object) LipSyncInterface into the Hierarchy. It does not have to be a part of any GameObject:



7.2.13.2. Add the OVRLipSyncContext Script

1. Add a component to the face of the Agent. You can do this by clicking “Add Component” and search for the “OVR Lip Sync Context” Script
2. This will add two components: OVR Lip Sync Context and Audio Source. You will leave the **Audio Source component unmodified**.
3. In the OVR Lip Sync Context
    a. Check the “Show Visemes” checkbox.
    b. Check the “Audio Loopback” checkbox.

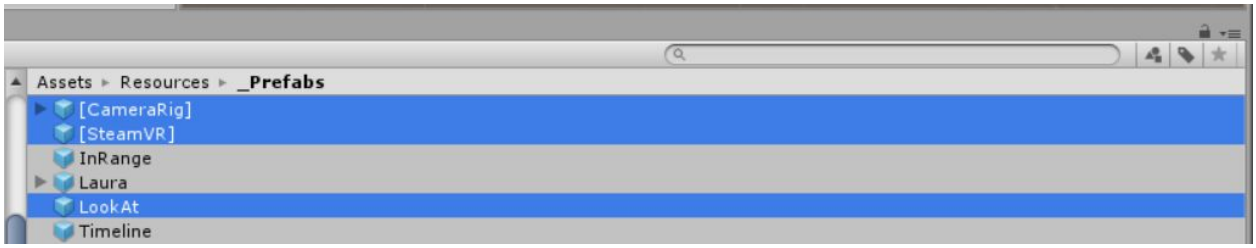
7.2.13.3. Add the OVRLipSyncContext Morph Target Script

1. Add a component to the face of the Agent. You can do this by clicking “Add Component” and search for the “OVR Lip Sync Context Morph Target” Script
2. Only **one modification** is needed for this component: from the Hierarchy, drag the face of the agent (the Head under the Agent in the Hierarchy) into the Skinned Mesh Renderer field of this component.

7.3. Integrating the Vive VR Headset

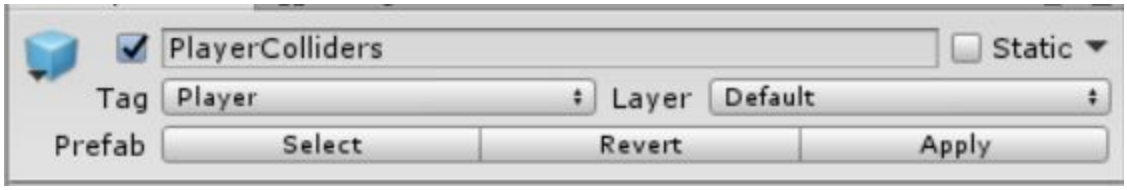
In order to use the SteamVR toolkit, you will need to add some more prefabs to your scene.

1. In the Project window, navigate to the Assets > Resources > _Prefabs directory.
2. Select the following Prefabs, and drag and drop them to the Hierarchy of GameObjects:

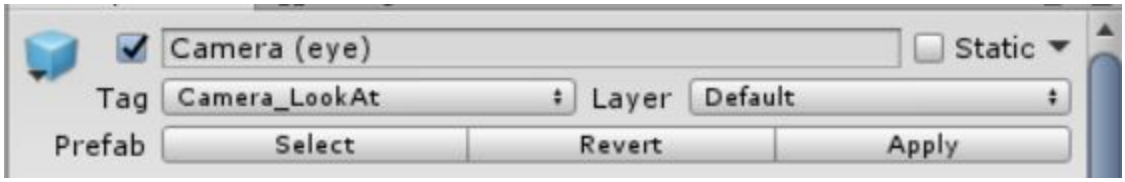


    a. [CameraRig] - part of the _steamVR library, needed to recognize the headset
        i. Expand this in the Hierarchy and make sure that [CameraRig] **AND** PlayerColliders are **Tagged as Player**:

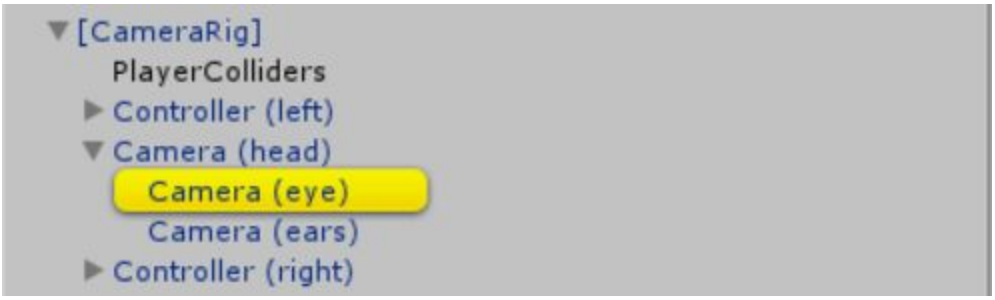
NOTE: if you cannot find these Tags in the list of Tags, follow steps 6-8 of 7.2 Adding the Managers to an Agent (non-prefab)

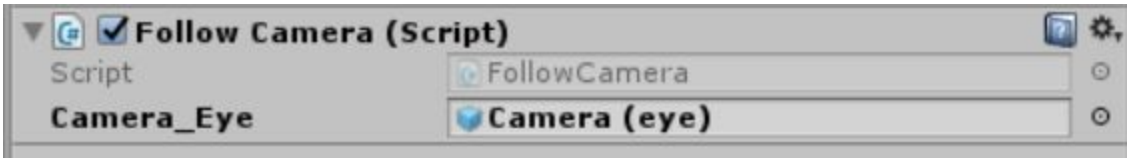


ii. Expand the Camera (head) in the Hierarchy and make sure that Camera (head) **AND** Camera (eye) are **Tagged as Camera_LookAt**:

