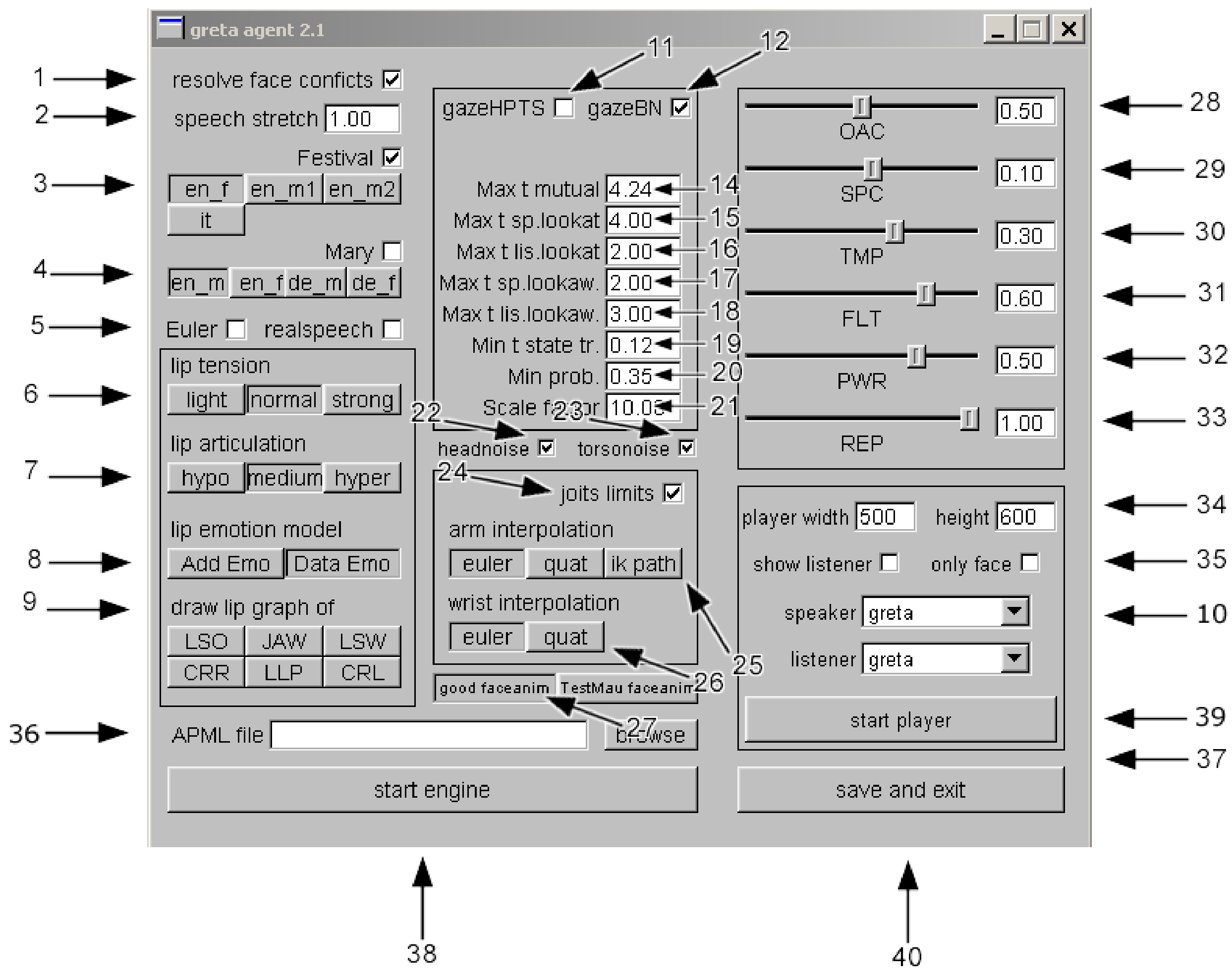
Greta

User's manual
may 2006

Table of Contents

The Greta's interface


greta agent 2.1

resolve face conficts ☑
speech stretch 1.00
Festival ☑
Mary ☐
Euler ☐ realspeech ☐

gazeHPTS ☐ gazeBN ☑
Max t mutual 4.24
Max t sp.lookat 4.00
Max t lis.lookat 2.00
Max t sp.lookaw. 2.00
Max t lis.lookaw. 3.00
Min t state tr. 0.12
Min prob. 0.35
Scale factor 10.05
headnoise ☑ torsonoise ☑
joits limits ☑

