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?

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 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. Wildcards: handle general verbal inputs (no need for recognition)
2. Responses: handle specific responses (recognition by keyword phrases)
3. 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.

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

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!

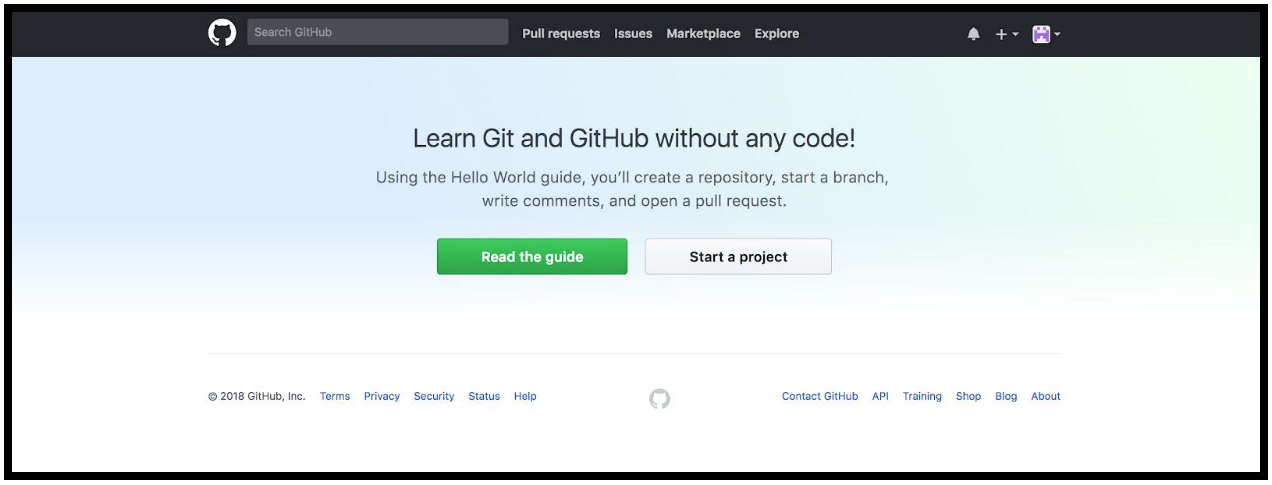
2. Required: Install Unity (2017.1 or later)

1. Install Unity 2017.1 or later at this page: https://unity3d.com/get-unity/download
2. Visit the VAIF package from the GitHub repository (available at https://github.com/isguser/VAIF).
3. Download the VAIF package as a ZIP folder. (*see instructions on page 5-6)*

Tutorial Video: VAIF 1

3. Downloading the GitHub Repository

1) Login to Github and you should see a screen as follows:

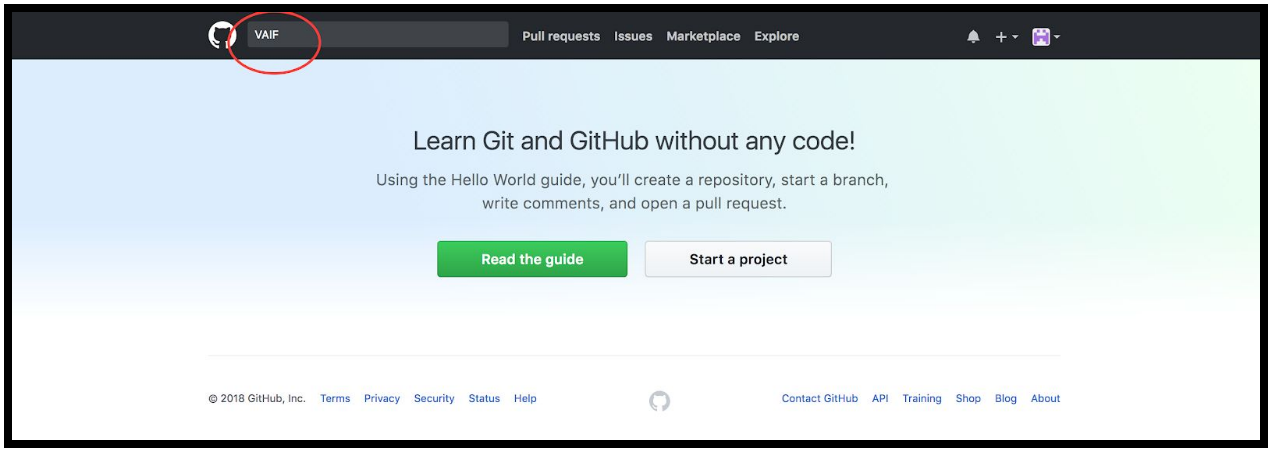


Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

© 2018 GitHub, Inc.