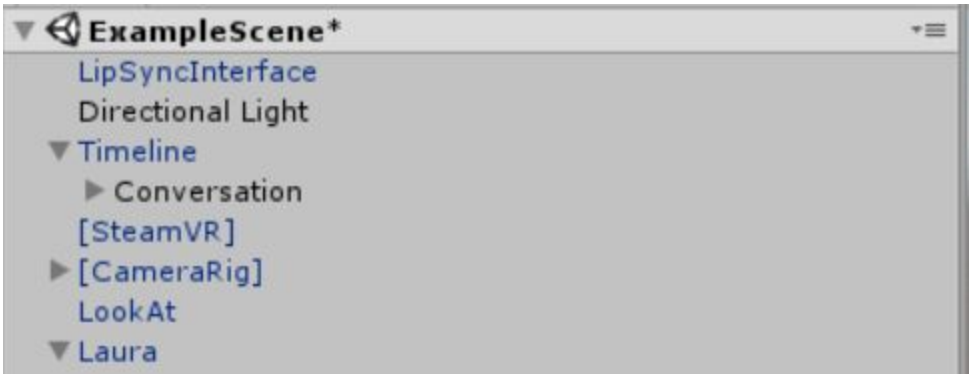


b. [SteamVR] - part of the _steamVR library, needed for use with HTC Vive
   i. No modifications are needed for this prefab
c. LookAt - needed to recognize where the VR headset is looking at in the scene
   i. Modify the Camera_Eye field by dragging and dropping the Camera (eye) GameObject from the Hierarchy to the Camera_Eye field.





The resulting Hierarchy will look like this:

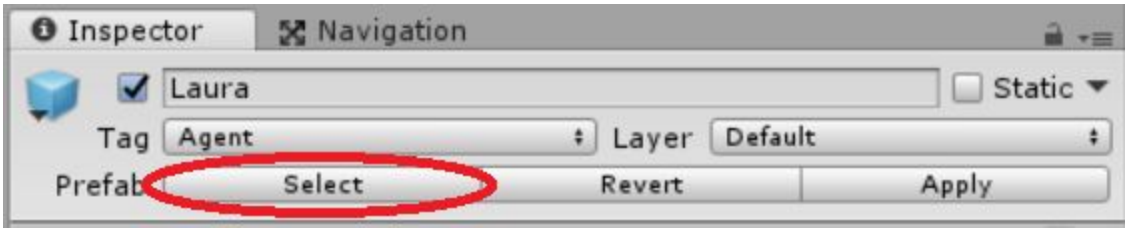


3. To the Agent GameObject in the Hierarchy, drag and drop the InRange Prefab (found in Assets > Resources > _Prefabs) to **each of your Agents**. (This was done in step 7.2.10, if you manually created your Agents).