OAC 0.50
SPC 0.10
TMP 0.30
FLT 0.60
PWR 0.50
REP 1.00

player width 500 height 600
show listener ☐ only face ☐
speaker greta
listener greta

APML file

As you start Greta a graphical interface is shown. From this interface you can setup all the parameters of the Greta's engine (animation generator) and player (animation viewer). The same parameters can be setup by editing the greta.ini file. So you will never need to go inside this file and manually edit it, except for 4 parameters that are only in the ini file and not in the interface. They are all related to the speech synthesizer used by Greta:

FESTIVAL_PATH=... (example: C:\festival\src\main\festival.exe)
it is the path on your hard drive where the Festival synthesizer executable is (the exe file, and NOT only the install folder of Festival). It is recommended that you install Festival in the root of the C: drive.

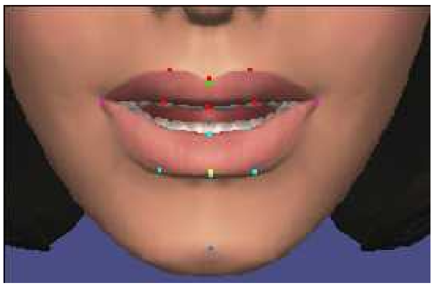
EULER_PATH=... (example: C:\EULER)
it is the path on your hard drive where the Euler system is installed.

EULER_LANGUAGE=FRENCH_FEMALE1
it is the voice that will be used by Euler to synthesize voice.

MARY_ADDRESS=127.0.0.1
it is the IP address of the machine running the Mary server.

The following table describes all the other parameters of the Greta's interface.

Function	***Description***
1	Enables/disables the facial conflict resolutions. In the APML input file there can be two nested tags which activate the same part of the face in a different way. For example a performative that raises the eyebrows can include an affective tag that lows them. In these cases the conflict resolver decides which one of the two conflicting actions will be performed. Without the conflict resolver the resulting animation could present unrealistic movements.
2	This value is used by the speech synthesizer (NOTE: at the moment only the Festival synthesizer considers this parameter) to produce a slower/faster speech. A value of 1 is a "normal" value, lower values will produce a faster speech (0.5 is a very fast speech).
3	By checking this box you choose to use Festival as speech synthesizer for Greta. You need that the Festival system is already installed on your machine. NOTE: Greta needs to know where Festival is and so you have to edit the greta.ini file manually and set the parameter FESTIVAL_PATH by putting the path of the executable file of Festival. At the moment this path cannot contain space. The buttons under the checkbox allow to select which Festival voice to use: english female (en_f), english male 1 (en_m1), english male 2 (en_m2), italian (it). Of course you will select as input an APML file written in the corresponding language. If you wish to use a new language defined in Festival, you need to specify the mapping between the phonemes of the new language and the existing visemes in Greta. Please contact us so we can tell you which file to modify. It is not difficult to do. If you want to use another speech synthesizer you can use Mary and Euler (see points 4 and 5). If you need to use a different speech synthesizer please contact us.
4	By checking this box you choose to use Mary as speech synthesizer for Greta. NOTE: Greta will interact with Mary using a TCP/IP socket. You have to edit the greta.ini file manually and set the parameter MARY_ADDRESS by putting the address of the Mary server. The buttons under the checkbox allow to select which Mary voice to use: english female (en_f), english male (en_m), german female (de_f), german male (de_m). Of course you will select as input an APML file written in the corresponding language.
5	With the first checkbox you can choose to use Euler as speech synthesizer for Greta. You need that the Euler system is already installed on your machine. NOTE: Greta needs to know where Euler is and so you have to edit the greta.ini file manually and set the parameter EULER_PATH by putting the path of the installation folder of Euler. This system supports only the French language. With the second checkbox you can choose to use an existing wav file to drive Greta's animation. NOTE: this function is not implemented.
6	Here you can select the lip tension. Lip tension appears when pronuncing bilabial consonants like "m". With higher tension you obtain an higher pressure between the lips.
7	Here you can select the quantity of lip articulation. Lip articulation is the quantity of movement of the lips during the animation. With higher values you obtain that mouth opens more during talking.

