Dialogue Utility AI System Manual

A Unity Package by Renae Aurisch

&

Contents

What is the Dialogue Utility AI System Package?

The Dialogue Utility AI System Package is a Unity tool for providing game developers with the scripts and tools to build their own interesting NPCs and/or dialogue. It is a combined dialogue system and an AI utility (actions have assigned values for actions based on the outcomes of those actions – highest value is prioritised) system tool for Unity game developers, saving time from dealing with AI and dialogue programming for small conversations in games. It also allows the developer to provide emotion to NPCs by having their needs affect their emotion, in addition to player dialogue choices. This allows for more life-like characters by allowing them to make decisions based on what they feel rather than evaluating purely for their needs.

For a video explanation of this system and a tutorial on how to build an NPC using this system from scratch, watch this YouTube video: https://youtu.be/bTvlOOsKVb8

Importing Package to Unity

To import the package into Unity, open Unity, go to the Assets -> Import Package -> Custom Package.

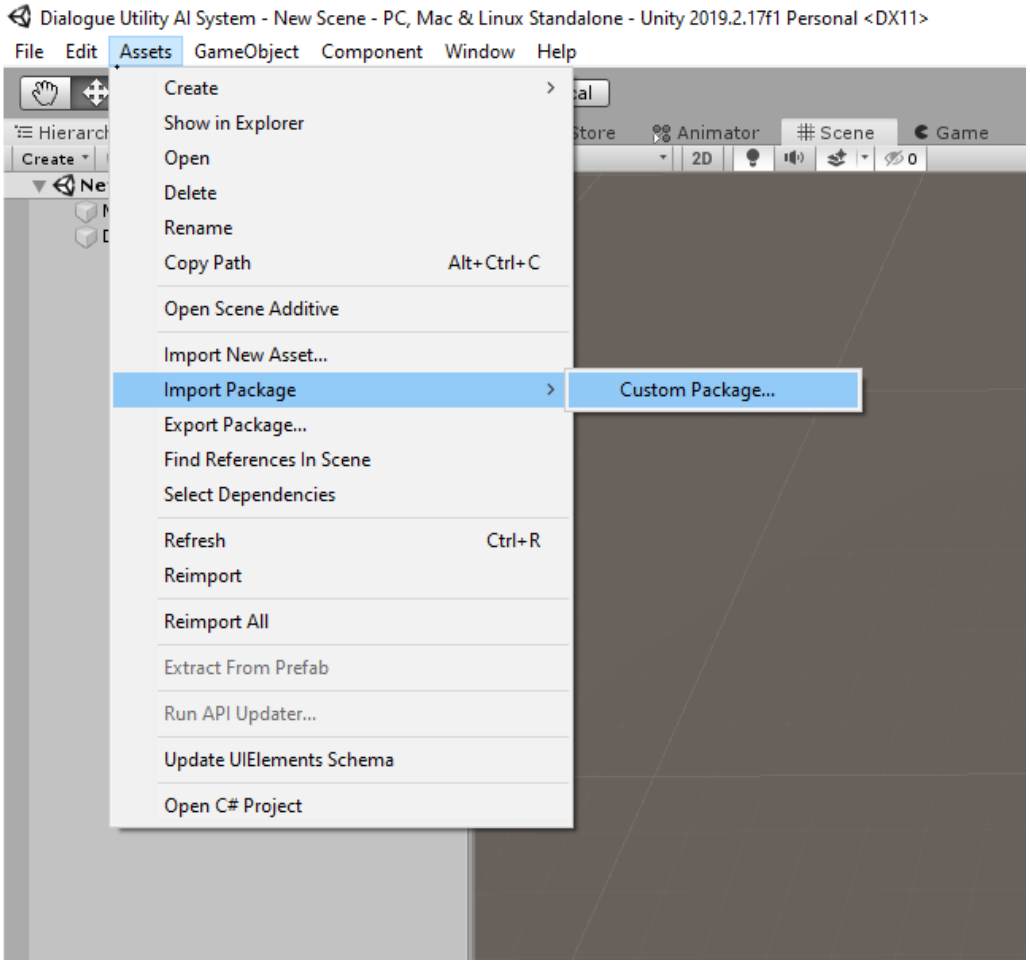

Figure 1. Importing a custom package to Unity

When the Import Package File Explorer window opens, find the DialogueUtilityAISystem.unitypackage file within this folder and select this file to import.

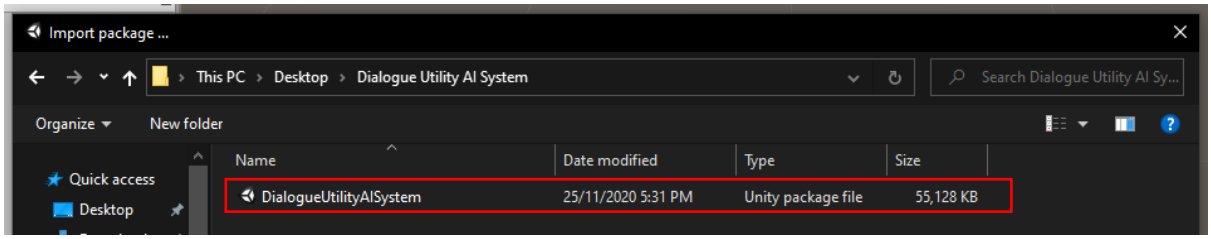

Figure 2. The Dialogue Utility AI System package file to import

AI Agent Setup

AI Canvas Prefab

The AI Canvas prefab is an optional asset for developers to use to view the state of the NPCs within the scene. Although, the current action and current emotion text will only display for the first NPC in the hierarchy, thus, this is best used with one NPC at a time.

Figure 3. The AI Canvas prefab in the prefabs folder

By placing this prefab into the scene, the user can view the current need and emotion and each of the needs and emotion values of the NPC. This can be useful for visually seeing how the NPC behaves based on the need and emotional values and problem solving if the NPC is behaving incorrectly.

Upon placing this prefab in the scene, two labels and side panels will appear. If the NPC does not have any needs or emotions, then this will remain looking like the following image.

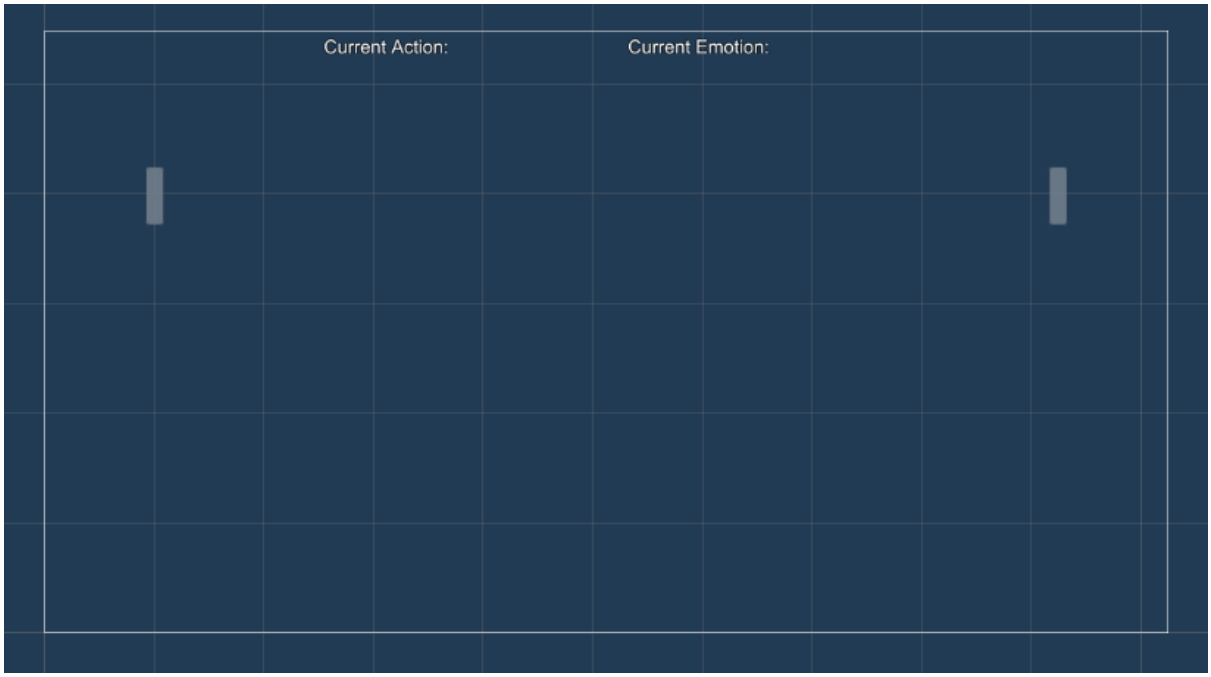

Figure 4. The AI Canvas prefab preview

However, if the NPC does have needs and emotions, they will instantiate bar images to represent the values of the needs and emotions, as seen in the image below.

The AI Canvas prefab can be turned off at any point without impacting the NPCs behavior or dialogue.

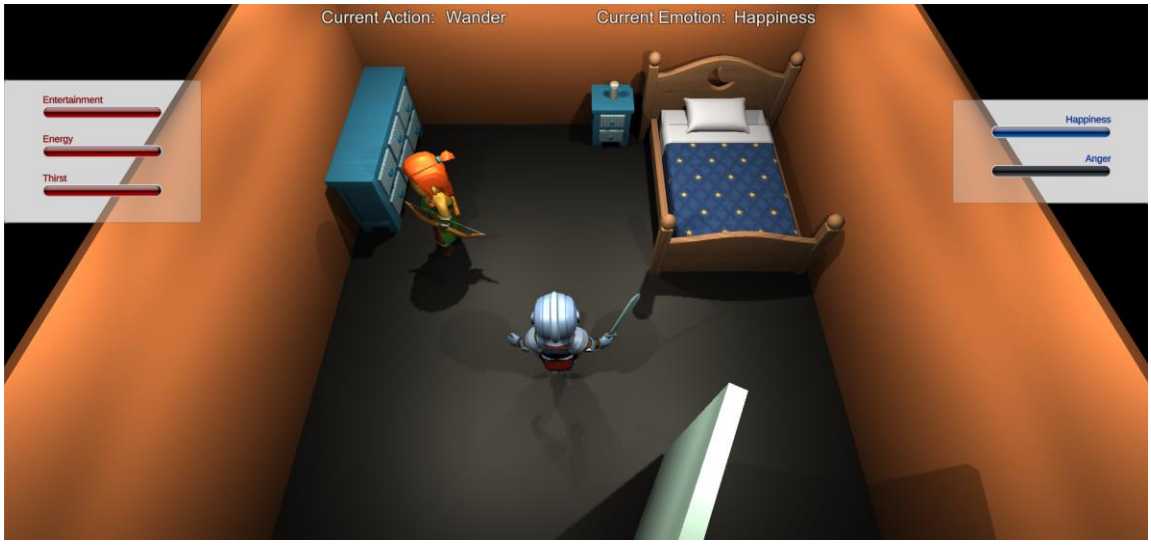

Figure 5. The AI Canvas prefab shown during gameplay (instantiates needs and emotions)

Needs and Emotions

Needs

Needs are ScriptableObjects which are used to represent the needs of NPCs. They can be created within the project folder by right clicking on the project folder going to Create -> DUAS_Need.

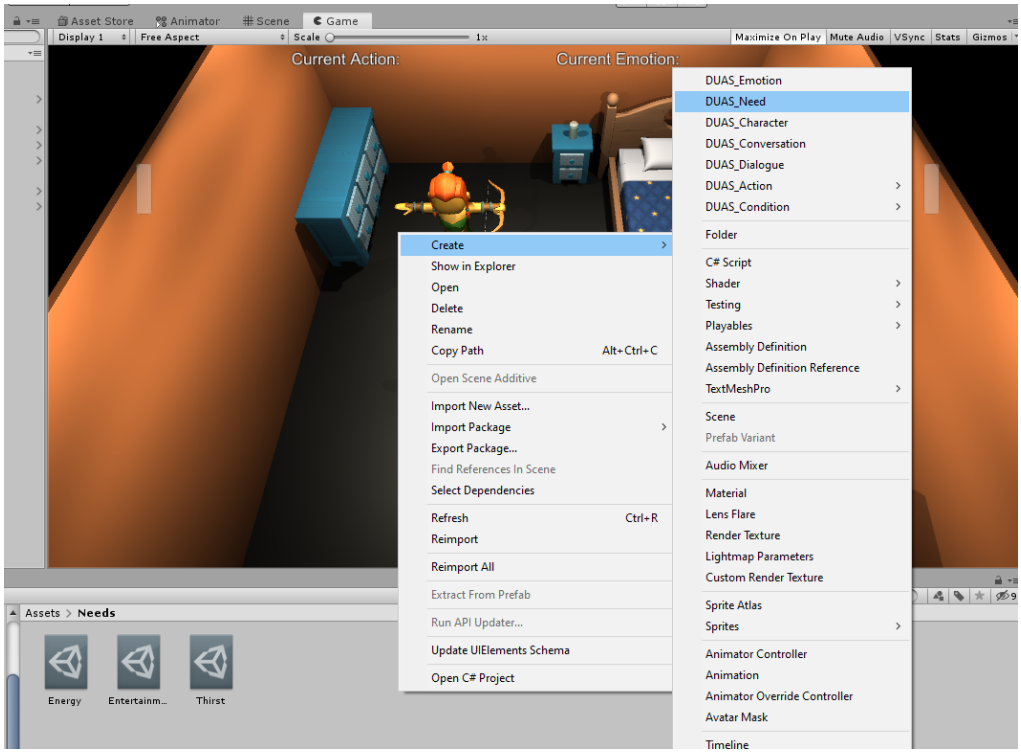

Figure 6. Creating a need from the Project folder

Once a Need has been created, it will appear in the project folder.