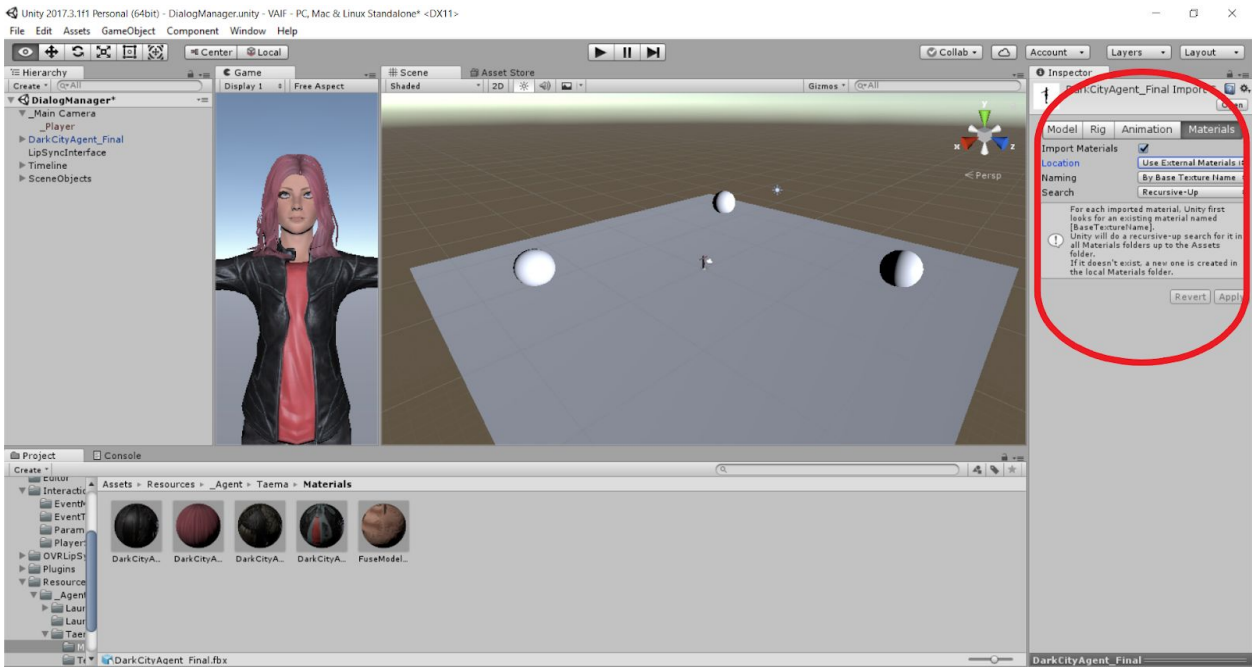
8. Troubleshooting

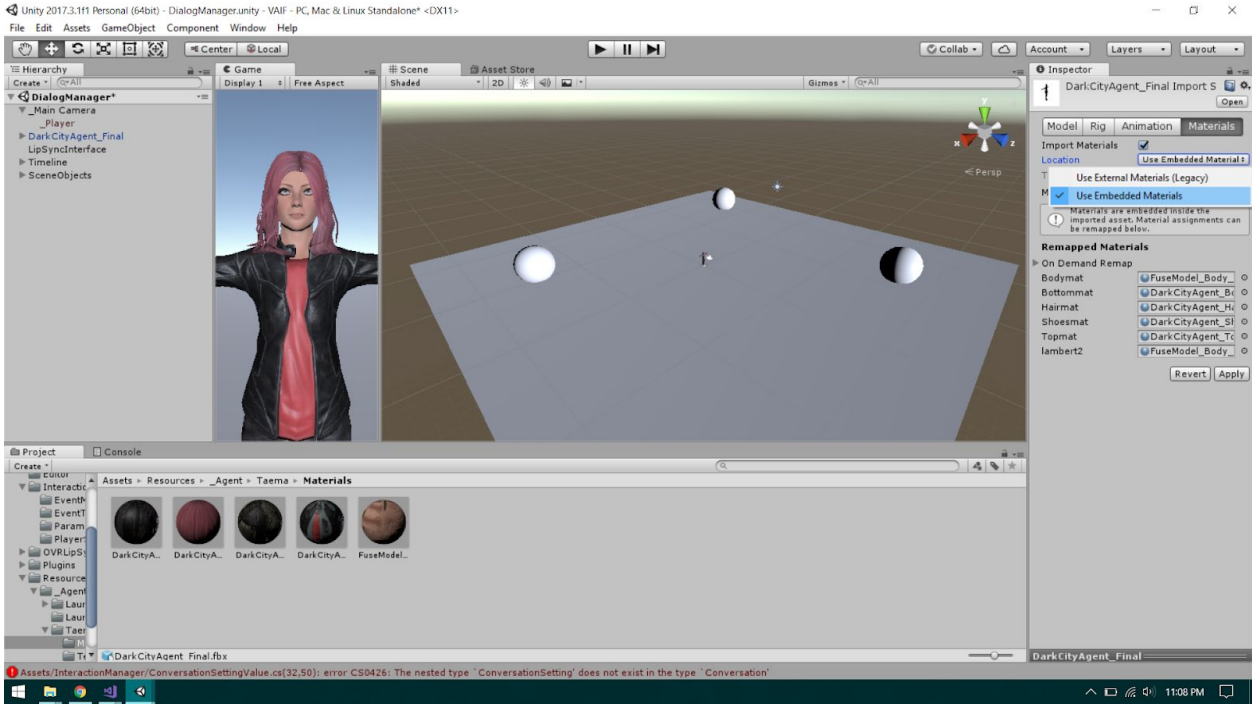
1. I opened the project and my Agents look weird.

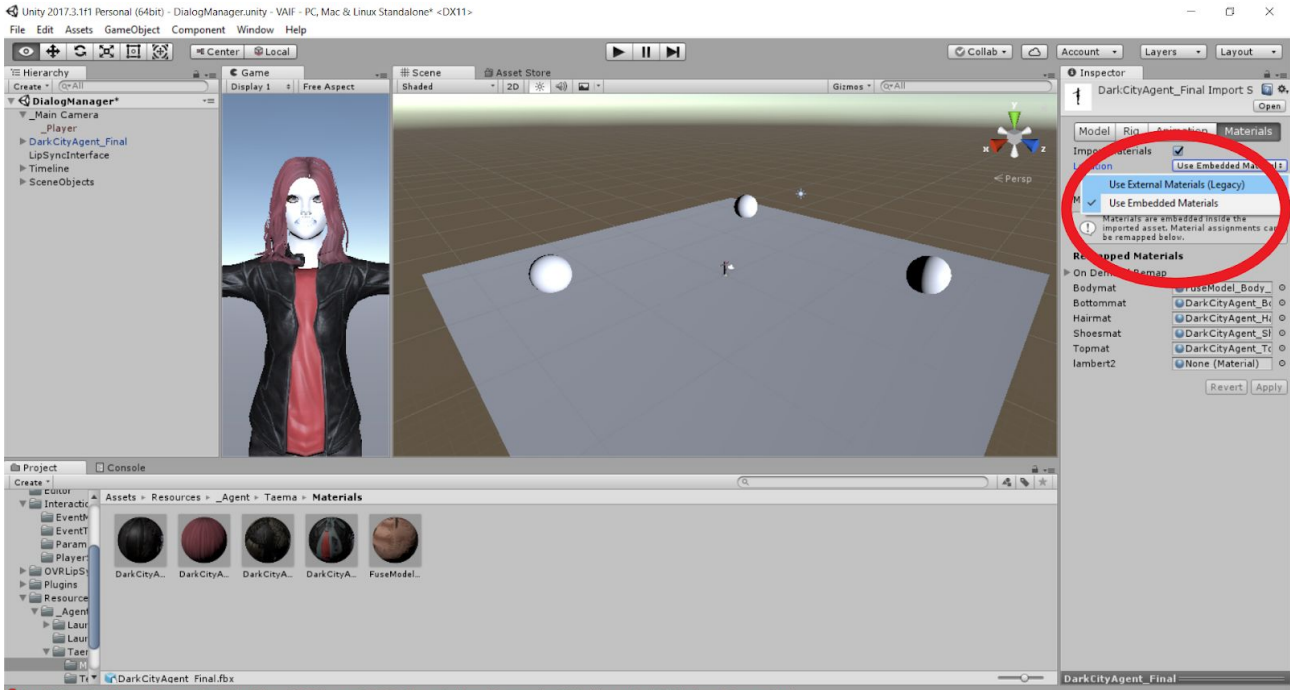
Fix: Click on the Agent in the Hierarchy, and click Select.