Function	***Description***
8	Here you can choose in which way emotions influence the lip movement. The "add emo" model simply adds the static expression of the emotion (joy, fear, ...) with the pronounced phonemes. The result can be very unrealistic. With the model "data emo" the lip shapes are taken from stored data coming from real movements and the results are much more realistic.
9	By selecting one or more of these buttons you obtain a graph where the movement of the lip in time is represented. The possible paramters are: ULH, LLH, JAW, LW, UP, LP, LC 
10	Greta can use several agent models. They are defined as a combination of a head model and a body model. These definitions are stored in some files in the 'runtime\characters' folder. Specify which character you want to use in 'runtime\greta.ini', in the line 'character_speaker='. The value must correspond to one of the files in the 'runtime\characters' folder, for example if there is a file called "girl.xml" you can write 'character_speaker=girl'. You can create new character files that use existing head and body models. For the moment there is only a woman and a man body, and several head models including the original greta's face. If you want to use a new head/body model, you should contact us as some works need to be done. NOTE: for the moment the quality of the animation can be very poor if you don't use the head model called "greta".
11	This checkbox corresponds to a model for the gaze that uses HTPS++ state machines to decide the behavior. This model has been used for some specific applications and you should not be used.
12	This checkbox corresponds to the more appropriate model for the gaze of Greta and is the one that should be used in normal animations.
13	This checkbox is a new gaze model that is going to replace the original model.
14	This number is the maximum consecutive time that the speaker and the listener should look one to each other.
15	This number is the maximum consecutive time that the speaker should look at the listener.
16	This number is the maximum consecutive time that the listener should look at the speaker.
17	This number is the maximum consecutive time that the speaker should look away from the listener.
18	This number is the maximum consecutive time that the listener should look away

Function	***Description***
	from the speaker.
19	?
20	?
21	?
22	Enables/disables the noise that is added to the head to obtain a more realistic animation.
23	Enables/disables the noise that is added to the torso to obtain a more realistic animation.
24	Enables/disables the limitation of the wrists rotations to avoid unrealistic position to be reached.
25	Here you can choose which type of interpolation has to be used for the shoulders of Greta: euler – the euler angles in the shoulders are interpolated between them quat – the interpolation is made by passing to quaternion representation ik path – the interpolation is made on the path of the arm in the space and then the angles are obtained with inverse kinematics
26	Here you can choose which type of interpolation has to be used for the wrists of Greta: euler – the euler angles in the wrists are interpolated between them quat – the interpolation is made by passing to quaternion representation
27	Here you select how face expressions are generated. With "normal faceanim" you select the original Greta's engine which is recommended. With "2005 faceanim" you select another kind of reasoning which is used in some applications of Greta and should not be used by normal users.
28	OAC – This is an expressivity parameter only for gestures. It influences the quantity of gestures produced by Greta from the same APML file.
29	SPC – This is an expressivity parameter only for gestures. It changes the quantity of space used by Greta to produce gestures.
30	TMP – This is an expressivity parameter only for gestures. It changes the speed of gestures, negativa values correspond to slow gestures.
31	FLT – This is an expressivity parameter only for gestures. It changes the fluidity of gestures, negative values corrispond to jerky animation, positive to fluid animation.
32	PWR – This is an expressivity parameter only for gestures. It changes the energy in gestures. Low values will also produce beat gestures with an open hand shape. High values will produce beat gestures with fist hand shape.
33	REP – This is an expressivity parameter only for gestures. It influences the repetition of gestures. For example in the APML consecutive emphasis in the same performative will produce only one gesture on the first emphasis if REP is low or a beat on each emphasis if REP is high.
34	These are the initial dimensions of the Greta player.
35	The checkbox "preload agent" loads the Greta agent as the player starts. It is useful to maintain it always selected. The checkbox "only face" loads and shows only the head of Greta. In this way the execution (frame rate) of the player is faster.

Function	***Description***
36	Here is where you specify the APML file to be used as input for Greta.
37	With this checkbox you can visualize two faces in the player. When you load the fap file one face will load the speaker file and the other face will load the listener file.
38	With this button you start the animation generation only. That is the APML file is read and the fap and bap files are generated.
39	With this button you start the player. No calculation is performed. The player window appears and at this point you can use the player or the interface of Greta at any time.
40	With this button you quit Greta and all the settings in the interface are stored in the greta.ini file.