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

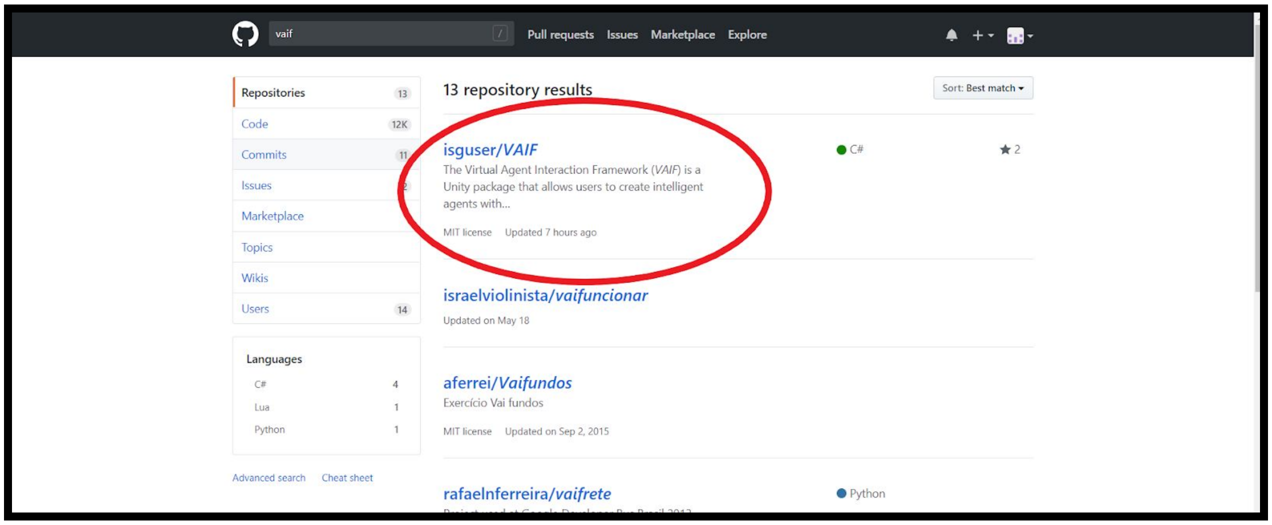


Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

© 2018 GitHub, Inc.

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



13 repository results

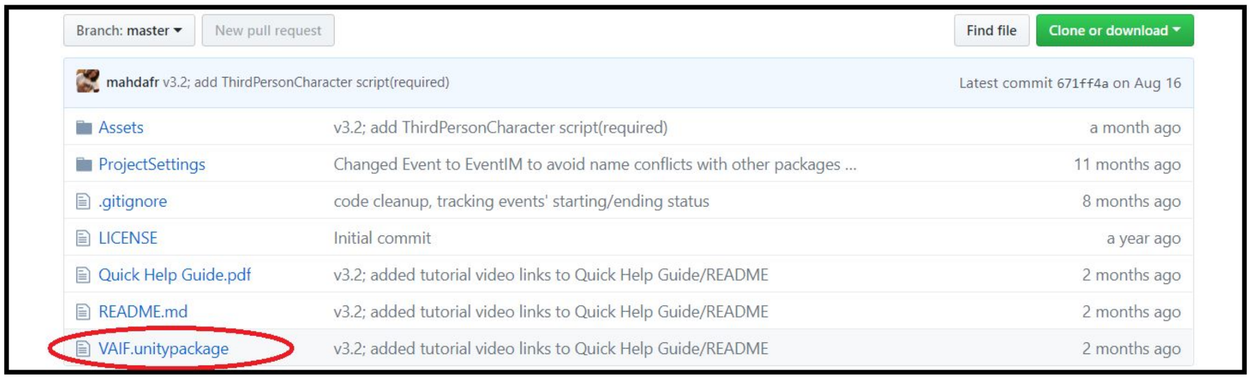
isguser/VAIF
The Virtual Agent Interaction Framework (VAIF) is a Unity package that allows users to create intelligent agents with...
MIT license Updated 7 hours ago
● C# ★ 2