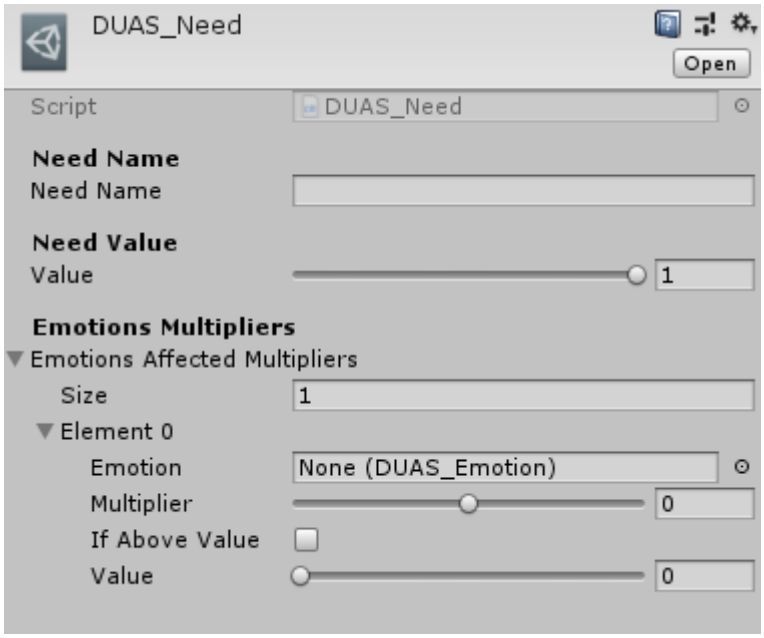

Figure 7. Need inspector view

The **Need Name** is the string name that represents the need (e.g. Hunger, Entertainment, Energy, etc.).

The **Need Value** the value of the need (between 0 and 1). Note: this value will be represented as the AI Canvas need bar fill amount if the AI Canvas is being used. This will automatically be set to 1 on starting the program.

Emotion Multipliers are an optional list of effects that can occur based on this need's value. Set **Size** to be the amount of emotions that will be affected by this need. For each **Element** there are fields to put in the **Emotion** object, a **Multiplier** that will apply to the **Emotion**'s value per second, a Boolean, **If Above Value**, which represents the whether the need is the **Value** or above (if set to true) or is the **Value** and below (if set to false).

Needs are used on Game Objects with the DUAS_Agent script on it, which will be further discussed in Agent section of this manual.

Emotions

Emotions, much like Needs, are Scriptable Objects which are used to represent the emotions of NPCs. They can be created within the project folder by right clicking on the project folder going to Create -> DUAS_Emotion.

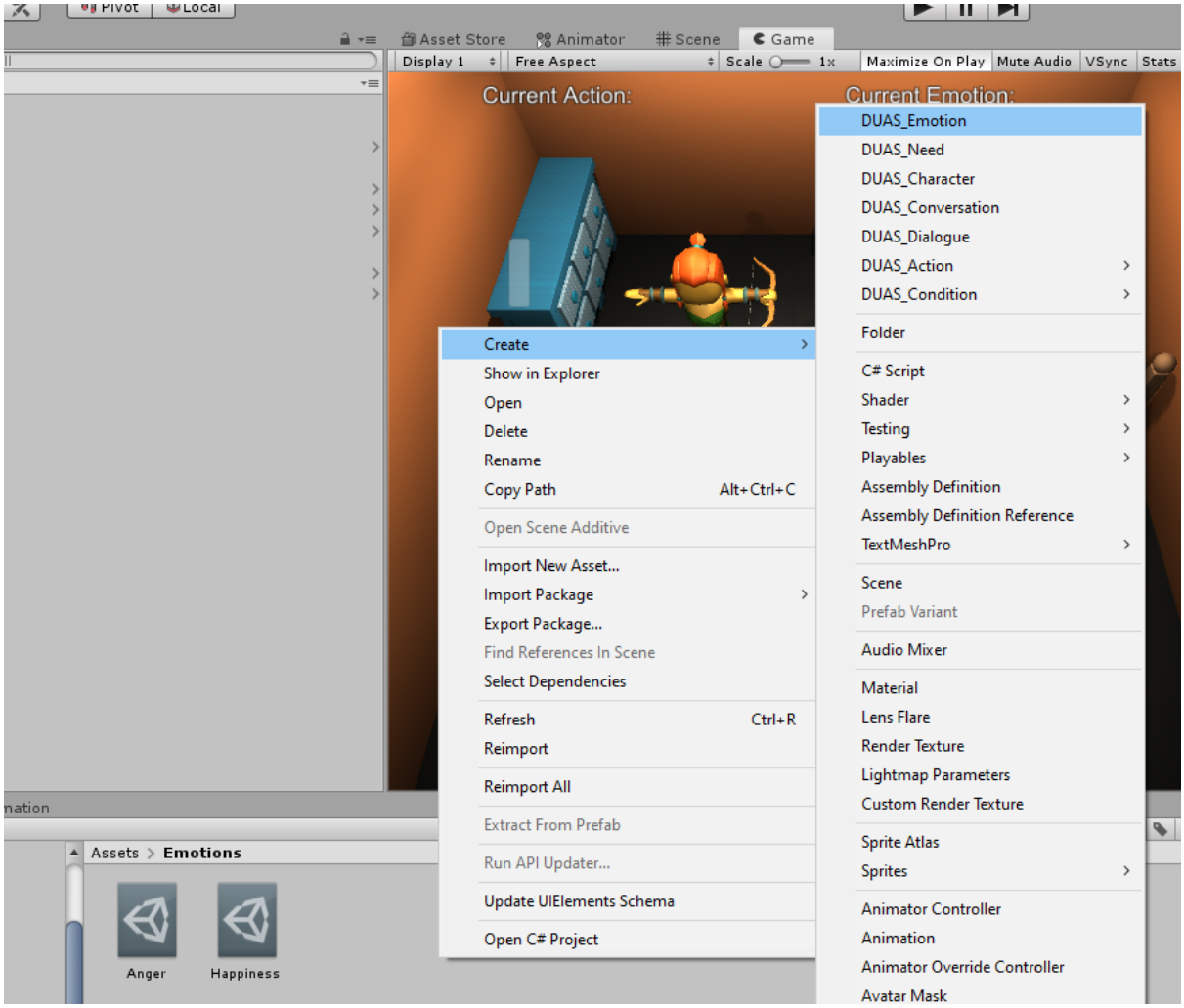

Figure 8. Creating an emotion from the project folder

Once an Emotion has been created, it will appear in the project folder.

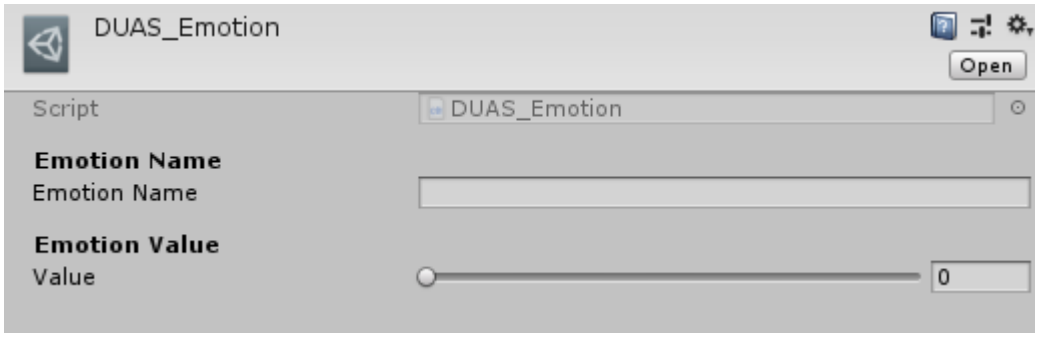

Figure 9. Emotion inspector view

The **Emotion Name** is the string name that represents the emotion (e.g. Happiness, Anger, Sadness, etc.).

The **Emotion Value** the value of the need (between 0 and 1). Note: this value will be represented as the AI Canvas emotion bar fill amount if the AI Canvas is being used. This will automatically be set to 1 on starting the program.

Emotions are used on Game Objects with the DUAS_Agent script on it, which will be further discussed in Agent section of this manual.

Actions

The DUAS_Action class is an abstract Scriptable Object class that has different types of action scripts that derive from it.

Wander Action

The Wander Action is a Scriptable Object that represents the intrinsic action of wandering around and standing idle. It can be created by right clicking the project folder and selecting Create -> DUAS_Action -> DUAS_WanderAction.

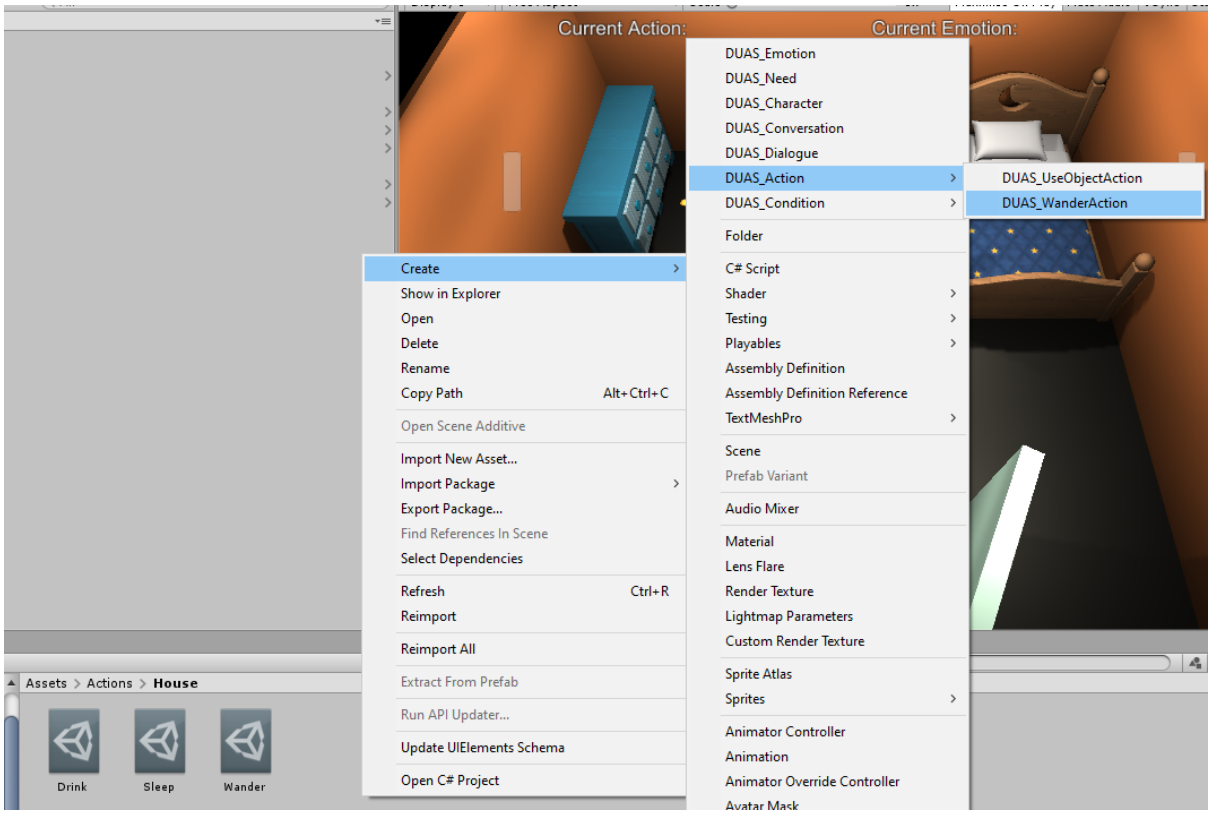

Figure 9. Creating a Wander Action from the Project folder

Any created Wander Action will allow for the system to recognise that the NPC is not performing an important action and can be talked to by the player.

This action can be used as the only action of an NPC to have them walk around and be idle similar to older style RPGs.