Files and Folders

File/Folder name	***Description***
%Greta_s_folder%/agents	it will contain defintion files for individualized agents. not used for the moment.
%Greta_s_folder%/bn	definition files for the belief networks used by the face engine. apml_face.net is used to solve facial conflicts (see function 1 of the interface). gaze.net is used to decide where speaker and listener have to llok when the active gaze model is the belief networks model (see function 12 of the interface).
%Greta_s_folder%/bodymodels	body definition files. not used for now.
%Greta_s_folder%/characters	characters definition files. each file contains the definition of the visual aspect of an agent as a combination of a head and a body model.
%Greta_s_folder%/coart	files that contain data that determine the mouth movement during speech. also see function 8 of the interface.
%Greta_s_folder%/curves	it is an output folder that contains data to visualize the curves of the lip movement, see function 9 of the interface.
%Greta_s_folder%/documentation	documentation files.
%Greta_s_folder%/environments	files describing environments that will be loaded in the Greta player. for now the player can load a static environment and displaying it.
%Greta_s_folder%/fd	facial expressions definition files. only facelibrary.xml is used by the engine.
%Greta_s_folder%/gestures	gesture definition files (see the section on the gesture engine for details).
%Greta_s_folder%/input	APML files to use as input for Greta.
%Greta_s_folder%/logs	log files.
%Greta_s_folder%/matlab	not used.
%Greta_s_folder%/output	FAP and BAP files generated by the engine. it also contains the associated WAV files and AVI files.
%Greta_s_folder%/poses	basic arms poses used by the gesture engine, see the section on the gesture engine for details.
%Greta_s_folder%/scores	scores definition files. they are used only by a particular application of Greta, called the AgentScore. not used by Greta.
%Greta_s_folder%/textures	bitmaps to use as texture maps for the environments.
%Greta_s_folder%/tmp	temporary files, mainly phoneme files.
%Greta_s_folder%/apml.dtd	the dtd of the APML language. you have to modify this

File/Folder name	***Description***
	file if you want to update the APML language.
%Greta_s_folder%/empty.bap	empy bap file loaded by the player at startup.
%Greta_s_folder%/empty.fap	empy fap file loaded by the player at startup.
%Greta_s_folder%/empty.wav	empy wap file loaded by the player at startup.
%Greta_s_folder%/face.jpg	an icon used in the AgentScore application.
%Greta_s_folder%/greta.ini	the Greta settings file. almost all the parameters are also set up by the graphical interface.
%Greta_s_folder%/mbrola.exe	it is used by Festival to synthesize speech.
%Greta_s_folder%/new.jpg	an icon used in the AgentScore application.

Face engine

The Greta's face engine creates the animation of the Greta's face, that is the facial expressions that appear on the Greta's face, the lip movement of Greta, the head movement caused by head nods/shakes and gaze, the eyes movements caused by gaze.

The face library

To generate facial expressions the Greta's face engine reads a static definition of these expressions from a library stored in the file %Greta_s_folder%/fd/facelibrary.xml.
For each expression the library contains a set of FAP values (that is a sequence of couples *FAP,value*) that if applied to face then cause a certain facial expression to appear. In the library the facial expression are defined using a hierarchical recursive definition. There are some basic actions that involve directly the movement of some FAPs. Then other higher level action use the basic action to create more complex action. The most high level actions are named as the tags and attributes of the APML language.
Here is an example, let's see how the facial expression *embarrassment* is defined:

```
<expression class="affect" instance="embarrassment">
<action name="embarrassment_mouth"/>
<action name="cheek_lift"/>
<action name="embarrassment_head_direction"/>
<action name="embarrassment_gaze"/>
<channel name="look_away"/>
<channel name="head_down"/>
<channel name="lip_corner_up"/>
</expression>
```

Then if we look up into the library file we can find the definitions of the actions that compose the *embarassement* expression:

```
<facelibrary>

<declaration>
...
<actiondeclaration name="embarrassment_mouth">
<fap num="3" value="-100"/>
<fap num="6" value="50"/>
<fap num="7" value="50"/>
<fap num="12" value="100"/>
<fap num="13" value="100"/>
<fap num="53" value="50"/>
<fap num="54" value="50"/>
<fap num="59" value="100"/>
<fap num="60" value="100"/>
</actiondeclaration>

<actiondeclaration name="cheek_lift">
<fap num="41" value="100"/>
<fap num="42" value="100"/>
</actiondeclaration>
```

```
<actiondeclaration name="embarrassment_head_direction">
<headdirection h="2" v="2" t="0"/>
</actiondeclaration>

<actiondeclaration name="embarrassment_gaze">
<eyesdirection h="0" v="4"/>
</actiondeclaration>

...

</declaration>
```

In this example there are some of the basic action (that is `embarrassment_mouth` and `cheek_lift`) that are defined by some FAPs movements and some (that is `embarrassment_head_direction` and `embarrassment_gaze`) that are defined in a completely different way. The action `embarrassment_head_direction` is defined by giving three angles that represent the rotations of the head around the horizontal, vertical and tilt axis. The action `embarrassment_gaze` is defined by giving two angles that represent the rotation of the eyes around the horizontal and vertical axis. There is another kind of definition of an action:

```
<actiondeclaration name="head_shake">
<headmovement type="shake" amplitude="8" period="1"/>
</actiondeclaration>
```

In this case there is not only a simple static defition but there is a movement definition. The first parameter (`head_m`) can have 2 possible values: *shake* and *nod*. The second one is the amplitude of the movement, in degrees. The third one is the period of the movement.

Moreover a facial expression can contain some information about the *channels* used by the expression:

```
<channel name="look_away"/>
<channel name="head_down"/>
<channel name="lip_corner_up"/>
```

Some facial expressions can use the same parts of the face and sometimes they could be displayed by Greta at the same time. In these cases the facial conflicts resolver will decide which expression should move some parts of the face and which expression should not. Here is an example:

```
<expression class="performative" instance="warn">
<action name="small_frown"/>
<action name="medium_close_upper_eyelids"/>
<action name="look_at"/>
<action name="head_down_left"/>
<channel name="frown"/>
<channel name="look_at"/>
</expression>

<expression class="affect" instance="surprise">
<action name="rising"/>
<action name="surprise_eyelids"/>
<action name="surprise_mouth"/>
<channel name="raising"/>
<channel name="tense"/>
</expression>
```

From this definition the conflict detection will find out that the expressions *performative=warn* and *affect=surprise* are both influencing the eyebrows but the first one will cause a frown and the second one a raising. These two movements cannot performed at the same time nor added together. So the conflict resolver will solve the conflict at the eyebrows level and decide if the eyebrows should be moved to do a frown or a raising.
Note that in this particular case the conflict will arise if the two expressions are performed simultaneously and this will happen if you give to Greta an APML file that looks like this:

```
<apml>
<performative type="warn">
Beauty without expression
<rheme affect="surprise">
is
<emphasis x-pitchaccent="Hstar"> really </emphasis>
<emphasis x-pitchaccent="Hstar"> boring </emphasis>
<boundary type="LL"/>
</rheme>
</performative>
</apml>
```

If you want to add a new expression you have to define it in one of the classes and specify by which basic action it is composed of. Then you will have to add some *channel* lines to allow conflict resolution on the new expression. Note that all the expressions corresponding to the actual definition of APML are already defined. So your new expression will need to match a new APML tag/attribute that you will need to add to the APML dtd.

Gesture engine

Gesture Definition

The gestures that can be performed by Greta are defined in some files inside the %Greta_s_folder%/gestures folder. Each file defines one gesture.

Gesture Definition File Format

[ ]	surround optional identifiers
< >	surround mandatory identifiers

Comments are defined by a double-forward slash at the beginning of a line (no preceding whitespace!)

```
// COMMENT LINE
```

A header is necessary for all gesture definition files. GESTURECLASS and GESTUREINSTANCE define the key under which the gesture is stored in the Gestuary database (as GESTURECLASS=GESTUREINSTANCE). DURATION is the default duration of this gesture in seconds. The optional keyword SIDE assigns a definite side to the gesture (useful mainly for gestures that require both arms.)

```
GESTURECLASS <Gesture class (string)>
GESTUERINSTANCE <Gesture instance (string)>
DURATION <Default duration in seconds>
[SIDE <SideType: left OR right OR both>]
```

Gesture frame blocks: define the position and shape of one (or both) arm and hand at a specific moment in time.

```
STARTFRAME <Relative Frame Time>
  [FRAMETYPE <GesturePhaseType>]
  [ARM <ArmX> <ArmY> <ArmZ>]
  [WRIST <PalmType> <FingerBaseType>]
  [HAND <FormType> [FormThumbType] [BendType]]             // OR:
  [HAND <SymbolType> [(SymbolOpenType OR SymbolClosedType)]]
  [FINGER <FingerType> <BendType>]                         // UP TO 4x
  [ADDNOISE]
ENDFRAME
```

Alternatively, a frame can be specified completely by a pose definition file (usually generated by write_angles.mel in Maya) loaded with the keyword STOREDPOSE:

```
STOREDPOSE <Relative File Name> <Relative Frame Time>
```