israelviolinista/vaifuncionar
Updated on May 18

aferrei/Vaifundos
Exercício Vai fundos
MIT license Updated on Sep 2, 2015

rafaelnferreira/vaifrete
● Python

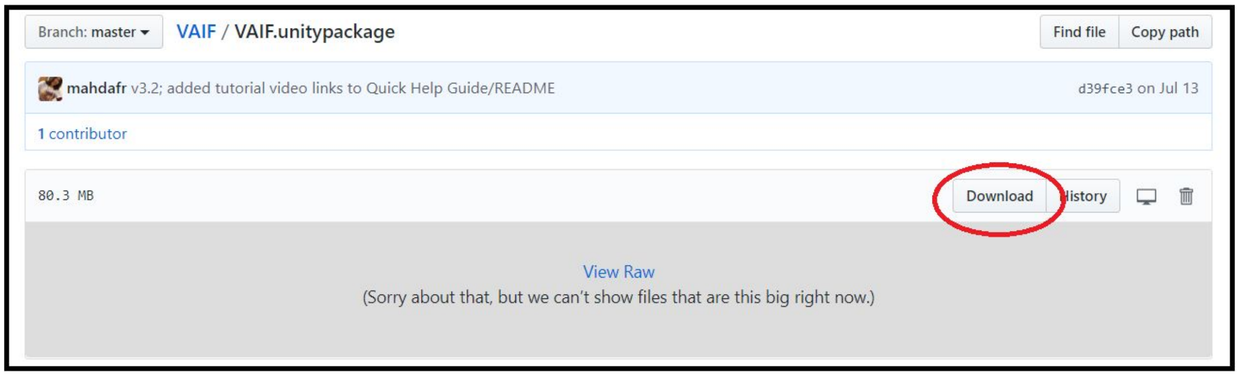
4) Click the **VAIF.unitypackage** file in the repository:



mahdafr v3.2; add ThirdPersonCharacter script(required) Latest commit 671ff4a on Aug 16

Assets	v3.2; add ThirdPersonCharacter script(required)	a month ago
ProjectSettings	Changed Event to EventIM to avoid name conflicts with other packages ...	11 months ago
.gitignore	code cleanup, tracking events' starting/ending status	8 months ago
LICENSE	Initial commit	a year ago
Quick Help Guide.pdf	v3.2; added tutorial video links to Quick Help Guide/README	2 months ago
README.md	v3.2; added tutorial video links to Quick Help Guide/README	2 months ago
VAIF.unitypackage	v3.2; added tutorial video links to Quick Help Guide/README	2 months ago

5) Click the **Download** option and the wait for the file to finish downloading to your computer:



mahdafr v3.2; added tutorial video links to Quick Help Guide/README d39fce3 on Jul 13

1 contributor

80.3 MB

(Sorry about that, but we can't show files that are this big right now.)

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, and select the VAIF.unitypackage file.
4. Click "Open" and wait for Unity to unpack all the materials.

Tutorial Video: *VAIF 2*

5. The Example Scene

Included in the downloadable package on Unity is an example scene in Unity called VAIF. This package contains an example of each of the 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. Extract the zip folder.
2. Open Unity and click Open. Navigate to the directory where you extracted the project folder.
3. Look at the example of two agents and an example of each of type of Event.

6. Create a Timeline

A timeline consists of one or more conversations and represent the sequence of events in your scene and when they will be played.

1. From the top menu add an empty GameObject. Click on the GameObject dropdown menu and select Create Empty. *(or use keyboard shortcut Ctrl+Shift+N)*
2. Rename this GameObject to Timeline.
3. Create a new Tag in the Timeline GameObject and call it Timeline. Attach this Tag to the Timeline GameObject.
4. Double click on the Timeline GameObject to open the Inspector Window. Click 'Add Component' to add the following components (in the following order) to the Timeline:
    a. Conversation IM - Script
    b. Interaction Manager - Script
    c. Emotion Check Manager - Script
    d. Response Manager - Script
    e. Trigger Manager - Script
    f. Wait Manager - Script
    g. Wildcard Manager - Script
    h. Memory Check Manager - Script
5. Now that you've created a Timeline, you can add Conversations to it.