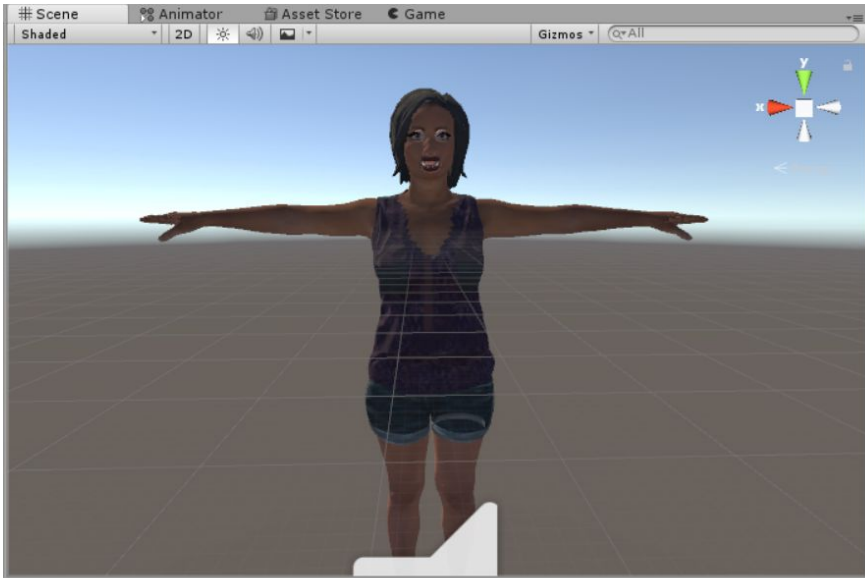
NOTE: If you are using the prefab version of the Agents, this fix will not work.



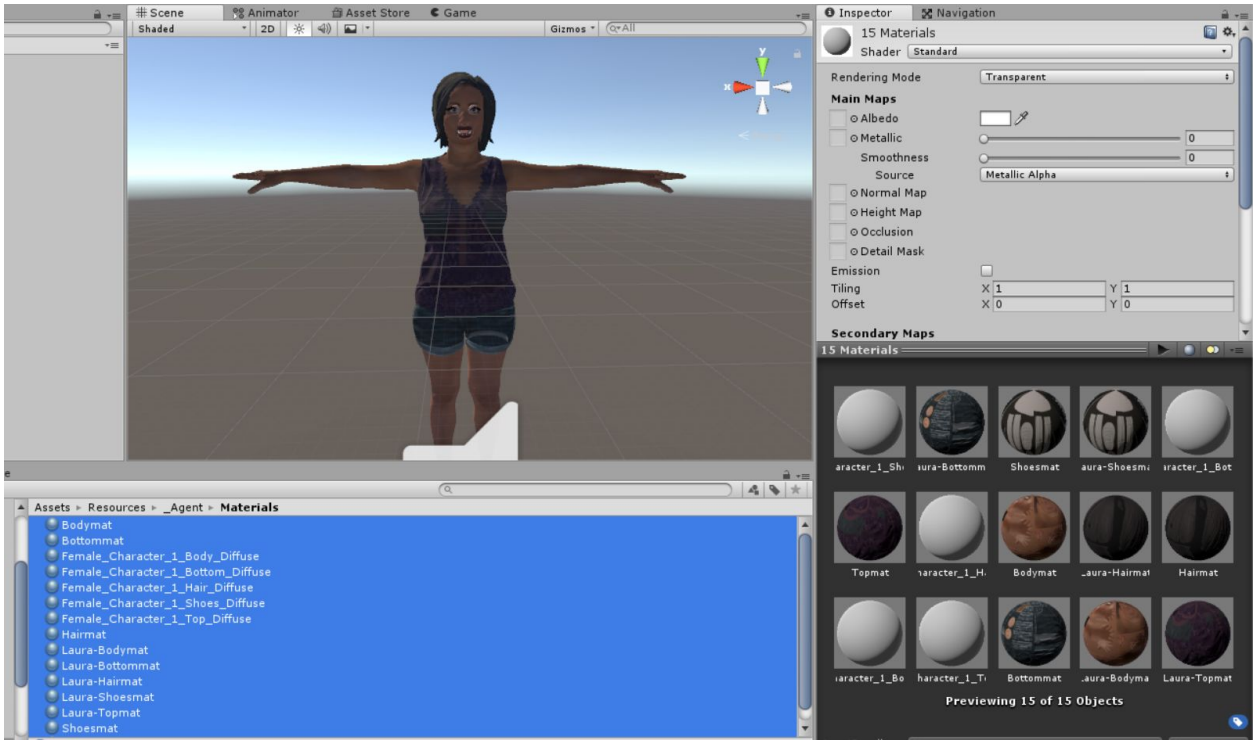

Unity 2017.3.1f1 Personal (64bit) - DialogManager.unity - VAIF - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
DialogManager*
_Main Camera
_Player
DarkCityAgent_Final
LipSyncInterface
Timeline
SceneObjects
Game
Scene
Asset Store
Inspector
DarkCityAgent_Final Import S
Model Rig Animation Materials
Import Materials
Use External Materials (Legacy)
Use Embedded Materials
Remapped Materials
On Demand Remap
Bodymat
Bottommat
Hairmat
Shoesmat
Topmat
lambert2
Revert Apply
Project Console
Assets ▸ Resources ▸ _Agent ▸ Taema ▸ Materials
DarkCityAgent_Final.fbx
Assets/InteractionManager/ConversationSettingValue.cs(32,50): error CS0426: The nested type `ConversationSetting' does not exist in the type `Conversation'
11:08 PM

