the following documentation file is work-in-progress

Skene User Guide

last updated 01-12-2015, Tiago Ribeiro

1. Introduction

Skene is a semi-autonomous behaviour planner that has been developed in the EMOTE project. It is designed to be reused in different applications and embodiments, and in SAIBA fits as a behaviour planner. In an interaction environment consisting of both robots and other devices (e.g., multi-touch table (MTT)), Skene is the component in which all of them meet (**Figure 1**). Its input is a high-level behaviour description language (Skene Utterances) and perception information (such as target locations). The utterances can also be loaded from a pre-authored Utterance Library and invoked through a category/intention. Its output consists both of scheduled BML and also non-BML actions (like sounds or application commands).

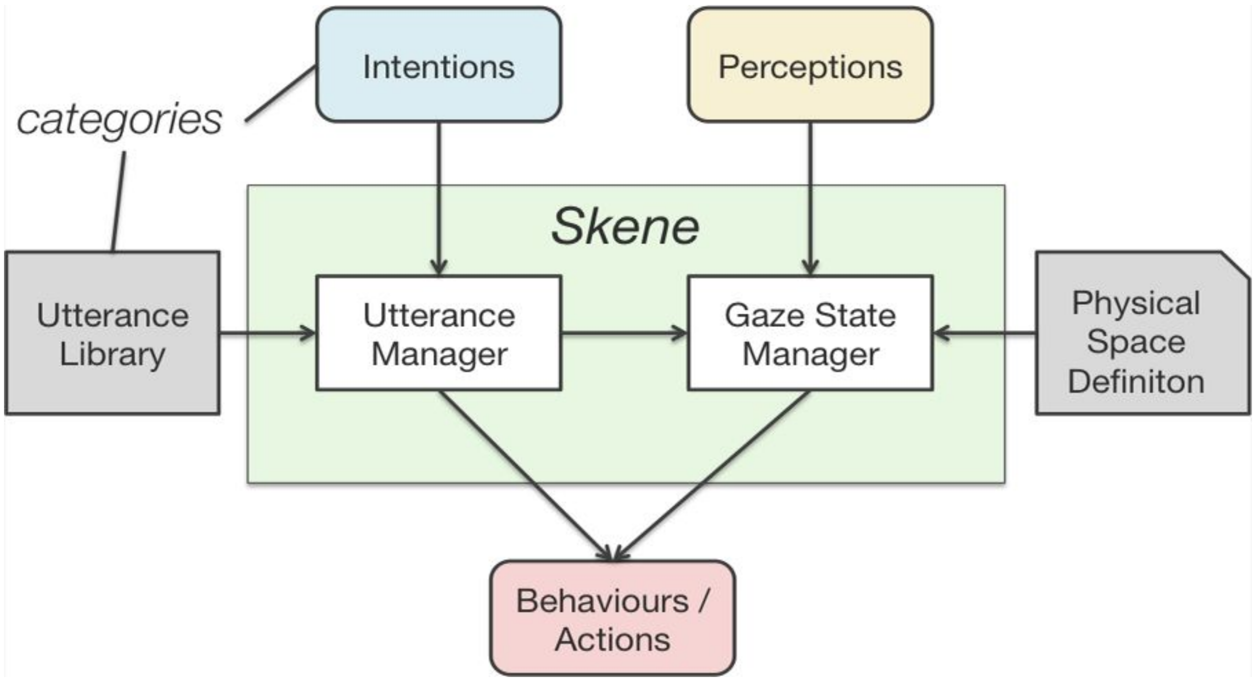

Figure 1: Components of Skene.

2. Skene GUI (ToDo)

2.1. Physical Space Setup

2.2. Gaze and Point

2.3. Targets

2.4. Idle Behaviour

2.5. Utterances

3. Skene Utterances

An utterance in Skene is a multi-modal instruction that may contain speech, gazing, pointing, waving, head nods, facial expressions, expressive animations or application/screen instructions. Here we present an example utterance:

<GAZE(*studentTwo*)> <ANIMATE(*metaphDicotomicLeft*)> In order to have more space to draw, <GLANCE(*iPadOne*)> press <HEADNOD(*2*)> the eraser icon. <GLANCE(*iPadOne*)> <POINT(*iPadOne*)> <pause=300> Just try it. <GLANCE(*iPadTwo*)>

The next sections specify the type of instructions that can be included in utterances along with what their parameters means.

3.1. Gaze/Glance

<GAZE(*target*)> Changes the gaze state to *target*.
<GLANCE(*target*)> Glances quickly (~1-2sec) at *target* and then goes back to gazing at the current gaze state.

3.2. Point/Wave

<POINT(*target*)> Points at *target*, if the embodiment is somehow able to point.
<WAVE(*target*)> Points at *target*, and waves, if the embodiment is somehow able to point.

3.3. Screen Highlight

<HIGHLIGHT(*target*)> Instructs the multimedia application (if any) to highlight the on-screen *target*.

3.4. Composed Gaze/Glance/Point/Wave/Hightlight

These all work as in the previous sections, with a compact representation.
<GAZEANDPOINT(*target*)>
<GAZEANDWAVE(*target*)>

<GLANCEANDPOINT(*target*)>
<GLANCEANDWAVE(*target*)>
<GAZEANDHIGHLIGHT(*target*)>
<GLANCEANDHIGHLIGHT(*target*)>
<POINTANDHIGHLIGHT(*target*)>
<WAVEANDHIGHLIGHT(*target*)>
<GAZEANDPOINTANDHIGHLIGHT(*target*)>
<GLANCEANDPOINTANDHIGHLIGHT(*target*)>
<GLANCEANDWAVEANDHIGHLIGHT(*target*)>

3.5. Animate

<ANIMATE(*animationName*)> Instructs the current animation system to perform *animationName*. This is non-blocking and will cause no issue if either the animation or animation system do not exist.

3.6. Head nods

<HEADNOD(*count*)> Performs an affirmative head-nod *count* times.
<HEADNODNEGATIVE(*count*)> Performs a negative head-nod *count* times.

3.7. Face expression

<FACE(*expressionName*)> Temporarily exhibits *expressionName* on the embodiment's face.
<FACESHIFT(*expressionName)*> Exhibits and holds *expressionName* on the embodiment's face.
<EYEBLINK(*count*)> Blinks the eyes *count* times at a normal speed.
<SLOWEYEBLINK(*count*)> Blinks the eyes *count* times at a slow speed.

3.8. Game instructions

<GAME(*instructionName, parameters*)> Sends instruction *instructionName* to the multimedia application (if any) with parameters *parameters*. This is used to have the robot saying something, even performing nonverbal behaviour, and only then performing an action in the application.

4. Utterances Verifier (ToDo)

5. Quick Start Guide

5.1. Create your Utterance Library.

You can copy an existing one and add your stuff, from **SERA\Apps\LeadPoetsSociety\Utterances**, or by duplicating this google spreadsheet: https://docs.google.com/spreadsheets/d/1j5Dxs8qD_uNYytIwn8OaZ9nI7S86nusWVeKuRNUPazE/edit?usp=sharing

5.2. Download the library to your computer from Google Spreadsheets:

File -> Download as -> Microsoft Excel (xlsx).
Save the file to the **SERA\Binaries\Skene\Utterances** folder.
You can optionally delete the xlsx files that are already there (from EMOTE).

5.3. Start Skene.exe.

If you haven't done so before, install the Microsoft Office Access Database Engine (this isn't installed even if you have Office installed). You only have to do this once. The installer can be found in **SERA\Binaries\Skene\INSTALL FIRST**
In the first tab (Physical Space Manager) make sure the checkboxes aren't empty.
If they are, load the "Multitaction" setting.

5.4. Navigate to the Targets tab

You can see a set of built-in targets that can be used within the utterances.
TO DO: Add a guide for creating your own targets.

5.5. Navigate to the Utterances tab

Make sure your library is on the list and select it.
It is highly recommended to switch off Use Composite Library.
Uncheck the Relative Speed box.

Note: Most of your settings in this GUI get saved so you don't have to change everything again next time Skene starts. This does not work with all the settings! So test it before you rely on it :)

5.6. Try the utterances

Make sure you have both **Skene** and the **SpeechClient** connected to your Thalamus character.
Select an utterance from the list and click Perform.
Troubleshoot:
In case you don't hear anything, make sure your speakers are at an appropriate volume.
Also check the SpeechClient to make sure it is printing out what it's speaking.
If it still doesn't work check your default sound device for windows TTS (google it).
If you hear "slash r s p d 75" you didn't uncheck the Relative Speed option.

If all seems fine but it interrupts what it's saying after 2 or 3 seconds something is working wrong with your TTS. You can however workaround it by changing the Utterance Timeout to something big like 20000 (it's in miliseconds so this is 20sec).

5.7. Publish from your Thalamus client to Skene

In your Thalamus client, add a reference to **SERA\Binaries\Skene\EmoteEvents.dll**.
Now in your publisher's interface, add **EmoteCommonEvents.IFMLSpeech** and implement the missing methods (not the explicitly one).
You can now invoke Skene to perform an utterance of a specified **category:subcategory** by publishing the **PerformUtteranceFromLibrary** message.
This message takes 5 parameters:

- id - a string-based identifier in case you need to be notified when a particular utterance has started or finished. Leave empty **""** otherwise;
- category and subcategory are the indexers to fetch utterances from your utterance library;
- tagNames and tagValues allow you to replace parts of the fetched utterance with real-time/execution data.
- example: imagine that in the utterance library there are three utterances:
  "Hello dear |name|"
  "Hey |name|, how are you?"
  "Hi |name|, haven't seen you since |lastInteraction|."

all with **interaction:greeting** category.
To perform a greeting you would call:
yourPublisher.**PerformUtteranceFromLibrary**("", "interaction", "greeting", { "|name|", |lastInteraction| }, { "Tiago", "yesterday" })
Skene will randomly select one of the three and replace each tag with its value.

5.8. Control the flow of interaction

Your AI/Decision can control the interaction by subscribing to the **EmoteCommonEvents.IFMLSpeechEvents** which makes it receive the UtteranceStarted and UtteranceStarted messages. If you didn't provide a blank Id in the PerformUtteranceFromLibrary method, you will be able to match your id to these. If you specified an empty Id then Skene will have automatically generated a random GUID.
Note that by adding the IFMLSpeech to your publisher you also get a useful CancelUtterance method.