Tutorial Video: *VAIF 3*

6.1. Create a Conversation

1. You need to create a Conversation in order to add Events to play.
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. Click on the Conversation GameObject. Create a new Tag in the Conversation GameObject and call it the same name as you chose in Step 2 (in our example, ConversationA). Attach this Tag to the Conversation GameObject you created (see the hierarchy on the left).
5. Now that you've created a Conversation, you can add Events to it.

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 General Events to a Conversation

1. You need to create a Conversation in order to run Events.
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. Select the Agent that the Event will be attached to. For example, an Animation event attached to your Agent of choice.

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.
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 Agent

7.1. Adding the Managers

Your Agent will be composed of several managers. 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.

7.1.1. 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

7.1.2. Move Manager

Dictates to the agent where to move next and calculates which animation to play with. The required scripts and components are:

- *Third Person Character* - Script
- *Nav Mesh Agent* - Script
- *_agent_animation_WLocoMotion* - Animator
    - Located in Resources > _Animations

7.1.3. Memory Manager (Future Implementation)

Develops a memory database for each user, keeping track of the agent and events in an interaction.

7.1.4. Emote Manager (Future Implementation)

With facial blendshapes, your agent will be able to display specific facial expressions as controlled by the emote event.

7.1.5. Dialog Manager

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

7.1.6. 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.

7.1.7. Look At Collisions

A box collider that is attached to the Player to keep track if the Player is looking at the Agent.

7.1.8. Nav Mesh Agent

This allows your Agent to move based on any given target.

7.1.9. Personalities (Future Implementation)

Eventually, Agents will be able to represent themselves based on the type of personality you instruct.

7.1.10. Emotions (Future Implementation)

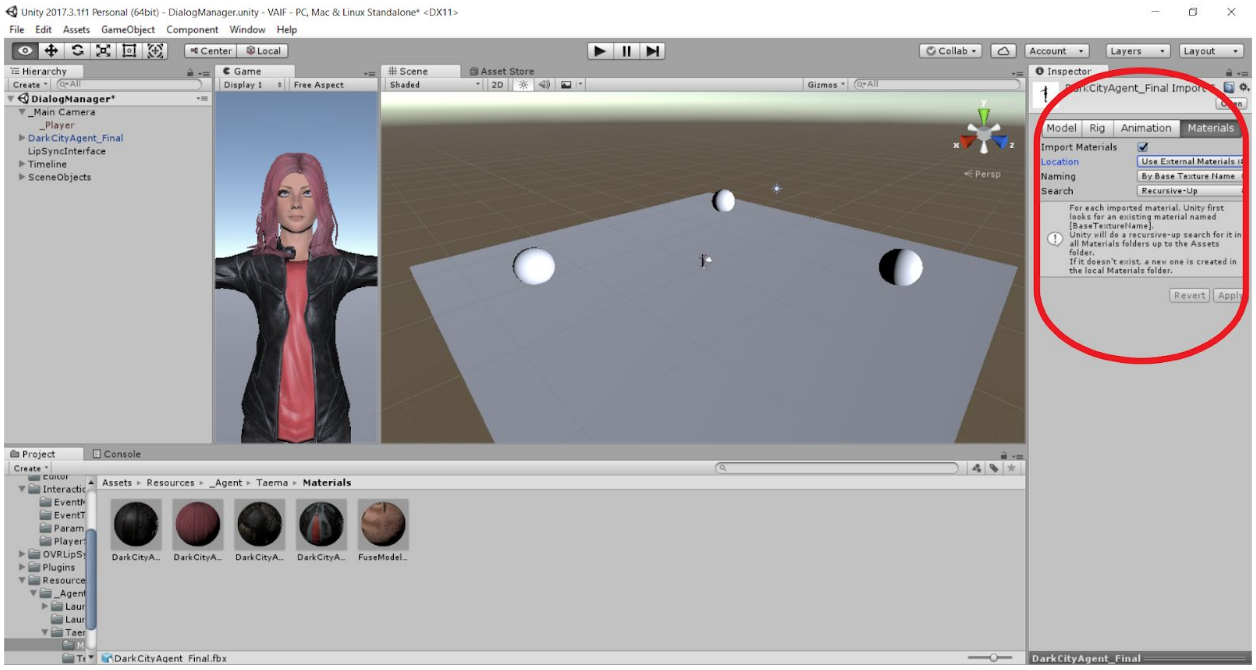
This manager keeps track of display of emotions for the agent.