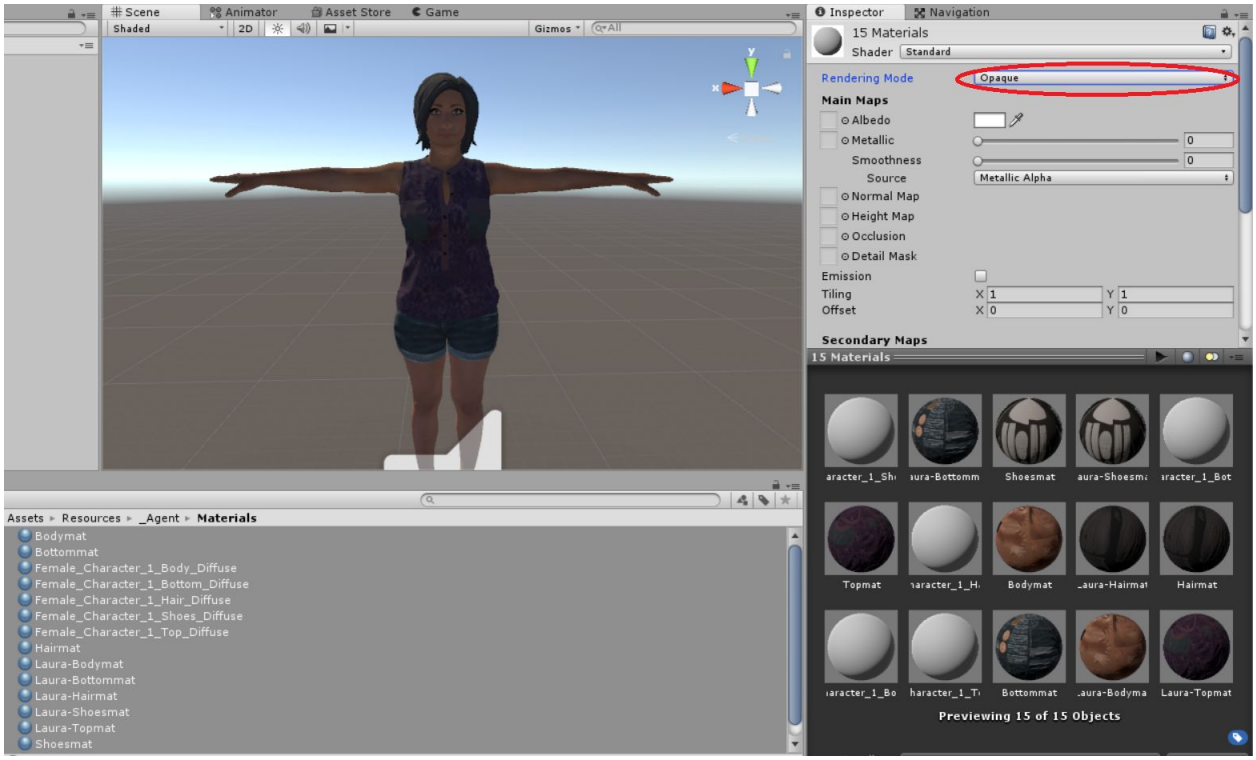

Unity 2017.3.1f1 Personal (64bit) - DialogManager.unity - VAIF - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
DialogManager*
_Main Camera
_Player
DarkCityAgent_Final
LipSyncInterface
Timeline
SceneObjects
Game
Scene
Asset Store
Inspector
DarkCityAgent_Final Import S
Model Rig Materials
Use External Materials (Legacy)
Use Embedded Materials
Bodymat
Bottommat
Hairmat
Shoesmat
Topmat
lambert2
None (Material)
Revert Apply
Project Console
Assets ▸ Resources ▸ _Agent ▸ Taema ▸ Materials
DarkCityAgent_Final.fbx