A sequence of Gesture frame blocks (not stored poses? – this has not been tested yet) can optionally be surrounded by the lines STARTLOOP and ENDLOOP. In this case the frames inside the loop will be repeated if the scheduled gesture duration is much longer than the default duration of the gesture (NOTE: this is not the same as repeating the stroke of a gesture for emphasis!)

```
STARTLOOP
  STARTFRAME…ENDFRAME
  STARTFRAME…ENDFRAME …
ENDLOOP
```

Hand Shapes

The specification terminology for hand shapes is borrowed from the Hamburg Notation System (see HamNoSys). The basic shape of the hand can be chosen from six basic forms, and six basic symbols (a hand shape is either a basic form or a basic symbol – it can never be both):

```
enum FormType     {form_default, form_fist, form_open, form_point1, form_point2,
                   form_2apart, form_openapart};
enum SymbolType   {symbol_default, symbol_1_open, symbol_1_closed,
                   symbol_2_open, symbol_2_closed, symbol_3_open,
                   symbol_3_closed};
```

Hands with basic type FormType can further be modified by specifying a thumb position and a bending modifier for the other four fingers (if no thumb position and/or no bending modifier is/are specified, the default position as shown in the diagram below will be used).

```
enum FormThumbType      {thumb_default, thumb_away, thumb_over};
enum BendType           {bend_default, bend_straight, bend_angled, bend_curved,
                         bend_closed};
```

Hands with basic type SymbolType can be modified in the following way: if the symbol is of the open variety, a thumb position defining the opening width can be specified; if the symbol is of the closed variety, a thumb position defining the type of connection between thumb and index finger can be specified.

```
enum SymbolOpenType     {open_default, open_thumbout, open_thumbin};
enum SymbolClosedType   {closed_default, closed_straight, closed_inside,
                         closed_tight};
```

In addition to the basic shape and the modifiers directly associated with it, all hand shapes can be further altered by adding separate finger specifications that override the positioning derived from the previous hand definition. Finger specifications can only be added for index, middle, ring, and pinky.

```
enum FingerType   {finger_default, thumb, index, middle, ring, pinky};
enum BendType     {bend_default, bend_straight, bend_angled, bend_curved,
                   bend_closed};
```

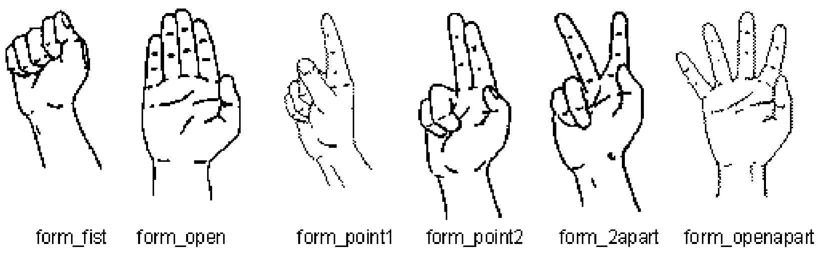
Thus, a valid hand shape definition within a gesture definition file is of the following forms:

```
HAND <FormType> [FormThumbType] [BendType]                  // OR:
HAND <SymbolType> [(SymbolOpenType OR SymbolClosedType)]    // PLUS:
[FINGER <FingerType> <BendType>]                            // UP TO 4x
```

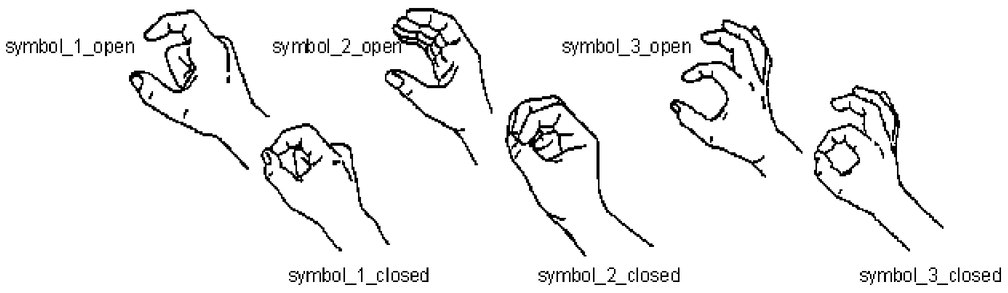
Some examples:

```
HAND form_open                                              // EXAMPLE 1

HAND symbol_3_closed closed_straight                        // EXAMPLE 2

HAND symbol_1_open                                          // EXAMPLE 3
FINGER index bend_curved
```