7.1.11. Gaze (Future Implementation)

Make the agent look at camera (participant inside unity scene) and other specific objects inside the unity scene.

8. Troubleshooting

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



Unity 2017.3.1f1 Personal (64bit) - DialogManager.unity - VAIF - PC, Mac & Linux Standalone* <DX11>

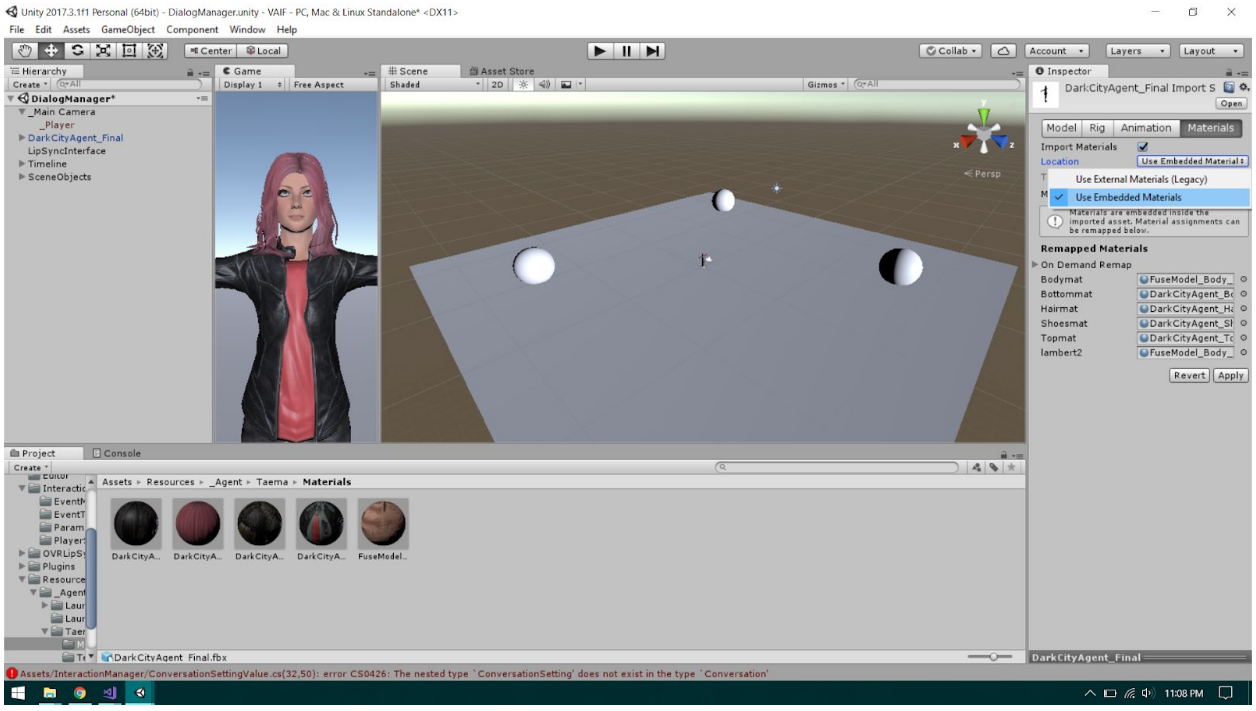
DarkCityAgent_Final Import Settings

Import Materials ✓
Location: Use External Materials (Legacy)
Naming: By Base Texture Name
Search: Recursive-Up

For each imported material, Unity first looks for an existing material named [BaseTextureName].
Unity will do a recursive-up search for it in all Materials folders up to the Assets folder.
If it doesn't exist, a new one is created in the local Materials folder.

Assets ▸ Resources ▸ _Agent ▸ Taema ▸ Materials

DarkCityAgent_Final.fbx



Unity 2017.3.1f1 Personal (64bit) - DialogManager.unity - VAIF - PC, Mac & Linux Standalone* <DX11>

DarkCityAgent_Final Import Settings

Import Materials ✓
Location: Use Embedded Materials

Use External Materials (Legacy)
✓ Use Embedded Materials

Materials are embedded inside the imported asset. Material assignments can be remapped below.