2. My Agents' materials look transparent:

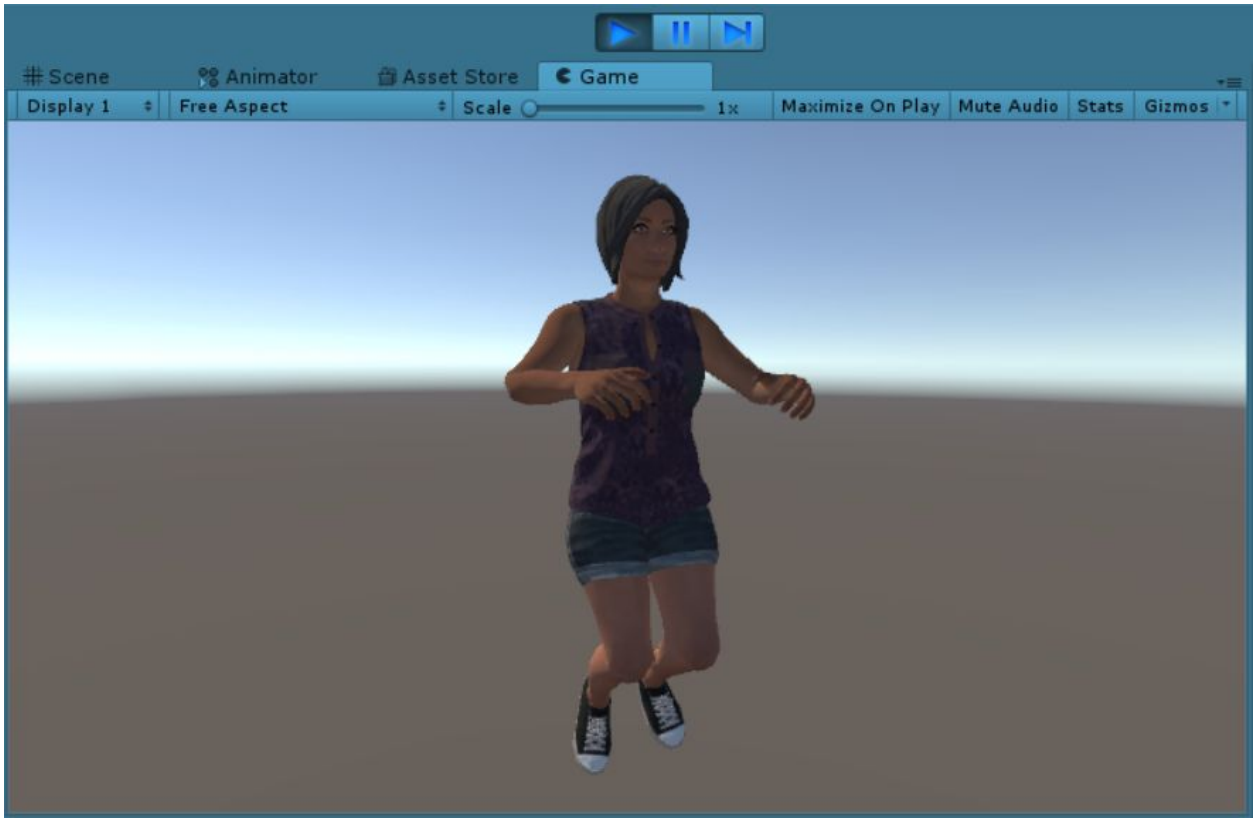


Fix: Select all Materials in the Assets > Resources > _Agent > Materials folder. Select "Opaque" in "Rendering Mode". You may have to do this for other materials found in other folders, depending on the agent and the organization your directories.

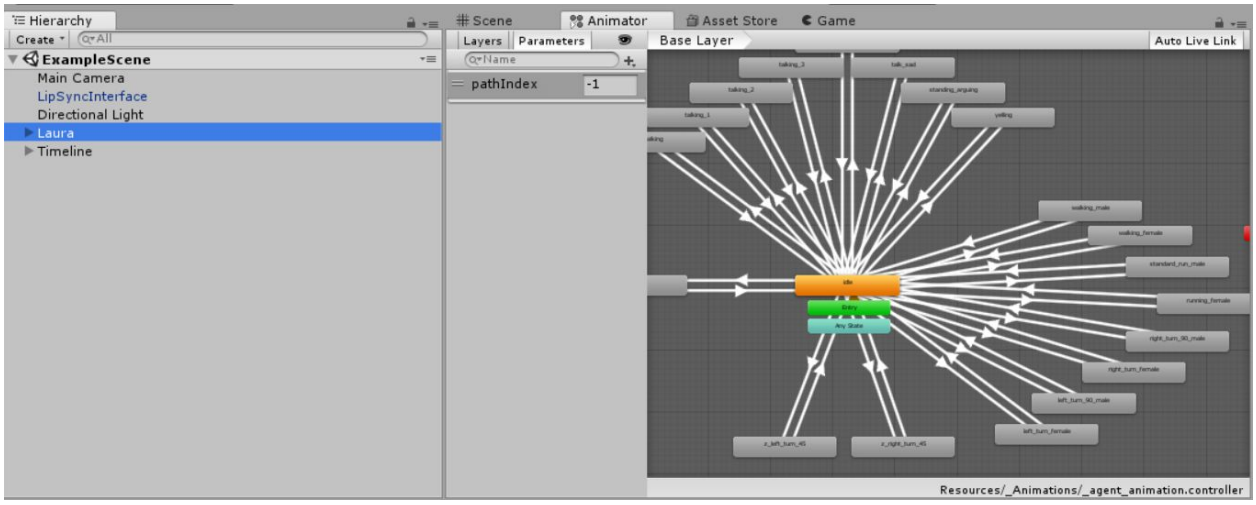



Scene
Animator
Asset Store
Game
Shaded
2D
Gizmos
Inspector
Navigation
15 Materials
Shader
Standard
Rendering Mode
Opaque
Main Maps
Albedo
Metallic
Smoothness
Source
Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling
Offset
Secondary Maps
15 Materials
Assets ▸ Resources ▸ _Agent ▸ Materials
Bodymat
Bottommat
Female_Character_1_Body_Diffuse
Female_Character_1_Bottom_Diffuse
Female_Character_1_Hair_Diffuse
Female_Character_1_Shoes_Diffuse
Female_Character_1_Top_Diffuse
Hairmat
Laura-Bodymat
Laura-Bottommat
Laura-Hairmat
Laura-Shoesmat
Laura-Topmat
Shoesmat
Topmat
Bodymat
Hairmat
Bottommat
Laura-Topmat
Previewing 15 of 15 Objects