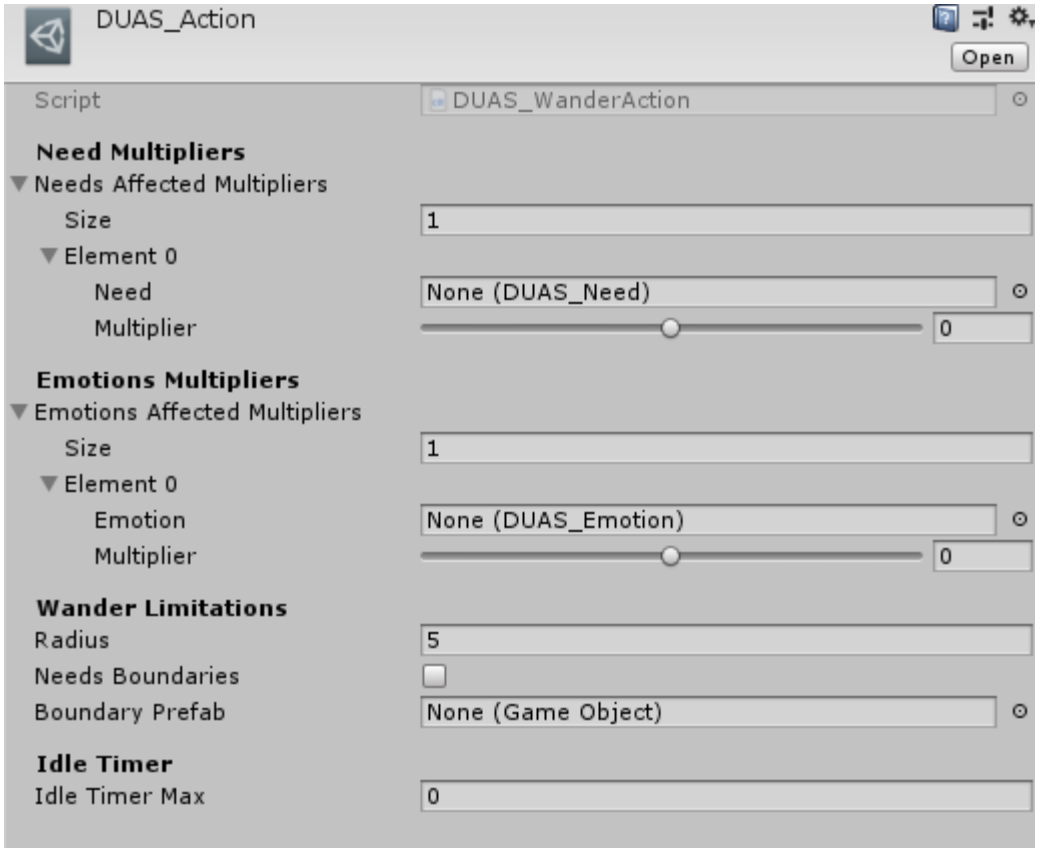

Figure 10. Wander Action inspector view

Deriving from DUAS_Action, **Need Affected Multipliers** are an optional list of effects that can occur if this action is currently being performed. Set **Size** to be the amount of needs that will be affected by this action. For each **Element** there are fields to put in the **Need** object, a **Multiplier** that will apply to the **Need**'s value per second.

Also deriving from DUAS_Action, **Emotion Effected Multipliers** are an optional list of effects that can occur if this action is currently being performed. Set **Size** to be the amount of emotions that will be affected by this action. For each **Element** there are fields to put in the **Emotion** object, a **Multiplier** that will apply to the **Emotion**'s value per second.

The **Radius** is the radius around the agent that the NPC can find their next target position for walking around. The **Needs Boundaries** field is Boolean determining if the NPC needs to be contained within the **Boundary Prefab**'s collider bounds, and the **Boundary Prefab** is an empty Game Object with default Transform values and a Box Collider that has a position and size that represents the space the NPC will contain its Wander Action within, see the picture below for an example of a **Boundary Prefab**. This prefab must be passed in from the project scene and not the hierarchy. **Idle Timer Max** is the amount of time that the NPC stays idle for before finding a new location to walk to.

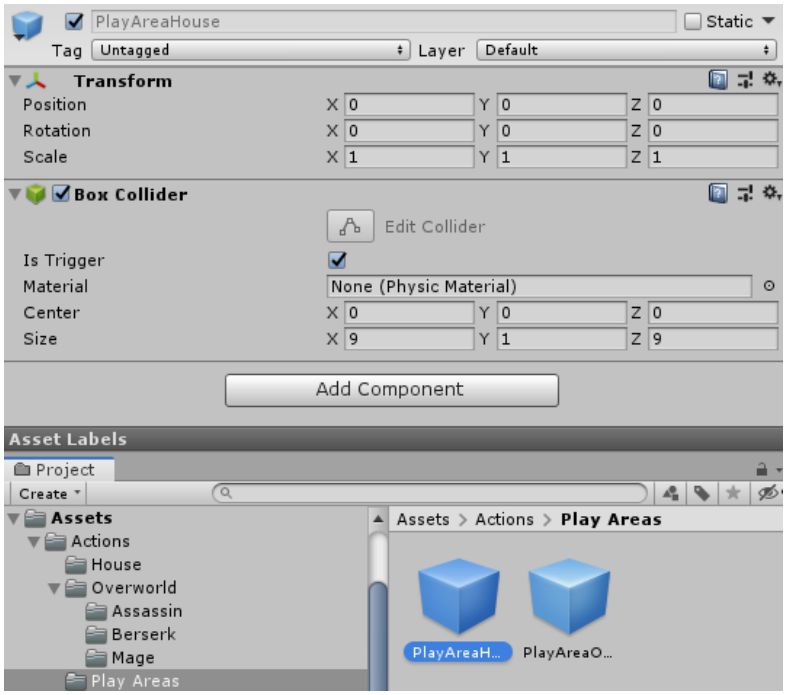

Figure 11. An example of a Boundary prefab from the Wander Action

Use Object Action

The Use Object Action is a Scriptable Object that represents the action of walking to an object, and either picking it up to use the object or use it in another way (with animations). It can be created by right clicking the project folder and selecting Create -> DUAS_Action -> DUAS_UseObjectAction.

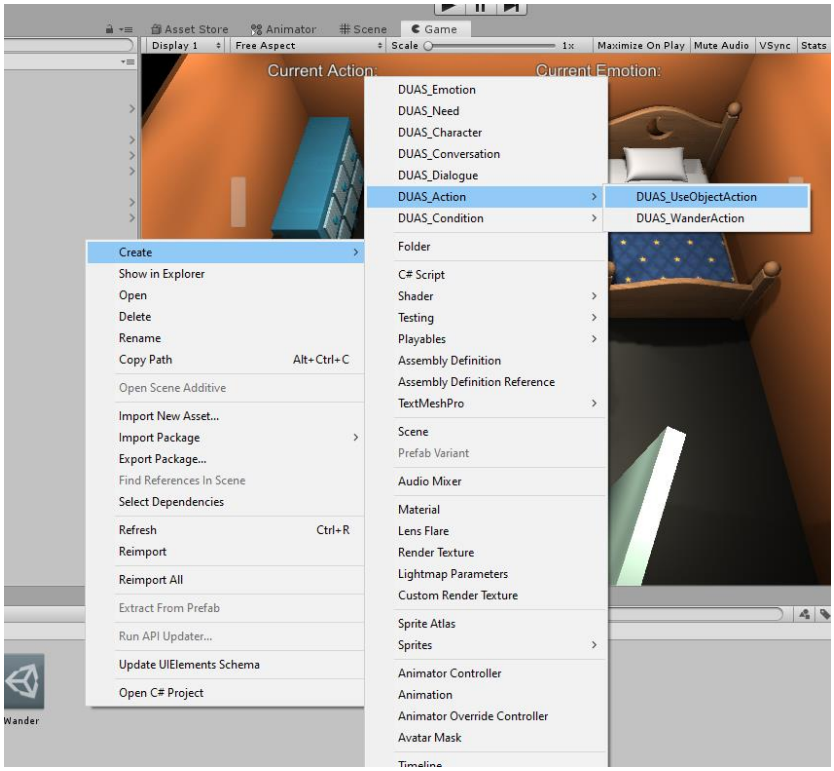

Figure 12. Creating a use object action in the project folder

Name the UseObjectAction object to an appropriate name to suit what the action is (e.g. Drink, Sleep, Write, etc.) These types of actions require a Useable in the scene that this action can be performed on, this will be discussed in the Useable Objects section below.

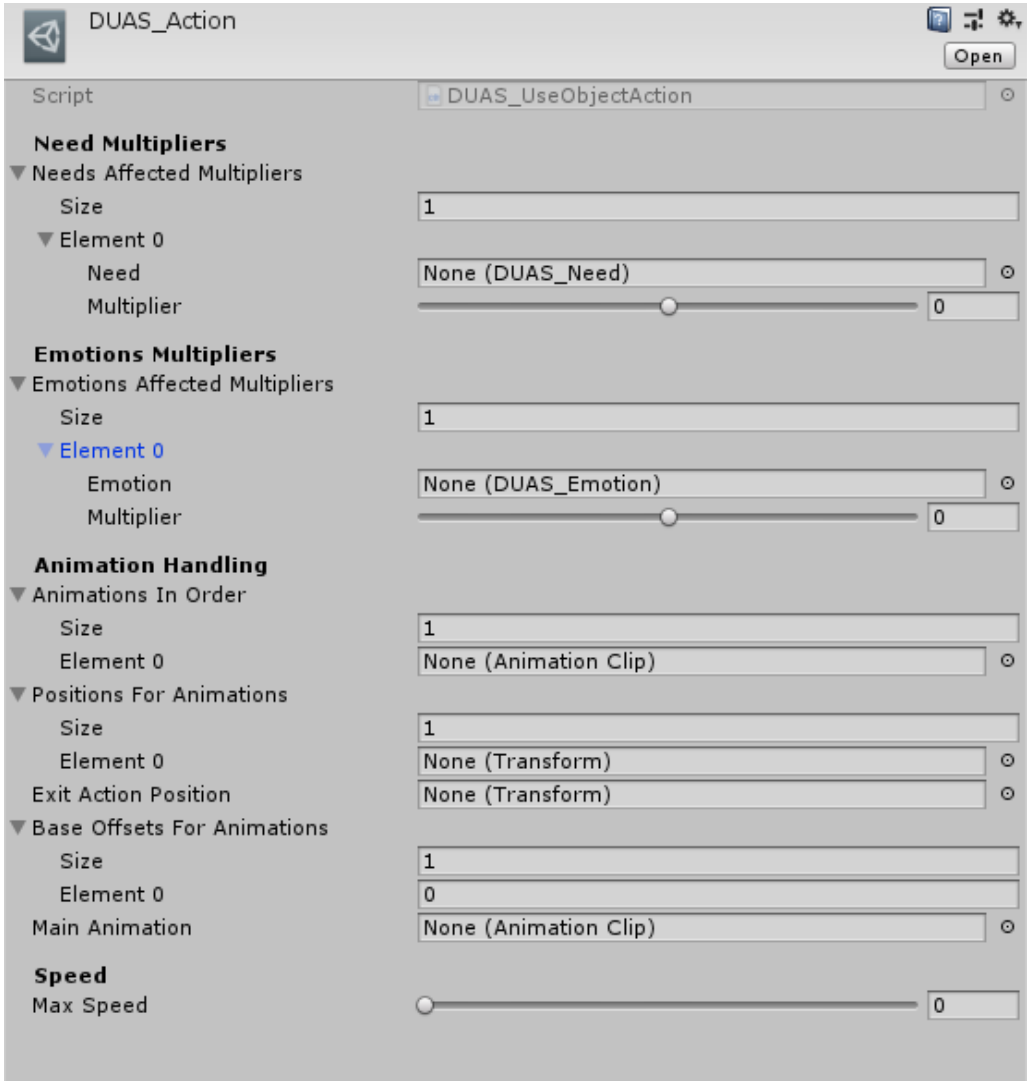

Figure 13. Use Object Action inspector view

Deriving from DUAS_Action, **Need Affected Multipliers** are an optional list of effects that can occur if this action is currently being performed. Set **Size** to be the amount of needs that will be affected by this action. For each **Element** there are fields to put in the **Need** object, a **Multiplier** that will apply to the **Need**'s value per second.

Also deriving from DUAS_Action, **Emotion Affected Multipliers** are an optional list of effects that can occur if this action is currently being performed. Set **Size** to be the amount of emotions that will be affected by this action. For each **Element** there are fields to put in the **Emotion** object, a **Multiplier** that will apply to the **Emotion**'s value per second.

When an NPC uses an object, they may need to perform a series of animations before they can get to performing the action itself (e.g. the NPC needs to sit on the bed first, then lay down, then finally they can perform the sleep action). This is what the **Animations In Order** field is for. Change the **Size** to suit the amount of animations that the NPC will do for this action, then pass in each **Element** the animations in order of how they should play (this must

include the main animation for the action, e.g. the sleep animation is the main animation of the sleep action).

Having the NPC perform a list of animations before performing the main animation may cause the NPC to be positioned incorrectly in relation to object. If this is true, create empty Game Objects and placing them is the scene positioned and rotated to suit how the NPC's Transform should face, then once at the correct position and rotation, drag it into the project folder to make it a prefab (you can delete the prefab from the scene now). The pass the position prefabs into the **Positions For Animations** field by setting the **Size** to be the amount of positions (must be equal to the **Size** of **Animations In Order** with each position element directly relating to each animation).

The **Exit Position** is the position and rotation of where the NPC will spawn once they have stopped performing this action.

In addition, as this system is supports the use of Unity's Nav Mesh and the **Positions For Animations** cannot apply the Y position due to the Nav Mesh Agent's Base Offset, the **Base Offsets For Animations** is a list of float values that will change the NPCs Base Offset during each animation. Set **Size** to the same number as both **Animations In Order** and **Positions For Animations** and assign where the NPC's Y position should be during each animation.

Main Animation is the animation that represents the action (e.g. sleep is the main animation of the sleep action, despite also having the animations sit and lay down). This should be the same animation as the final animation in the **Animations In Order** list.

Lastly, **Max Speed** is a float value that represents the maximum speed that the NPC can walk at, while they are walking to the object.

Useable Objects

Useable Objects are Game Objects (such as beds, drinks, torches, etc.). that can have an action performed on them. There must be a type of Collider on the Game Object before you can add the DUAS_Useable script to it.

Any Useable Object must use the "Useable" tag.

If the Game Object is large, a Nav Mesh Obstacle component may be needed as the NPC may walk through it.

An example of a Useable Object set up is shown below.

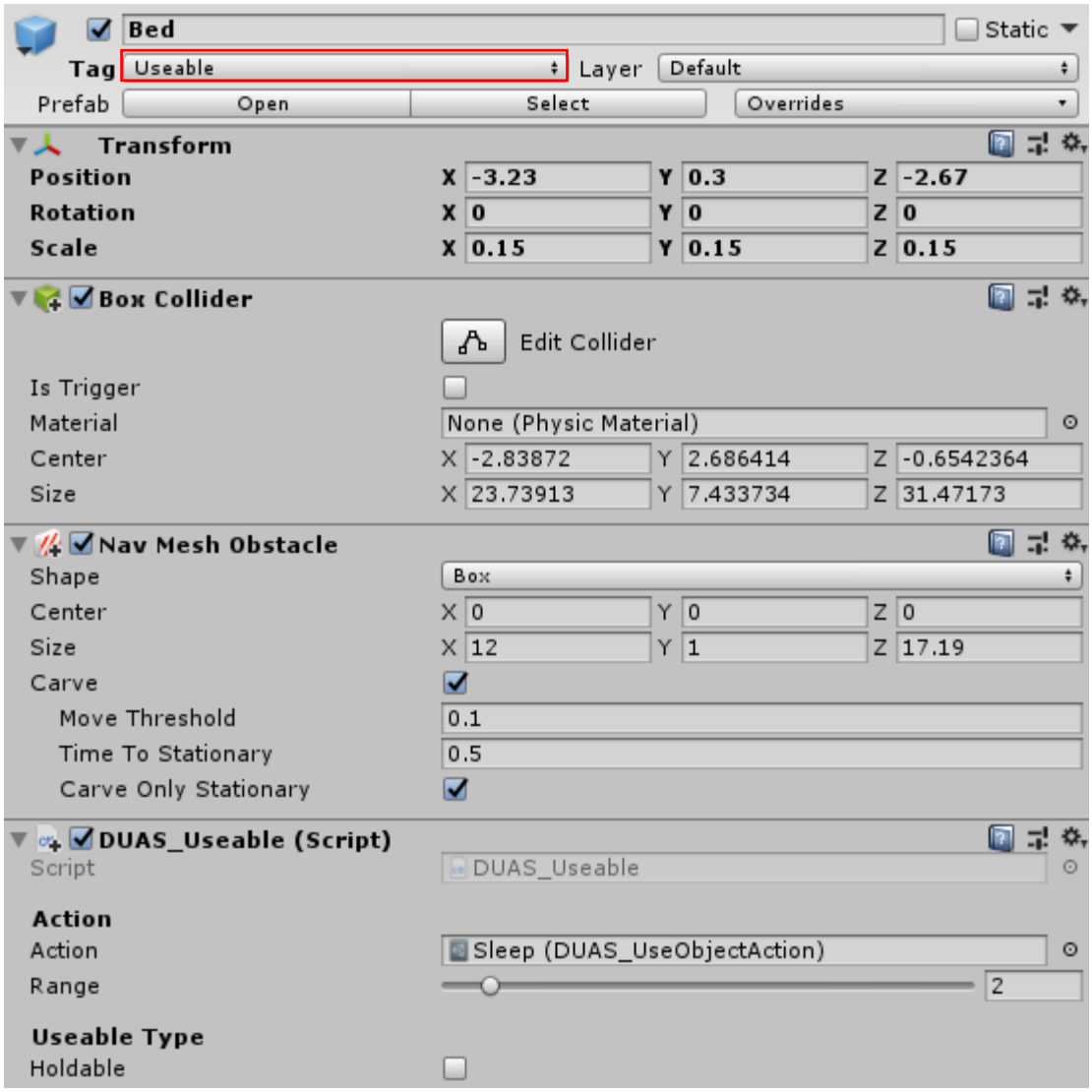

Figure 14. Example bed Useable object

The **Action** is the Use Object Action that can be performed on this Useable. **Range** is the distance that the NPC needs to be within the Game Object before they can begin performing the action. The **Holdable** field, if set to true, allows for Useables to be picked up by the NPC (this should be used for any handheld objects), if set to false, the NPC will perform the animations only.

Conditions

The DUAS_Condition class is an abstract Scriptable Object class that has different types of condition scripts that derive from it.

Time of Day Condition

Time of Day Conditions are conditions that rely on the time of day (day, night or both). They can be created by right clicking the project folder and selecting Create -> DUAS_Condition -> DUAS_TimeOfDayCondition.

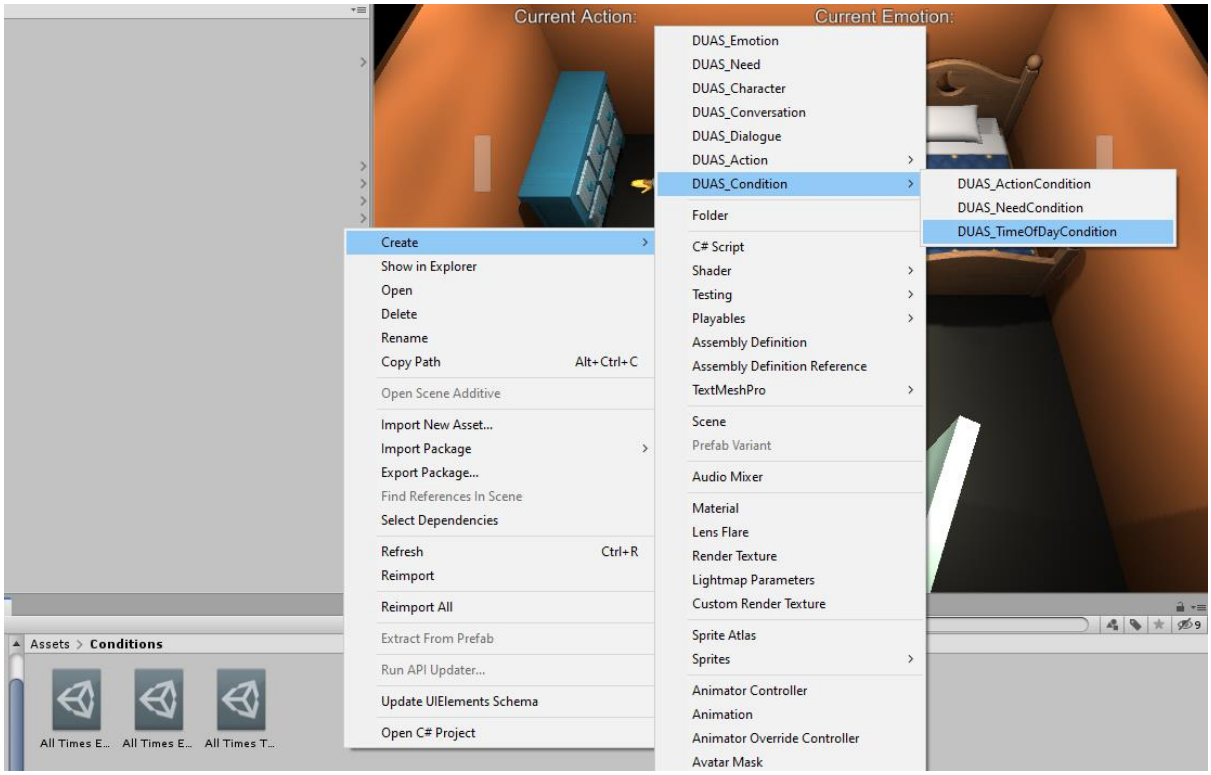

Figure 15. Creating a Time of Day Condition from the Project folder

This type of Condition requires the Sun prefab in the scene (or another equivalent Game Object with the “Sun” tag and Sun Rotation script). The Light component on the Sun prefab is optional (when turned on, it will replicate sunlight in the scene). The Sun prefab needs to be set up with the DUAS_Agent script, which will be discussed in the Agents section.

Below is a picture of the sun prefab in the folder and another picture showing the Inspector view of the Sun prefab.

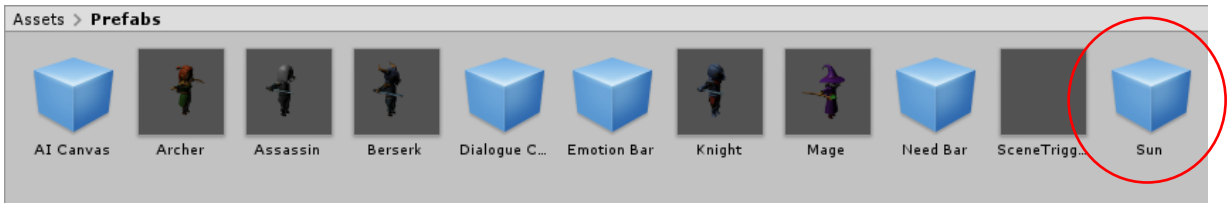

Figure 16. The Sun prefab in the Prefabs folder

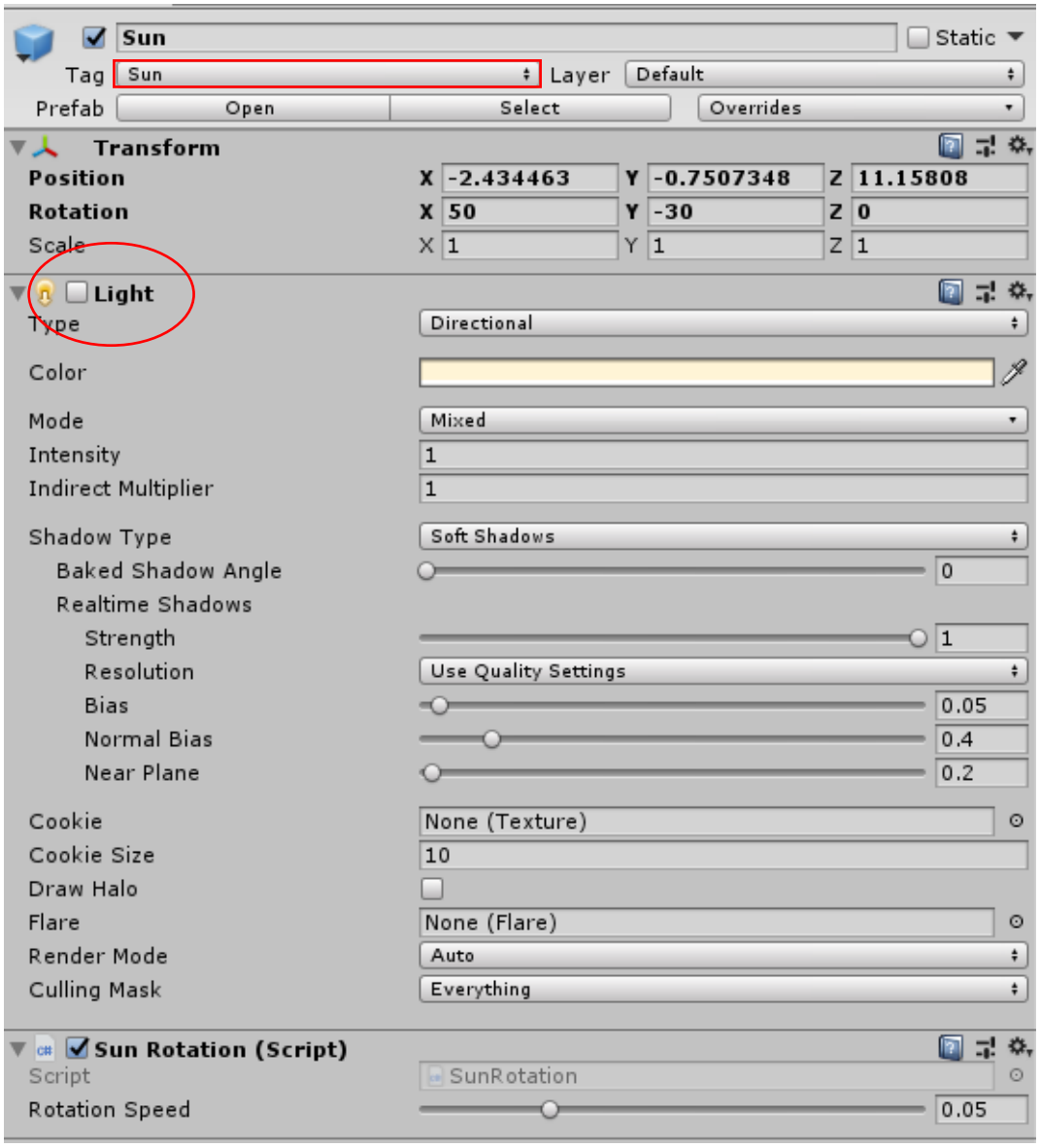

Figure 17. The Sun prefab's inspector view

A Time of Day Condition should be named appropriately (e.g. NightWarmthDecrement is a clear name that explains that during nighttime, the warmth need will decrease).

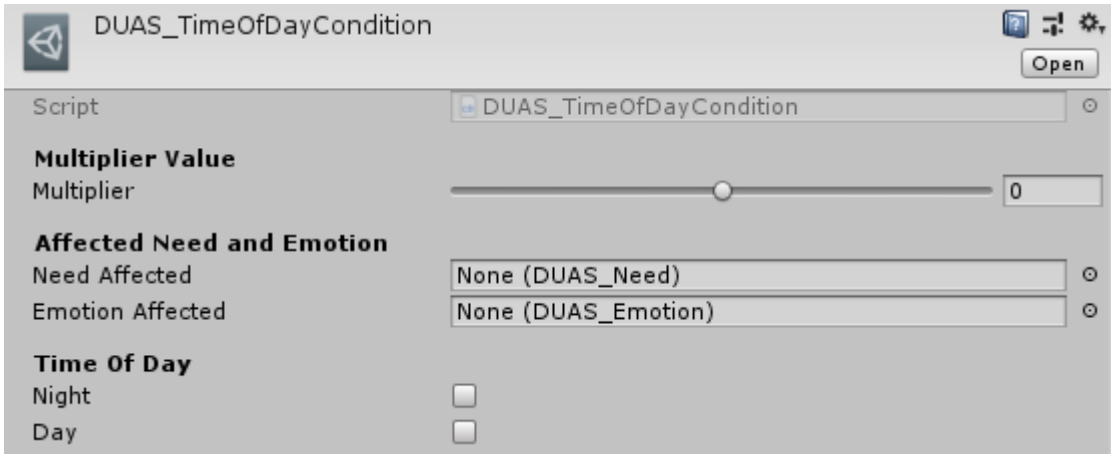

Figure 18. Time of Day Condition inspector view

Deriving from DUAS_Condition, the **Multiplier** is the amount that the **Need Affected** and **Emotion Affected** are changes by each second while the condition is true. The **Need Affected** is the need and **Emotion Affected** is the emotion that will have its value changed while the condition is true.

Night when set to true, will constantly check if the Sun's X rotation is between 180 and 360, and if so, the condition will be met. **Day** when set to true, will constantly check if the Sun's X rotation is between 0 and 180, and if so, the condition will be met.

This type of Condition is the only type that needs to be passed into DUAS_Agent, this will be discussed in the Agents section.

Action Condition

Action Conditions are conditions that rely on if a certain action is being performed. They can be created by right clicking the project folder and selecting Create -> DUAS_Condition -> DUAS_ActionCondition, however, these types of Conditions are automatically created if an Action that applies to the NPC has **Need Affected Multipliers** and/or **Emotion Affected Multipliers** set up.

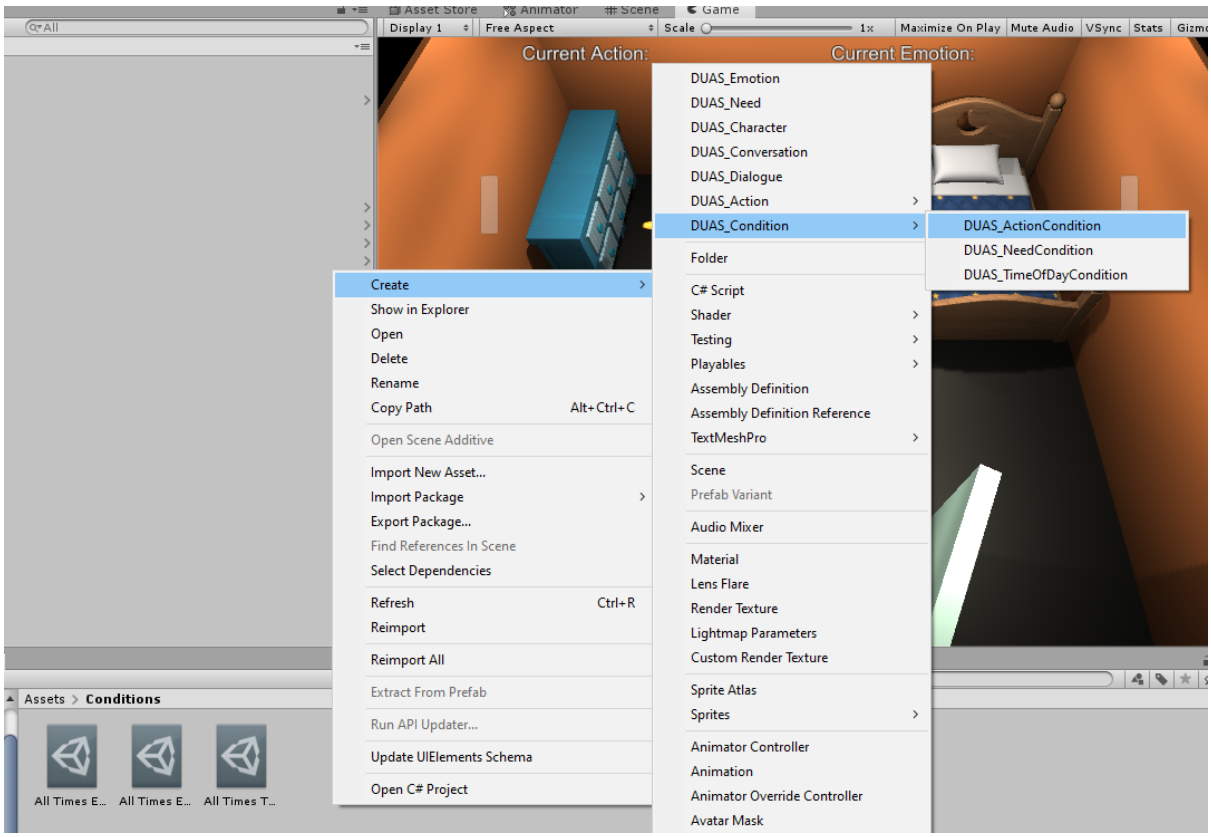

Figure 19. Creating an Action Condition from the Project folder

This means that creating these Action Conditions manually are unnecessary if you have Actions with **Need Affected Multipliers** and/or **Emotion Affect Multipliers**, such as the Drink example below.

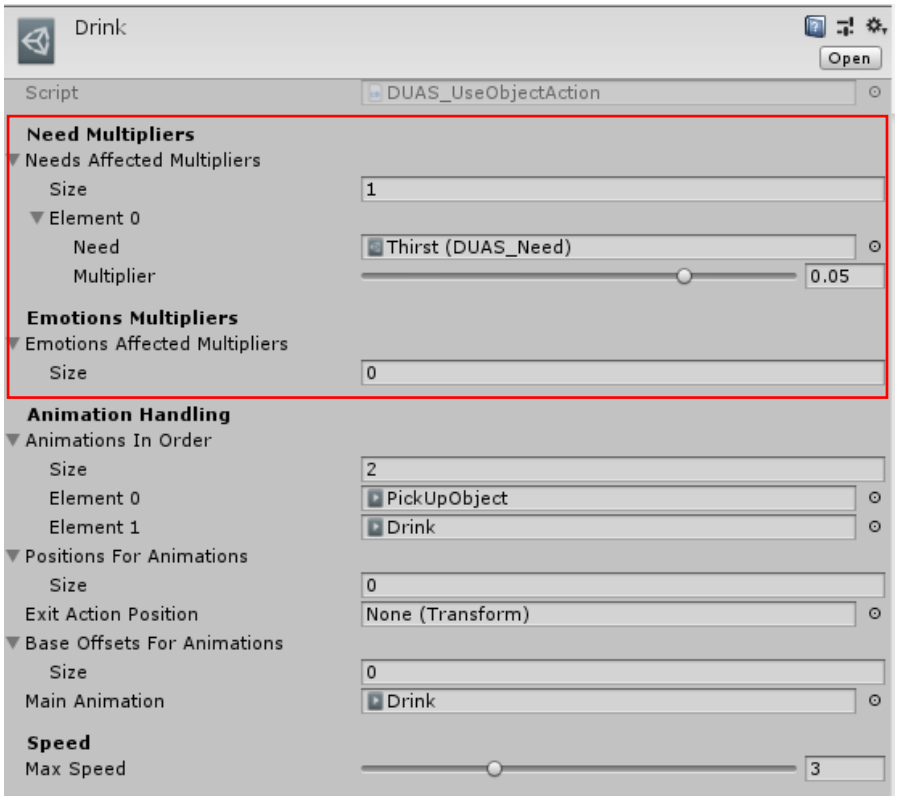

Figure 20. Example 'Drink' Use Object Action (highlighting to Needs and Emotion Affected Multipliers that will automatically create Action Conditions)

Automated Action Condition creation applies to every NPC in the scene.

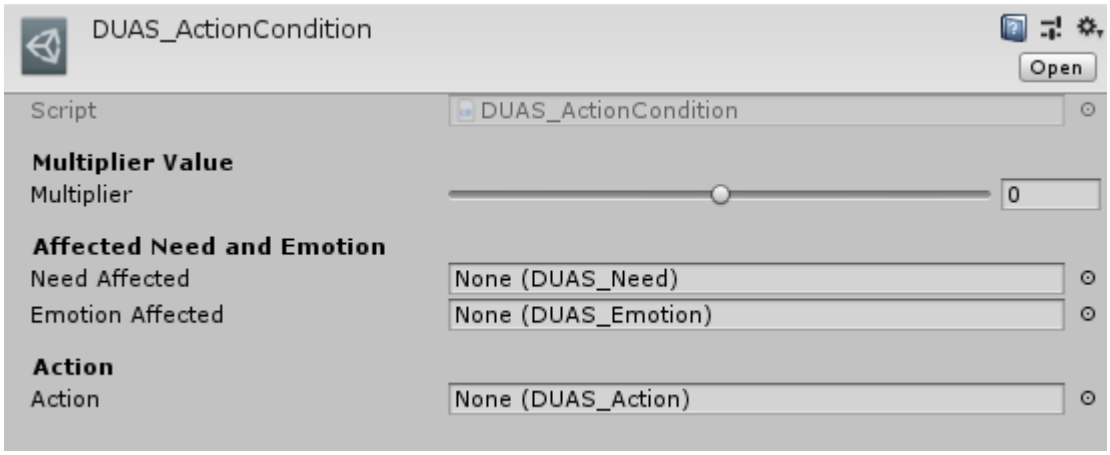

Figure 21. Action Condition inspector view

Deriving from DUAS_Condition, the **Multiplier** is the amount that the **Need Affected** and **Emotion Affected** are changes by each second while the condition is true. The **Need Affected** is the need and **Emotion Affected** is the emotion that will have its value changed while the condition is true.

The **Action** is the type of action that will trigger the condition to be true if the NPC is performing it.

Need Condition

Need Conditions are conditions that rely on if a certain need is above or below a specified value. They can be created by right clicking the project folder and selecting Create -> DUAS_Condition -> DUAS_NeedCondition, however, these types of Conditions are automatically created if need that applies to the NPC has **Emotion Affected Multipliers** set up.

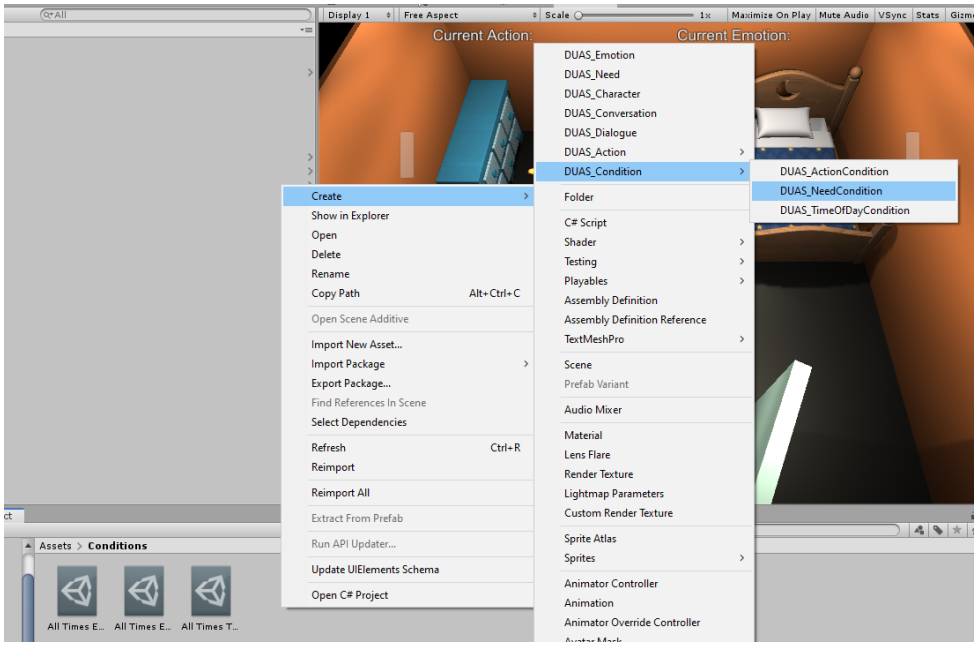

Figure 22. Creating a Need Condition from the Project folder

This means that creating these Need Conditions manually are unnecessary if you have Needs with **Emotion Affect Multipliers**, such as the Energy example below.

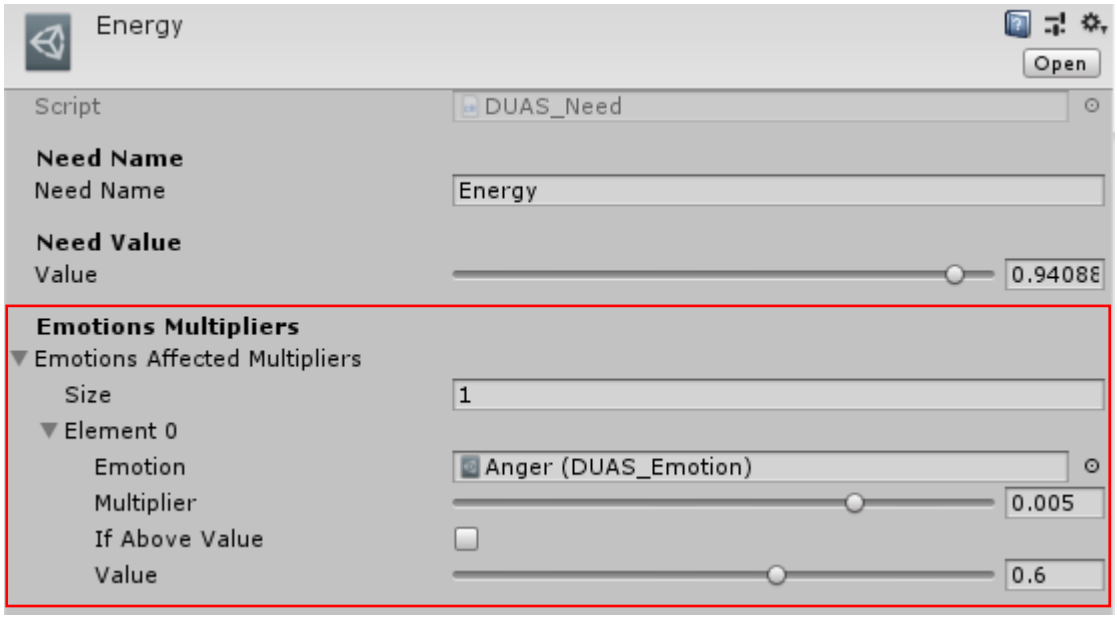

Figure 23. Example 'Energy' Need (highlighting the Emotion Affected Multipliers that will automatically create Need Conditions)

Automated Need Condition creation applies to every NPC in the scene.

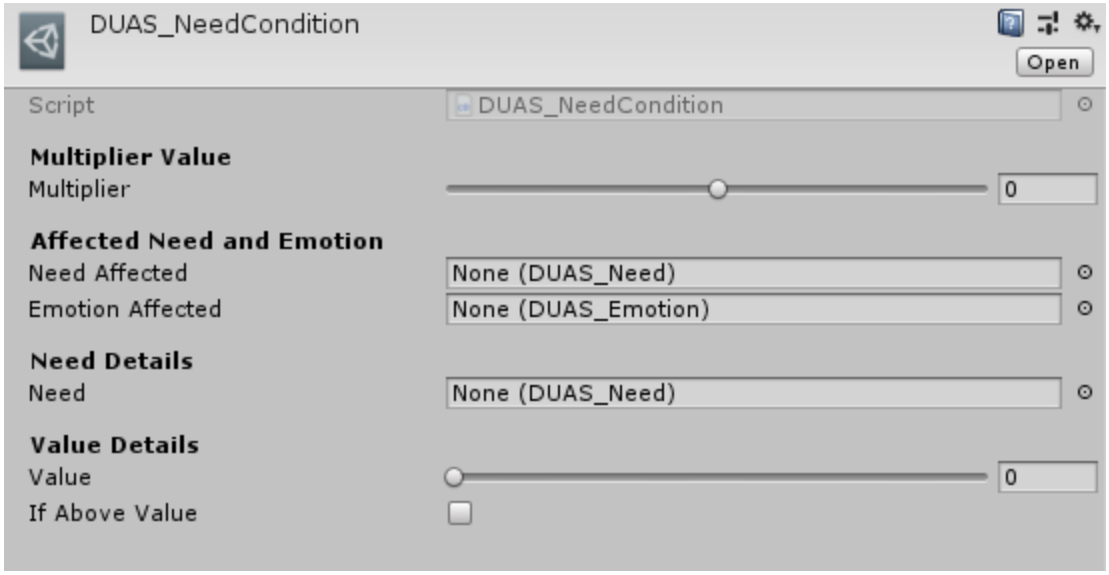

Figure 24. Need Condition inspector view

Deriving from DUAS_Condition, the **Multiplier** is the amount that the **Need Affected** and **Emotion Affected** are changes by each second while the condition is true. The **Need Affected** is the need and **Emotion Affected** is the emotion that will have its value changed while the condition is true, however, for Need Conditions, **Need Affected** should remain empty.

Need is the need that will have its value checked if it meets the conditional **Value**.

Value is the value that will either check above or below this value (based on the **If Above Value** field below) and check if it meets the condition. If the **If Above Value** is true, the condition will check the **Value** if it is of the **Value** or higher, and if false, will check if it is of the **Value** or lower.

Agents

The DUAS_Agent script used to provide models with AI behaviour. Upon attaching the script to the parent Game Object of the model, the Nav Mesh Agent, Sphere Collider and Capsule Collider components will also attach to the model. The Nav Mesh Agent should be sized around the model appropriately using the Radius and Height fields. The Sphere Collider must be a trigger, this is used to detect if a player is within the trigger to allow for the NPC to have a conversation. This trigger should be in front of the NPC. The Capsule Collider is a collider to allow for the NPC Game Object to be detected by other triggers (if necessary). An example of these components size and position is shown below.

Set the tag of the GameObject to “NPC”.

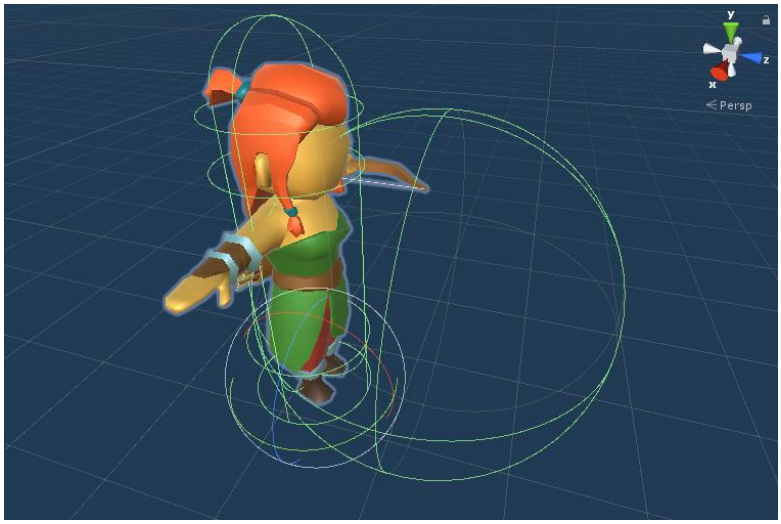

Figure 25. The Archer NPC's Sphere trigger, Capsule Collider and Nav Mesh Agent sizes and positions

There are many fields to the DUAS_Agent script, however, most of which have been covered in the previous sections.

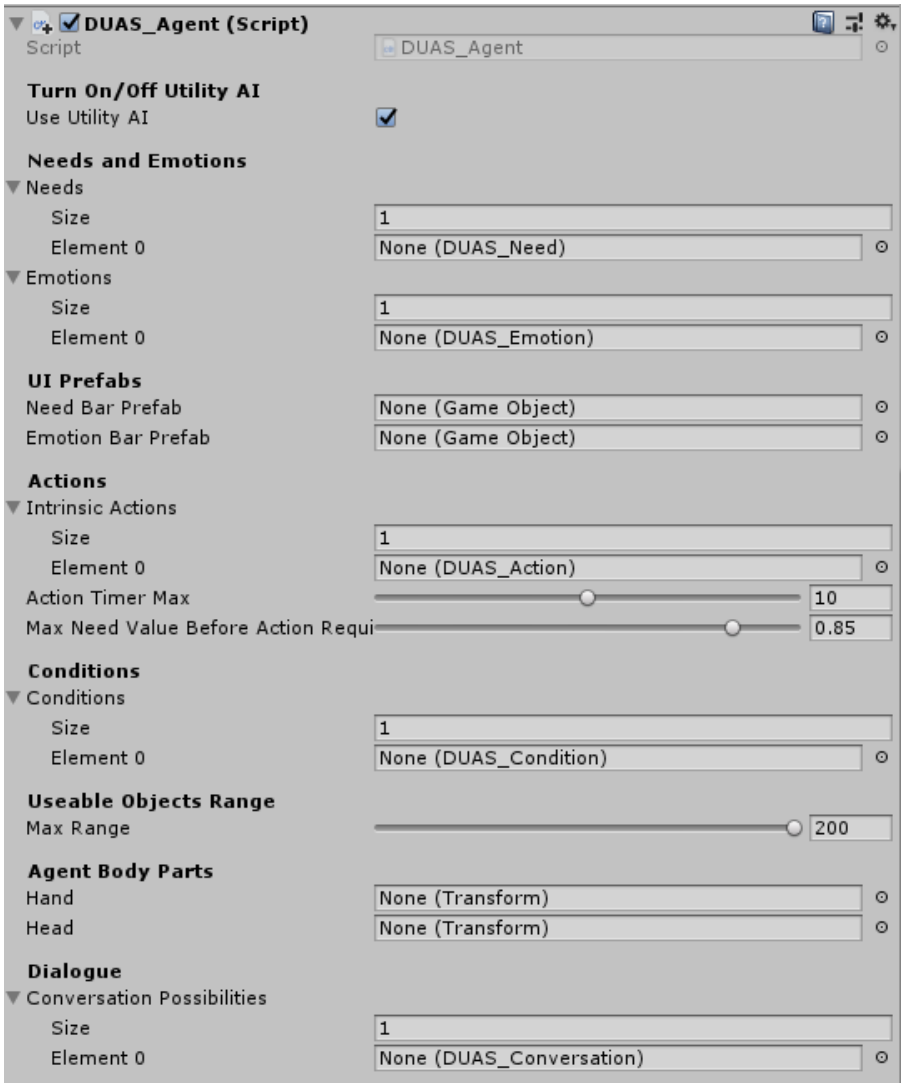

Figure 26. Agent inspector view

Use Utility AI is a Boolean value used to determine if the NPC should perform actions and have its needs and emotions affected (set to true) or if it should remain idle in place (set to false).

The **Needs** field take all the needs and the **Emotions** field takes all the emotions created that apply to this NPC. If the NPC does have emotions, it needs to have an emotion called 'Happiness', as the Happiness emotion is determined by the values of all the other emotions (assuming they are negative emotions).

Need Bar Prefab and **Emotion Bar Prefab** needs to take in the premade prefabs Need Bar and Emotion Bar, which are shown in the Prefabs folder below. These will be instantiated per need and per emotion if the AI Canvas is being used.

Figure 27. Emotion Bar and Need Bar prefabs in the Prefab folder

Intrinsic Actions are any actions that are not a Use Object Action, so this is where the Wander action should be placed. Note: Use Object Actions will automatically be applied to the NPC based on its surroundings. **Action Timer Max** is the time between agent evaluations their needs and deciding on their next best action. **Max Need Value Before Action Required** is used to stop the NPC from evaluating their needs and possible actions, if the needs are above the value provided in this field.

Conditions will automatically fill up any Action or Need Conditions based on if the Action/Need has **Need Affected Multipliers** and/or **Emotion Affected Multipliers**. This means that the only conditions that need to manually be put into this field are Time of Day Conditions.

The **Useable Objects Range** is the maximum distance at which the NPC can find Useable Objects. Any Useable Object (with collider and "Useable" tag set up) within this range will be a potential Use Object Action for the NPC.

The **Head** and **Hand** fields simply take the child Game Object of the model that corresponds to the model's head and hand. These are used to have the NPC look at the player when they are in range of interacting with and assigning a parent Game Object for picking up a holdable object.

Conversation Possibilities is the field where any conversations with this NPC are passed in. The following sections will discuss conversations and dialogue further.

The last step to setting up the agent, is attaching an Animation component to the parent Game Object of all the child body parts. Set the size of Animations to the amount of different animations that this NPC can perform (each NPC must have an 'Idle', 'Walk' (if **Use Utility AI** is true) and 'Talk' (if using the Dialogue system) animation – named exactly the same). An example of this set up is shown in the image below.

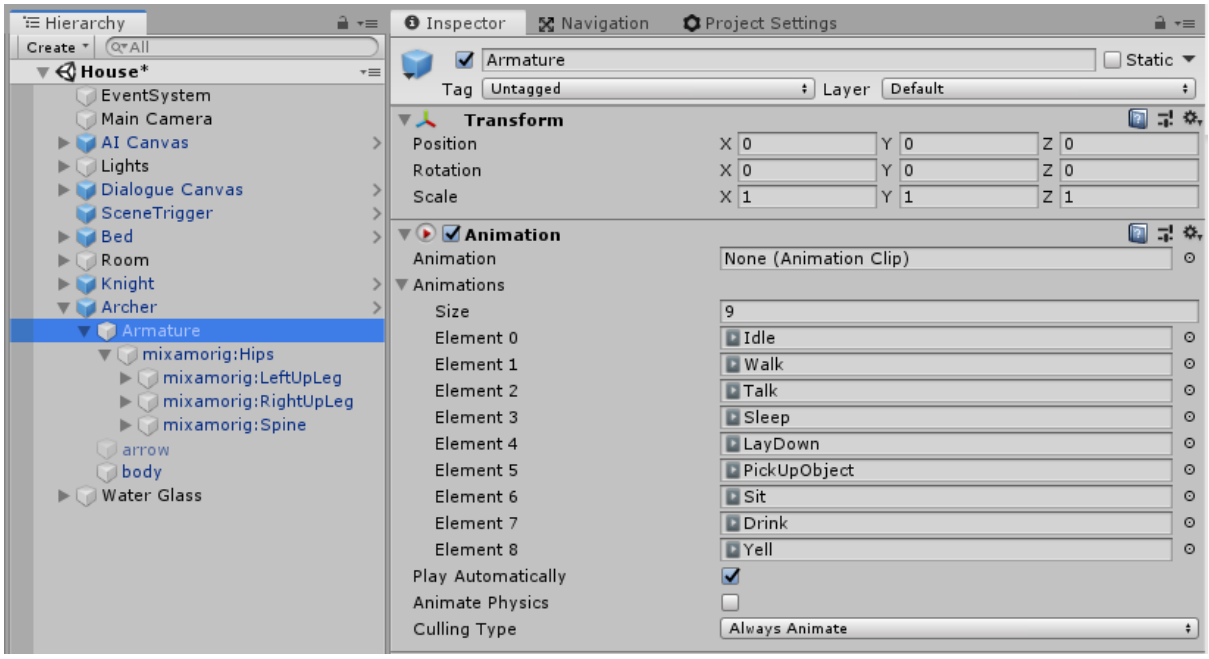

Figure 28. Animation component with animations on the Archer NPC

Dialogue Setup

Dialogue Canvas Prefab

The Dialogue Canvas prefab is a necessary asset for developers to use if they wish to have the Dialogue System in their game. This prefab must be placed into the scene if dialogue is desired.

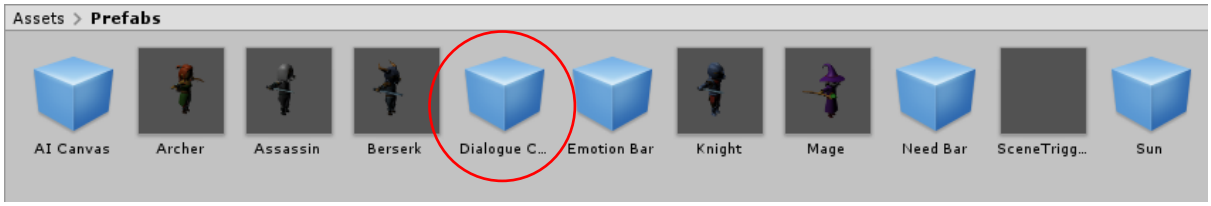

Figure 29. The Dialogue Canvas prefab in the Prefabs folder

By placing this prefab into the scene, the user will be able to view dialogue sequences with customisable character portraits, audio, animations, dialogue boxes, name text and dialogue text.

Upon placing this prefab in the scene, nothing will be visible as the system will only turn on during the dialogue sequences, however, below is an example image of what dialogue can look like.

Figure 30. Example of how the Dialogue Canvas can look during gameplay

Dialogue Manager

Attached to this prefab is a script called DUAS_DialogueManager, this handles all of the functionality for the Dialogue System.

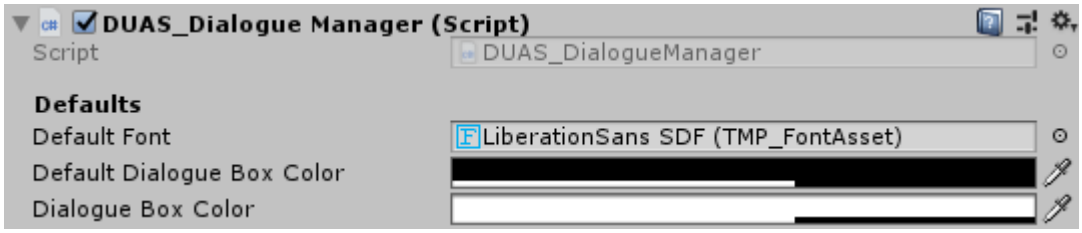

Figure 31. Dialogue Manager inspector view

The **Default Font** is a font that is used if the character and dialogue do not have a specified font (this will be discussed in following sections, Characters and Dialogue). There must be a font within this field.

The **Default Dialogue Box Color** is used for when there is no dialogue box passed into the character who is currently talking. This will create a blank image used in place of the dialogue box and the colour specified in this field will be the colour of this image.

Dialogue Box Color is the colour used for character dialogue boxes that have been specified. This allow for the image to be transparent or have a colour tint.

Characters

Characters are Scriptable Objects that are used for character-specific customisations in dialogue sequences. They can be created by right clicking the project folder and selecting Create -> DUAS_Character.

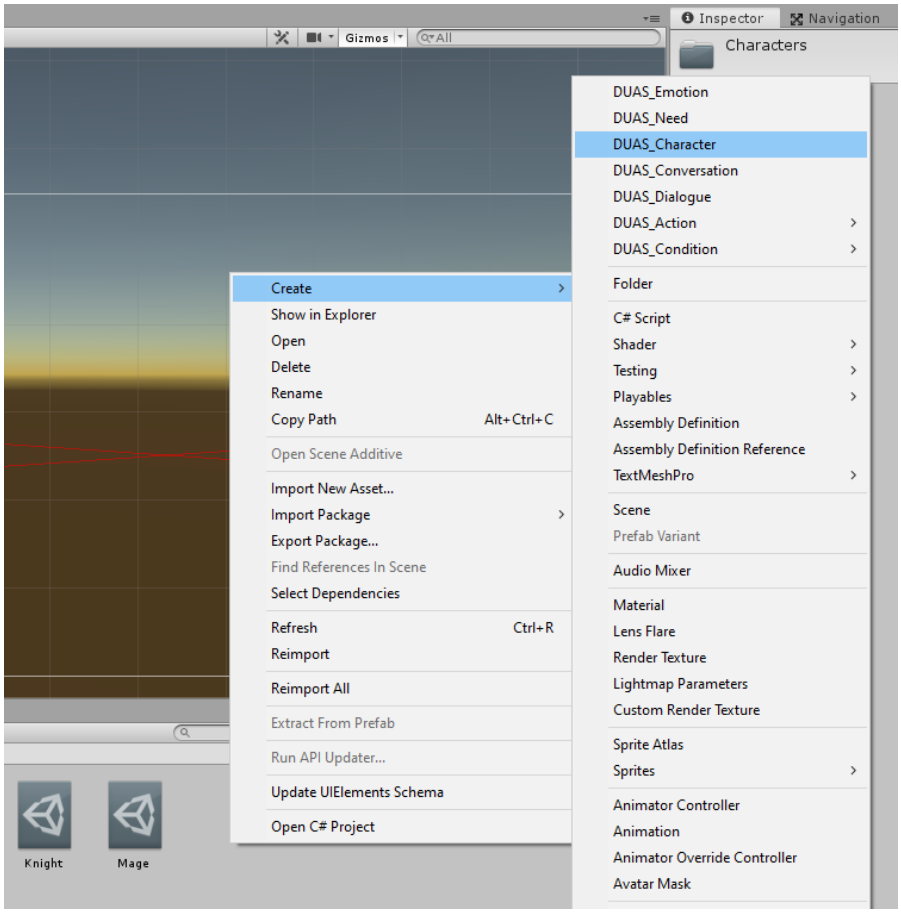

Figure 32. Creating a Character from the Project folder

A Character must be made for every NPC that will require dialogue.

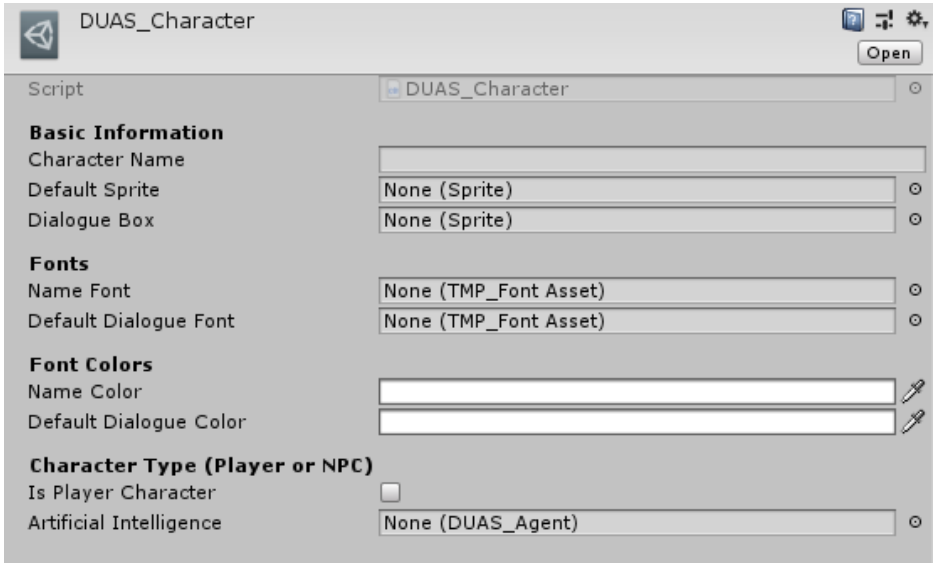

Figure 33. Character inspector view

The **Character Name** is what will be displayed during dialogue sequences as the name of the speaking character. **Default Sprite** will be used for the character's portrait during dialogue sequences when other sprite is not specified in Dialogue (discussed in the next

section). The **Dialogue Box** is the sprite that will be used for the background image for dialogue sequences with this Character.

Name Font is used for the name text during dialogue sequences with this Character. **Default Dialogue Font** is used for dialogue text during dialogue sequences with this Character when other font is not specified in Dialogue (discussed in the next section).

Name Color is used to set the name text colour and **Default Dialogue Color** is used for dialogue text colour during dialogue sequences with this Character.

The **Is Player Character** field should be set to true if the Character is the player's Character, otherwise, if this is an NPC, it should be set to false (this will determine if the Character portrait is displayed on the left or right side of the dialogue box. If this is an NPC, then the **Artificial Intelligence** field should have the NPC prefab that this Character represents passed in (this must be passed in from the project folder – remember to update the prefab when changes have been made).

Dialogue

Dialogues are Scriptable Objects that are used for containing information about the Dialogue (e.g. the Character speaking, the sentences they speak, text speed, etc.). They can be created by right clicking the project folder and selecting Create -> DUAS_Dialogue.

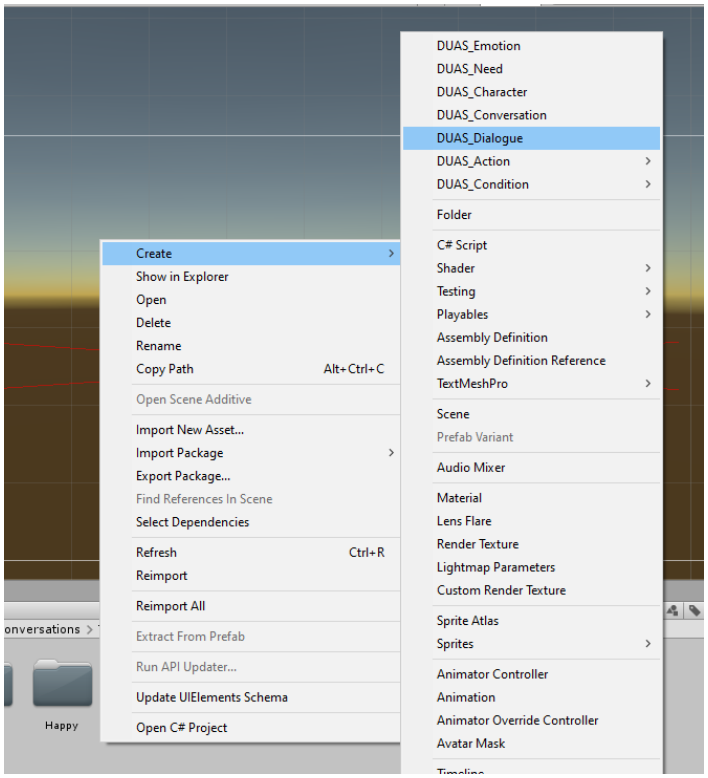

Figure 34. Creating a Dialogue from the Project folder

There are many fields to the DUAS_Dialogue script, but majority of them are optional.

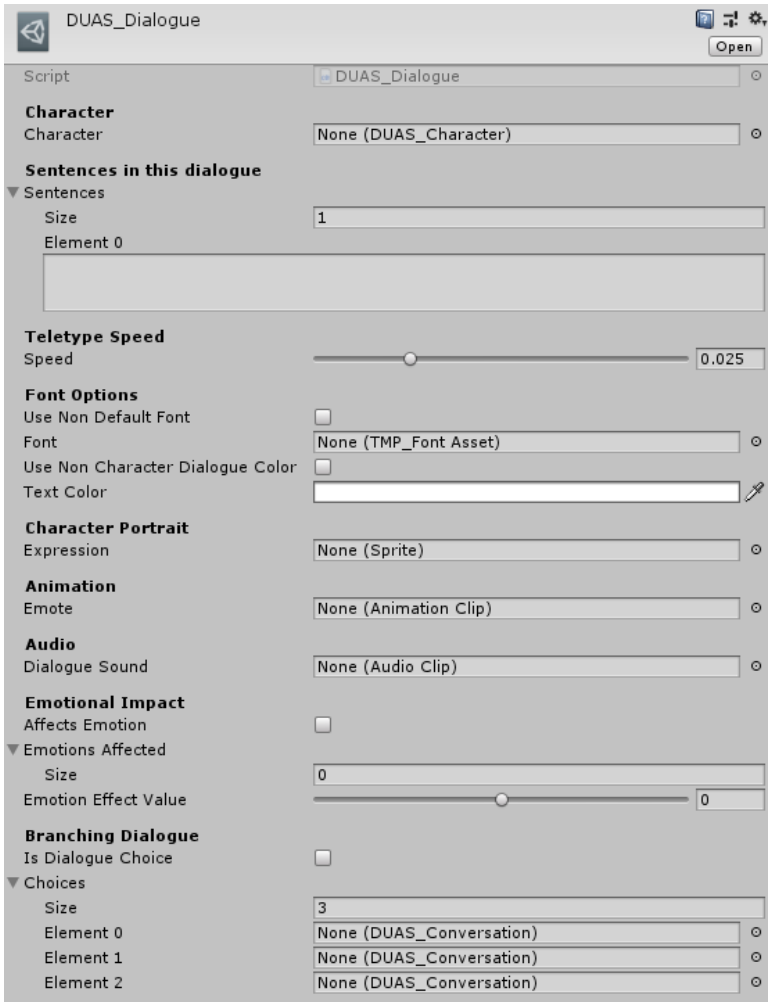

Figure 35. Dialogue inspector view

The **Character** is the Character that will be speaking this Dialogue, this is used for displaying the corresponding name, fonts, font colours and dialogue box during dialogue sequences.

Sentences is a list of strings that will display one at a time during dialogue sequences, **Size** should be set to the number of chunks of text that this Character will say before swapping to the next Dialogue.

Speed is the speed at which the dialogue text will teletype.

The following fields of Dialogue are optional.

If **Use Non Default Font** is set to true, then the **Font** specified in the field below will be used instead of the Default Font of the Character or Dialogue Manager. If **Use Non Character Dialogue Color** is set to true, then the **Text Color** below will be used in place of the Character's Default Dialogue Color.

Expression is a sprite field used to change the Character's portrait (for this Dialogue only) to provide different expressions and **Emote** is an animation field used to change the NPC's animation for this Dialogue only to provide different body language for varying Dialogue tones.

The **Dialogue Sound** field takes an AudioClip that can be an audio recorded version of the Dialogue or even voice interjections and plays it at the beginning of this dialogue.

If the **Affects Emotion** field is set to true, the emotions within the **Emotions Affected** field list will have its value changed by the float amount put into the **Emotion Effect Value**.

If the **Is Dialogue Choice** is set to true, then the choice buttons will display with the **Choices'** Choice Title (discussed in the Conversations section). **Choices** is a list of 3 conversations that can be chosen from by the player, when a choice button is clicked, the conversation relating to that button will trigger.

Conversations

Conversations are Scriptable Objects that are used for containing the ordered list of Dialogues that are a part of a whole Conversation. They can be created by right clicking the project folder and selecting Create -> DUAS_Conversation.

These objects are within the DUAS_Agent script for assigning possible conversations to the NPC and DUAS_Dialogue script for branching dialogue options through player choices.

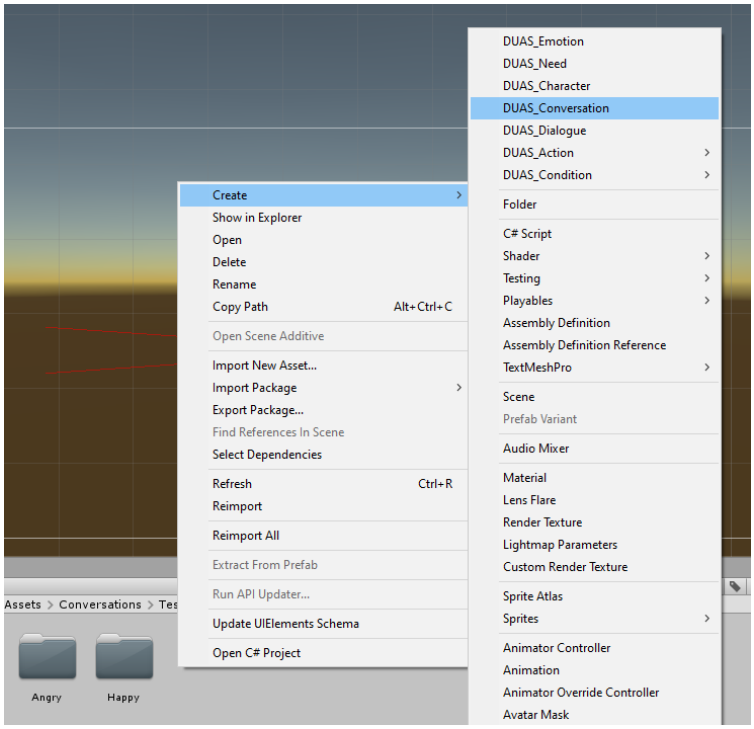

Figure 36. Creating a Conversation from the Project folder

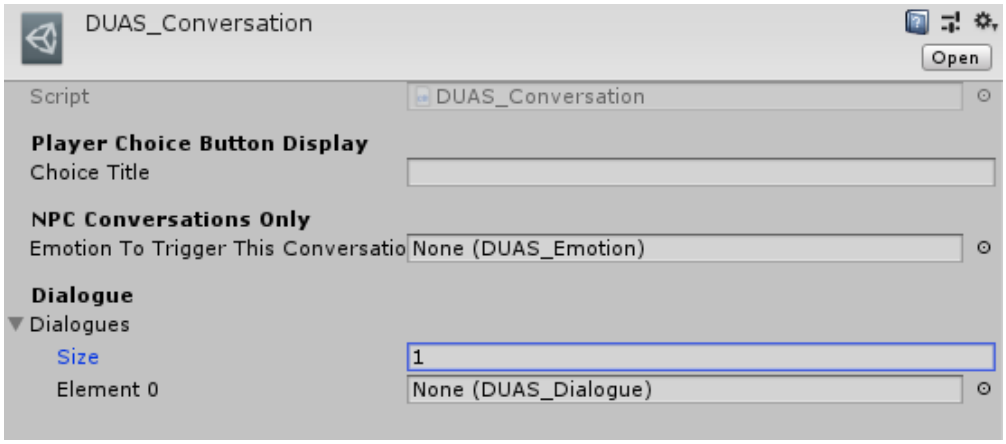

Figure 37. Conversation inspector view

The **Choice Title** is an optional field for if this Conversation is part of a player choice. This text will be displayed on the Choice buttons (so it should be concise).

Emotion To Trigger This Conversation requires an emotion that the NPC has and needs to be the highest Emotion at the time for it to be a possible triggered Conversation in-game.

Dialogues is a list of Dialogues that make up the entire Conversation and must be in order of how they should be spoken.

Demos

Demo Scenes

The demo scenes in this package, make use of an example Player Game Object. It has a player script that slows for this system to work with it (refer to this script when creating a custom Player Game Object. Note: The Player must be tagged with the 'Player' tag.

House

The House scene has one NPC, the Archer, and the Player. The Archer has been given 3 needs, Entertainment, Energy and Thirst, and 2 emotions, Happiness and Anger. She also has 3 Time of Day Conditions that decrease each need value consistently over time. Once one of the need values is less than 0.85, then she will begin to perform actions other than Wander. The Archer should cycle through wandering around, sleeping on the bed and drinking form the glass of water. The Drink action is an example of a holdable object action, and the Sleep action is an example of a non-holdable action that requires a series of prior animation and set positions.

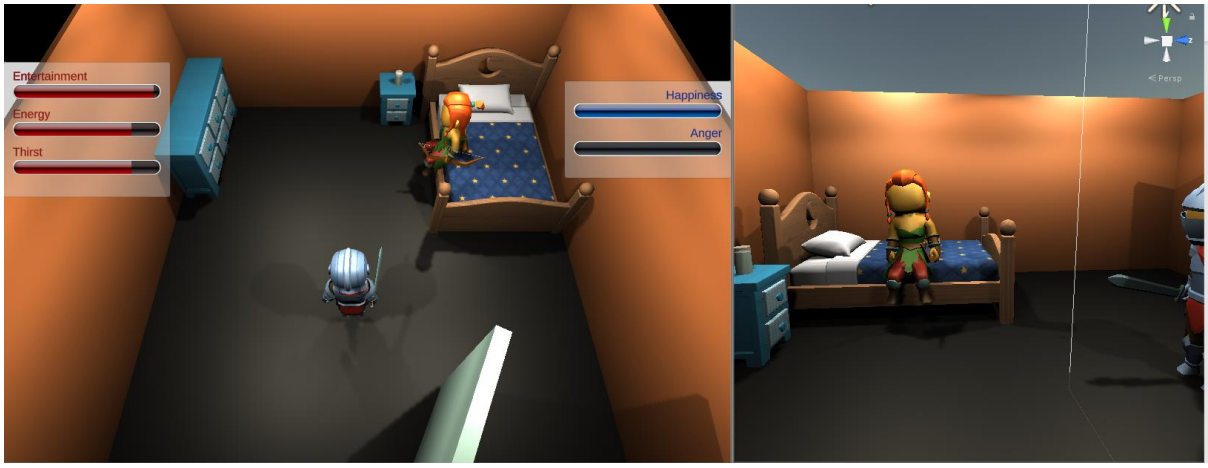

Figure 38. House scene with NPC and Player

The player can interact with the Archer by walking close to her forward direction (in front of her) and pressing Spacebar. If she is happy, she will trigger the Archer-Happy conversation that provides dialogue choices to the player. This specific conversation showcases how teletype speed, text colour, expression, emote and emotional impact fields work where the dialogue choice 'You're right', will cause the Archer to get mad and perform more aggressive emotes and expressions. Once this option has been selected, if the player attempts to talk to her again, she will trigger the Archer-Angry conversation where the Archer will yell at you. In addition, if the Archer is tired (has an Energy value of 0.6 or less), she will slowly become angrier, which will in turn make her happiness drop.

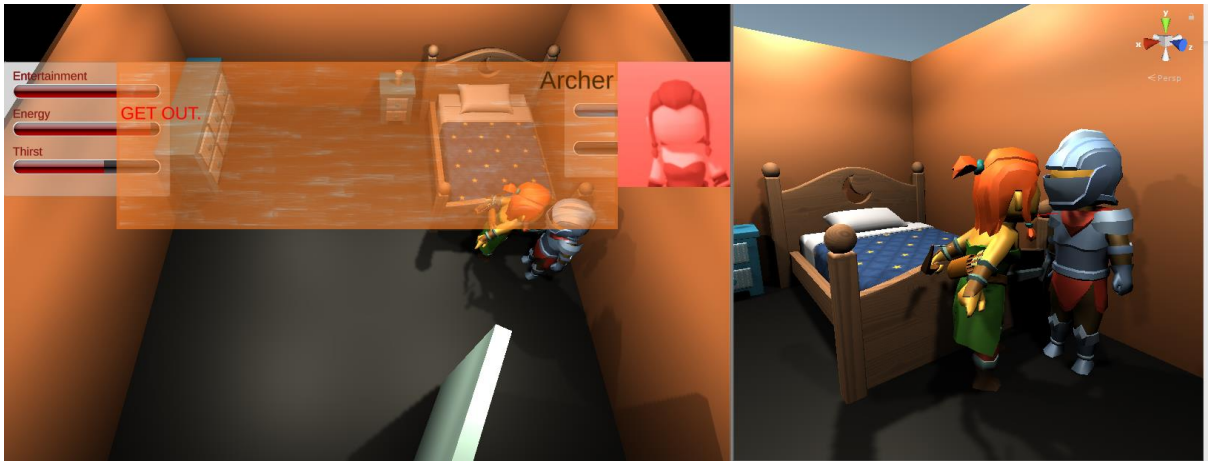

Figure 39. NPC angry response to Player dialogue choice

This scene connects to the Overworld scene by walking through the front door of the House (trigger object).

Overworld

The Overworld scene has three NPCs, the Assassin, Berserk and Mage characters, and the Player.

Figure 40. Overworld scene with 3 NPCs and Player

Each of the NPCs have been given the Entertainment need as this demonstrate the old-school RPG style of NPC behaviour. The Mage has its Use Utility AI field set to false, keep him at the top left house performing the Idle action. He can still have a conversation with the player if the player interacts with him.

The Assassin and Berserk characters only have the Wander action, causing them to walk around and be idle. The player can interact with both characters to have a conversation with them.

Figure 41. Example Conversation from Overworld scene

This scene is used to showcase how this system can work in a much less complex manner compared to the House scene. This Overworld scene also connects to the House scene by walking up to the front door of the house that you spawn in front of in the Overworld scene (trigger object).

Demo Scripts

Camera Control

The CameraControl script requires the player to be set as the **Target** object and has the camera follow its movement applying a **Damp Time** value for smooth movement.

Load Scene

The LoadScene script uses a trigger to check if the player entered the trigger and loads the other demo scene.

Player

The Player script handles player movement, animations and head rotation during conversations. It takes a **Speed** value for how quick its movement is, and the **Head** field is for the model's head object.

Sun Rotation

The SunRotation script takes a **Rotation Speed** for how quickly the day/night cycle runs.

Package Exclusions

Within this package, there are demonstrations that use assets that were not created by Renae Aurisch, these assets include:

- **Mixamo animations** (used for Knight, Archer, Assassin, Berserk and Mage NPCs)
  - Sourced from Mixamo.com
- **Audio files** (used for example of dialogue sounds)
  - Created by Fusehive Interactive Media Ltd.
  - Purchased from humblebundle.com
- **Bed & Bath Furniture Pack** (used for creating the House demo scene environment)
  - Created by Pepperjack
  - Sourced from Unity Asset Store
- **HandTorch** (used for creating the Overworld demo scene environment)
  - Created by SimpleModelsForMe
  - Sourced from Unity Asset Store
- **HyperCasual_FantasyCharacters** (textured models used for NPC examples)
  - Created by Stegobubbles
  - Purchased from Unity Asset Store
- **MedievalTownExteriors** (used for creating the Overworld demo scene environment)
  - Created by Lylek Games
  - Sourced from Unity Asset Store
- **TextMesh Pro** (used for improved text for Dialogue Canvas and AI Canvas)
  - Sourced from Unity Packages
- **bars.png** (AI Canvas needs and emotion bars)
  - Sourced from Google Images