Remapped Materials

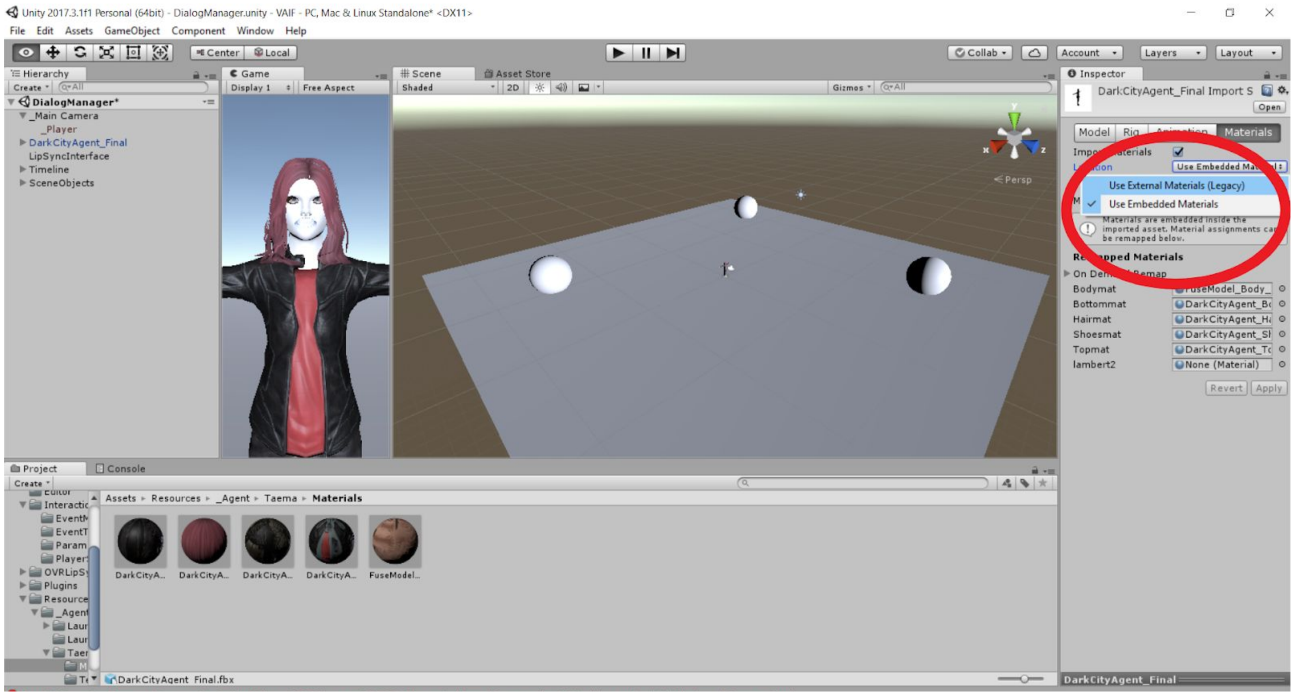
On Demand Remap

Bodymat	FuseModel_Body_
Bottommat	DarkCityAgent_Bo
Hairmat	DarkCityAgent_Ha
Shoesmat	DarkCityAgent_Sh
Topmat	DarkCityAgent_To
lambert2	FuseModel_Body_

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'
```



DialogManager*
- _Main Camera
  - _Player
- DarkCityAgent_Final
- LipSyncInterface
- Timeline
- SceneObjects

Assets ▸ Resources ▸ _Agent ▸ Taema ▸ **Materials**

DarkCityA... DarkCityA... DarkCityA... DarkCityA... FuseModel...

DarkCityAgent_Final.fbx

DarkCityAgent_Final Import S

Import Materials ✓

Location: Use Embedded Materials

- Use External Materials (Legacy)
- ✓ Use Embedded Materials

Materials are embedded inside the imported asset. Material assignments can be remapped below.

Remapped Materials

On Demand Remap

Bodymat	FuseModel_Body_
Bottommat	DarkCityAgent_Bo
Hairmat	DarkCityAgent_Ha
Shoesmat	DarkCityAgent_Sh
Topmat	DarkCityAgent_To
lambert2	None (Material)

DarkCityAgent_Final