3. My agents freeze in this position on play:

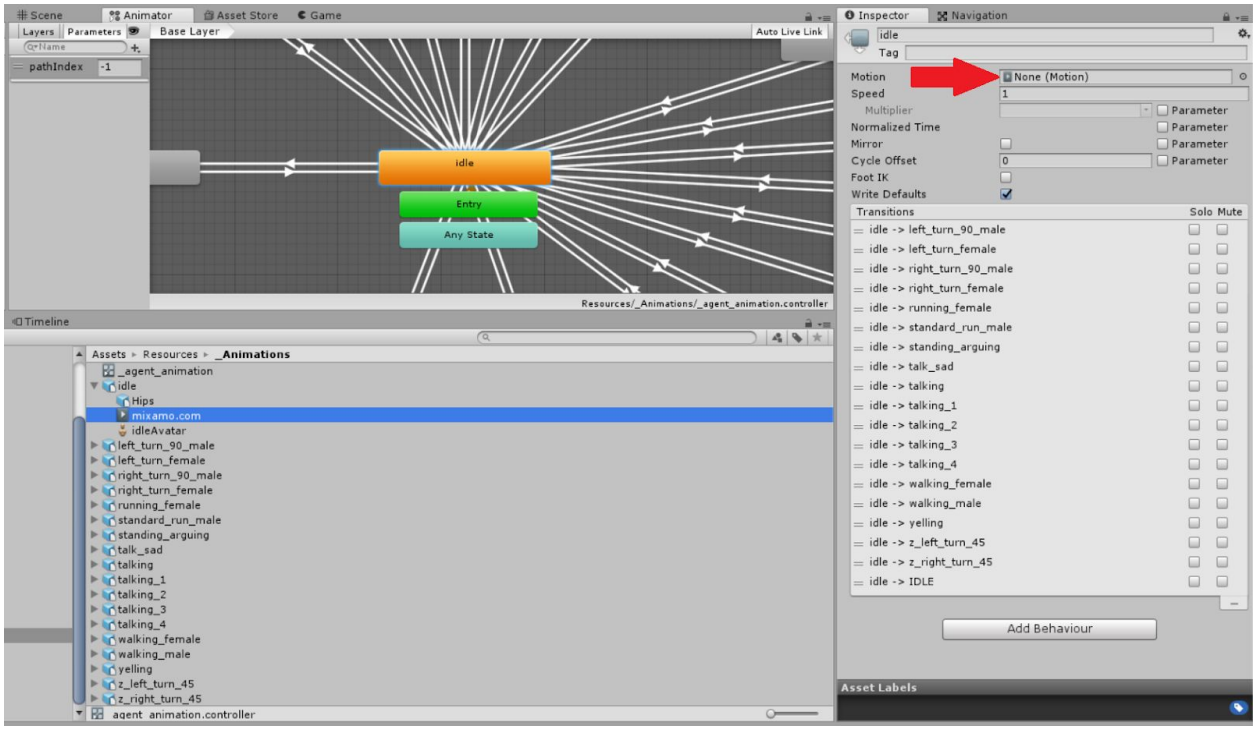


Fix: Open the "Animator" tab. If nothing shows, click on an Agent in the Hierarchy tab, so you can see the animation tree (or graph).

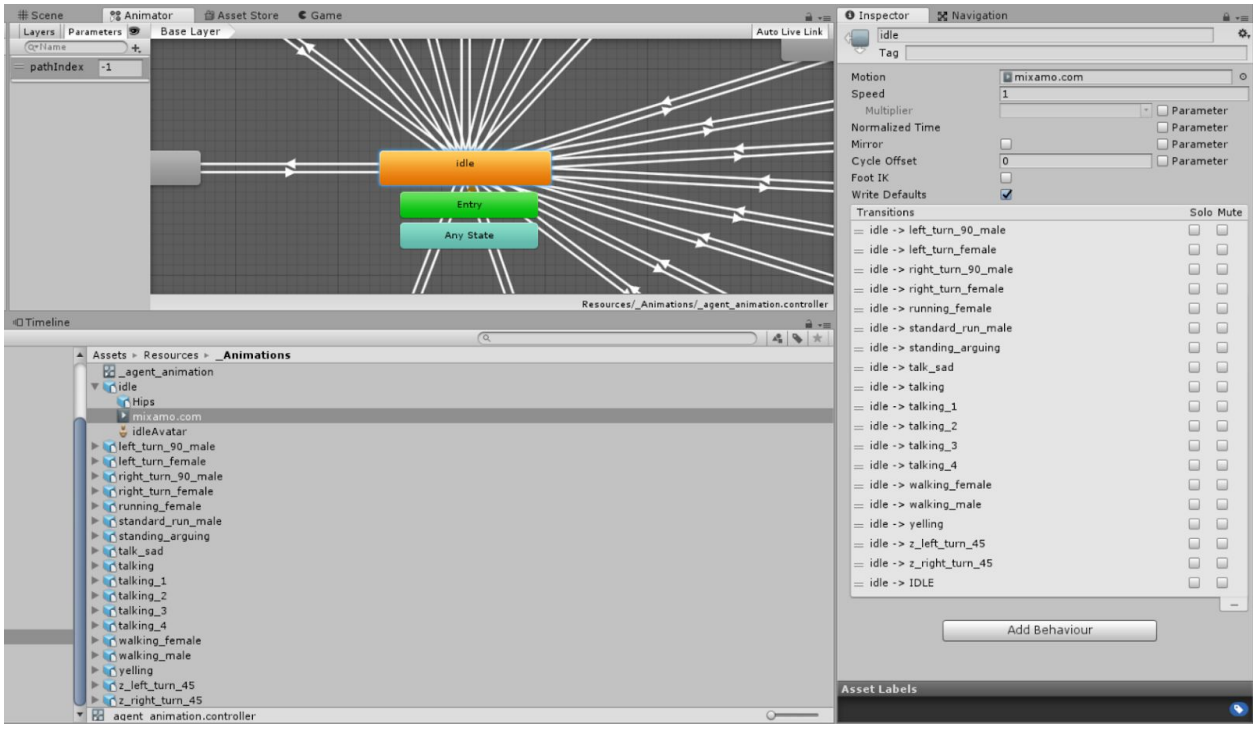


FOR EACH of the boxes, EXCEPT the "Entry" (green box), "Any State" (blue-green box) and "Exit" (red box), you must find (in Assets > Resources > _Animations folder):

a. Drag and drop each animation named in the box (in this example, "idle", you must expand the "idle" animation in the folder) to the Motion field of the Inspector.