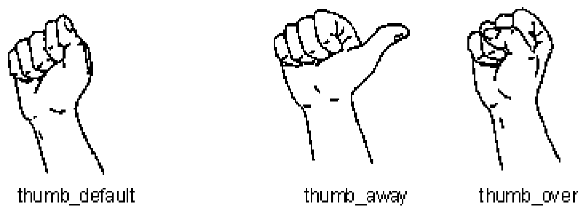
Hand shapes 6 basic forms:



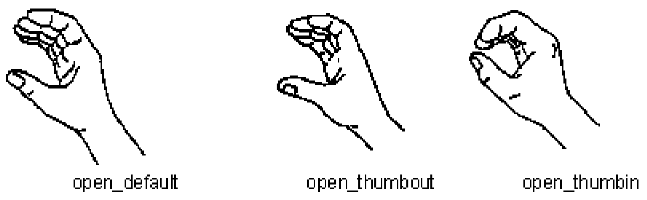
Hand shapes 6 basic symbols:



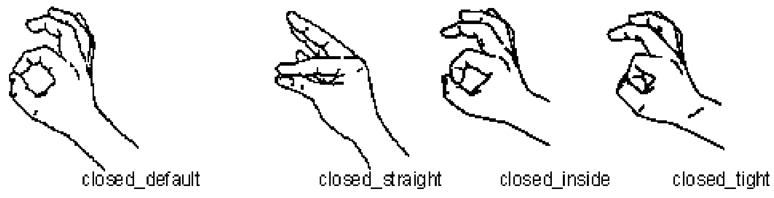
Thumb modifiers for hand shapes of type "form_*":



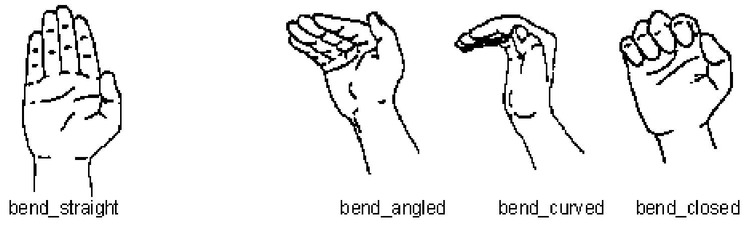
Thumb modifiers for hand shapes of type "symbol_*_open":



Thumb modifiers for hand shapes of type "symbol_*_closed":



bend modifiers for individual fingers (or all 4 fingers for "form_*"):



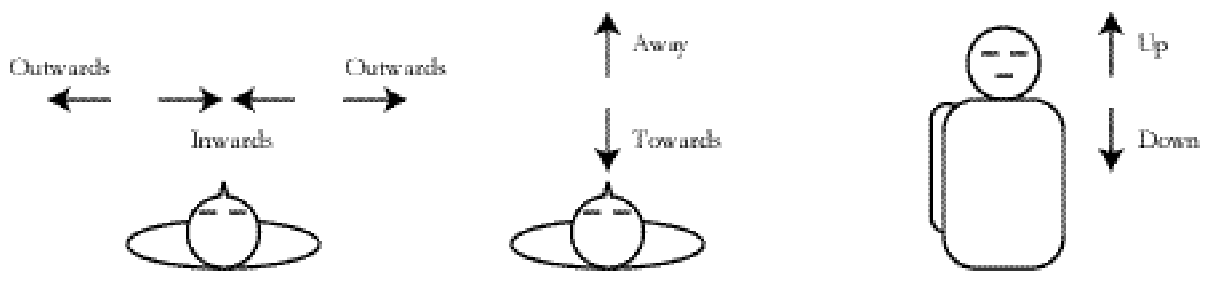
Wrist Orientation

Wrist orientation is specified as a combination between two orthogonal vectors: the FingerBase vector and the PalmNormal vector. Because of orthogonality, not all combinations of these types make sense. In the case that an impossible combination of PalmType and FingerBaseType is read in a gesture definition file, a warning message is output and the line is ignored.

```
enum PalmType           {PalmDefault, PalmUp,PalmDown, PalmAway,PalmTowards,
                         PalmInwards,PalmOutwards,PalmNone};
enum FingerBaseType     {FBDefault, FBUp, FBDown, FBAway, FBTowards, FBInwards,
                         FBOutwards,FBNone};
```

A valid wrist orientation definition within a gesture definition file is of the following form:
```
WRIST <PalmType> <FingerBaseType>
```

Example:
```
WRIST FBUp PalmInwards                              // EXAMPLE
```



Arm Position

Abstract arm positions (thus called since they operate on the angles of the arm, although they specify the location of the wrist in space) are specified using the following enumeration types for their relative X,Y, and Z coordinates:

```
enum ArmX {XEP=0, XP, XC, XCC, XOppC, XDefault};
enum ArmY {YUpperEP=0, YUpperP, YUpperC, YCC, YLowerC, YLowerP, YlowerEP,
           YDefault};
enum ArmZ {ZNear=0, ZMiddle, ZFar, ZDefault};
```

Thus, a valid hand shape definition within a gesture definition file is of the following forms:

```
ARM <ArmX> <ArmY> <ArmZ>
```

No axis type may be omitted – the definition always needs a complete (x,y,z) value triple. The terms are derived from McNeill's definition of the human gesture space (1992, p. 89ff). Here is a key to the abbreviations I used:

- C = Center
- CC = Center-Center
- P = Periphery
- EP = Extreme Periphery
- Opp = Opposite Side

NOTES: During MotorPlanner::Init(), the function ArmPosition::Concretize() is called. This functions transforms the (ArmX,ArmY,ArmZ) coordinates into a second abstract set of coordinates. For each possible (X,Y,Z) combination of these secondary coordinates, a **Pose Definition File** (currently in %Greta_s_folder%/poses/armpositions) exists in which the Euler-angles for shoulder and elbow that move the wrist to the specified location are stored. The function GetAngles() thus simply reads the Pose Definition File that corresponds to the desired (X,Y,Z) position and writes all angles contained in the file to the current BAPframe (using BAPframe::ReadPose()).

There is a slight problem with the secondary coordinate system. Namely, no Y coordinate corresponds to ArmY::YLowerExtremePeriphery. A workaround was found for the long resting position (one single pose was defined), but I strongly advise against using YLowerEP in Gesture Definition Files for now.

Greta's tools

FaceLibraryViewer

One can create facial expressions interactively by moving one FAP at a time. Eyebrow Action Unit (from FACS) have been inserted into the tools. You need to save the expression in either the FAP or the library XML format. When saved in the library XML format you need to report the description given for the Face Engine.

Please refer to "documentation\ReadMe - FaceLibraryViewer.txt".

You can view different facial expressions that have been created for each APML tags. To each tag can be associated several expressions with their probability of appearance. Please refer to our paper to get more information of facial expression language.

B. de Carolis, C. Pelachaud, I. Poggi, M. Steedman,``APML, a Mark-up Language for Believable Behavior Generation'', in H. Prendinger, Ed, Life-like Characters. Tools, Affective Functions and Applications, Springer, 2004.

FacePartsTool

This tools allows you to place a head onto the Greta body, to move the eyes into the eye socket and the teeth into the mouth. These new eyes and teeth positions are saved in the character-name.xml file in the characters folder.

GestureEditor

Through this interface you can create gestures. The gestures are specified in a symbolic language described in:

B. Hartmann, M. Mancini, C. Pelachaud, Formational parameters and adaptive prototype instantiation for MPEG-4
compliant gesture synthesis, Computer Animation, Geneva, june 2002.

(see also the section on the Gesture Engine).

Once you have created a gesture you can visualize it. To do so, you need to save the gesture and load it back. Then by selecting 'test' the gesture, the greta player window will play the gesture. At each change of the gesture specification, you need to save it and load it back before being able to visualize it. When creating a gesture you need to specify:
- GESTURE CLASS: that should be selected within the APML function tags (emotion, certainty, performative, ...)
- INSTANCE: a name for the particular gesture. Be aware that it is this information you will be using in an APML file.
The names should not be too long (less than 32 characters).

When testing the gesture, you can select different expressivity values. You will see how the expressivity parameters affect the gesture. If a gesture parameter should not be modified by the expressivity parameters, you should select the 'fixed' box.

PoseEditor

While GestureEditor allows you to create gesture using a representation language, PoseEditor allows you to create a gesture by moving one joint at a time. You can also use this tool to refine a gesture that has been created with GestureEditor. If you create a gesture using solely PoseEditor, you need to run GestureEditor to get the proper format. Within GestureEditor, simply provide the gesture class and instance. See the 'greet' gesture as described in 'greet.txt' for example.