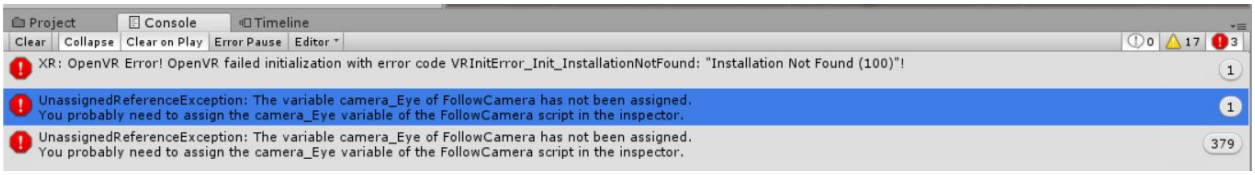


b. The Motion field will be renamed to the animation you dragged, but the animation box in the animation tree will remain unchanged.

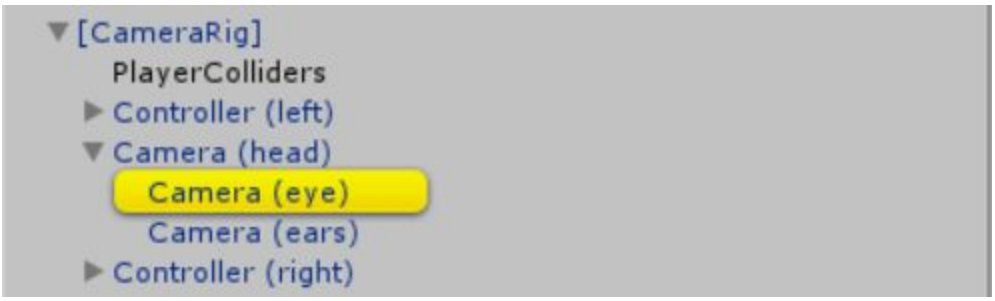


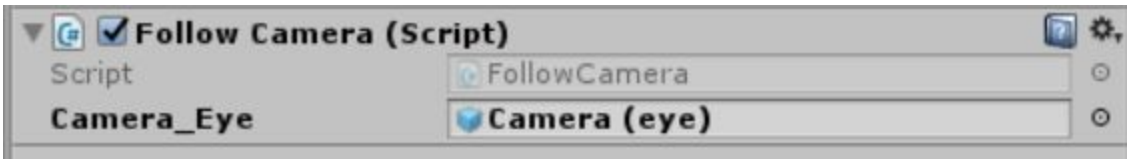
c. You might have to do this for all of the animations in the animation tree (all of the boxes).

4. I get this error when I run the scene:

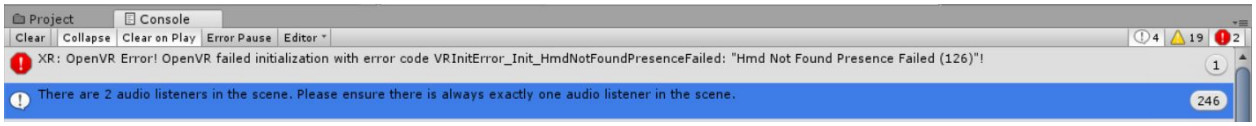


Fix: Expand the [CameraRig] GameObject in the Hierarchy, and drag and drop the Camera (eye) GameObject from the Hierarchy to the Camera_Eye field of the LookAt GameObject:

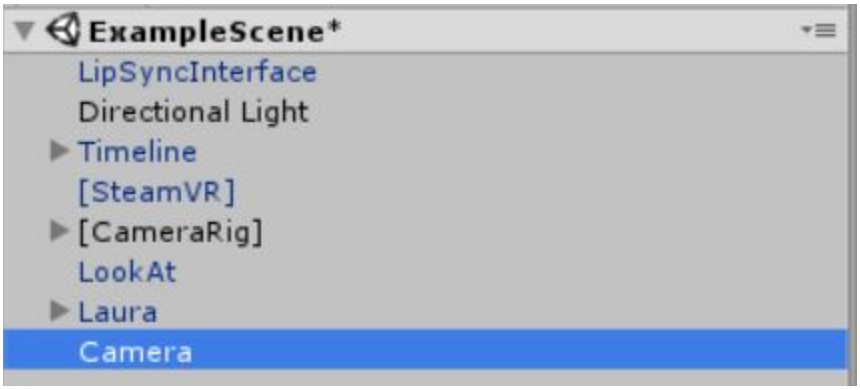




5. I get the following error:



Fix: Delete the “Camera” or “MainCamera” GameObject from the Hierarchy:



Make sure that you DO NOT delete the [CameraRig] GameObject.

6. My dialogs don't play.

Fix: Click on the Agent's Head in the Hierarchy. In the OVR Lip Sync Context (Script) component, make sure the Audio Loopback box is checked: