BEAT: the Behavior Expression Animation Toolkit

Justine Cassell, Hannes Vilhjálmsson, Timothy Bickmore
MIT Media Lab 20 Ames St., E15-315
Cambridge, MA 02139 USA
+1 617 253 4899
{justine, hannes, bickmore}@media.mit.edu

ABSTRACT

The Behavior Expression Animation Toolkit (BEAT) allows animators to input typed text that they wish to be spoken by an animated human figure, and to obtain as output appropriate and synchronized nonverbal behaviors and synthesized speech in a form that can be sent to a number of different animation systems. The nonverbal behaviors are assigned on the basis of actual linguistic and contextual analysis of the typed text, relying on rules derived from extensive research into human conversational behavior. The toolkit is extensible, so that new rules can be quickly added. It is designed to plug into larger systems that may also assign personality profiles, motion characteristics, scene constraints, or the animation styles of particular animators.

Keywords

Animation Systems, Facial Animation, Speech Synthesis

1. INTRODUCTION

The association between speech and other communicative behaviors causes particular challenges to procedural character animation techniques. Increasing numbers of procedural animation systems are capable of generating extremely realistic movement, hand gestures, and facial expressions in silent characters. However, when voice is called for, the issues of synchronization and appropriateness render disfluent otherwise more than adequate techniques. And yet there are many cases where we may want to animate a speaking character. Cartoon political rallies or cocktail party scenes, for example, demand a crowd of speaking and gesturing virtual actors. While spontaneous gesturing and facial movement occurs naturally and effortlessly in our daily conversational activity, when forced to think about such associations between nonverbal behaviors and words in explicit terms a trained eye is called for. For example, untrained animators, and autonomous animated interfaces, often generate a pointing gesture towards the listener when a speaking character says "you". ("If you want to come with me, get your coat on"). A point of this sort, however, never occurs in life (try it yourself and you will see that only if "you" is being contrasted with somebody else might a pointing gesture occur) and, what is much worse, makes an animated speaking character seem stilted, as if speaking a language not her own. In fact, for this reason, many animators rely on video footage of actors reciting the text, for reference or rotoscoping, or more recently, rely on motion captured data to drive speaking characters. These are expensive methods that may involve a whole crew of people in addition to the expert animator. This may be worth doing for characters that play a central role on the screen, but is not as justified for a crowd of extras.

In some cases, we may not even have the opportunity to handcraft or capture the animation. Embodied conversational agents as interfaces to web content, animated non-player characters in interactive role playing games, and animated avatars in online chat environments all demand some kind of procedural animation. Although we may have access to a database of all the phrases a character can utter, we do not necessarily know in what context the words may end up being said and may therefore not be able to link the speech to appropriate context sensitive nonverbal behaviors beforehand.

BEAT allows one to animate a human-like body using just text as input. It uses linguistic and contextual information contained in the text to control the movements of the hands, arms and face, and the intonation of the voice. The mapping from text to facial, intonational and body gestures is contained in a set of rules derived from the state of the art in nonverbal conversational behavior research. Importantly, the system is extremely permeable, allowing animators to insert rules of their own concerning personality, movement characteristics, and other features that are realized in the final animation. Thus, in the same way as Text-to-Speech (TTS) systems realize written text in spoken language, BEAT realizes written text in embodied expressive behaviors. And, in the same way as TTS systems are permeable to trained users, allowing them to tweak intonation, pause-length and other speech parameters, BEAT is permeable to animators, allowing them to write particular gestures, define new behaviors and tweak the features of movement.

The next section gives some background to the motivation for BEAT. Section 3 describes related work. Section 4 walks the reader through the implemented system, including explaining the methodology of text annotation, selection of nonverbal behaviors, and synchronization. An extended example is covered in Section

5. Section 6 presents our conclusions and describes possible directions for future work.

2. CONVERSATIONAL BEHAVIOR

To communicate with one another, we use words, of course, but we also rely on intonation (the melody of language), hand gestures (beats, iconics, pointing gestures [23]), facial displays (lip shapes, eyebrow raises), eye gaze, head movements and body posture. The form of each of these modalities – a rising tone vs. a falling tone, pointing towards oneself vs. pointing towards the other – is essential to the meaning. But the co-occurrence of behaviors is equally important. There is a tight synchrony among the different communicative modalities in humans. Speakers accentuate only the important words by speaking more forcefully, gesture along with the word that a gesture illustrates, and turn their eyes towards the listener when coming to the end of a thought. Meanwhile listeners nod within a few hundred milliseconds of when the speaker's gaze shifts. This synchrony is essential to the meaning of conversation. Speakers will go to great lengths to maintain it (stutterers will repeat a gesture over and over again, until they manage to utter the accompanying speech correctly) and listeners take synchrony into account in what they understand. (Readers can contrast "this is a **stellar** siggraph submission" [big head nod along with "stellar"] with "this is a **. . .** stellar siggraph submission" [big head nod during the silence]). When synchrony among different communicative modalities is destroyed, as in low bandwidth videoconferencing, satisfaction and trust in the outcome of a conversation is diminished. When synchrony among different communicative modalities is maintained, as when one manages to nod at all the right places during the Macedonian policeman's directions, despite understanding not a word, conversation comes across as successful.

Although all of these communicative behaviors work together to convey meaning, the communicative intention and the timing of all of them are based on the most essential communicative activity, which is speech. The same behaviors, in fact, have quite different meanings, depending on whether they occur along with spoken language or not, and similar meanings are expressed quite differently when language is or is not a part of the mix. Indeed, researchers found that when people tried to tell a story without words, their gestures demonstrated entirely different shape and meaning characteristics – in essence, they began to resemble American Sign Language – as compared to when the gestures accompanied speech [23].

Skilled animators have always had an intuitive grasp of the form of the different communicative behaviors, and the synchrony among them. Even animators, however, often turn to rotoscoping or motion capture in cases where the intimate portrayal of communication is of the essence.

3. RELATED WORK

Until the mid-1980s or so, animators had to manually enter the phonetic script that would result in lip-synching of a facial model to speech (c.f. [26]). Today we take for granted the ability of a system to automatically extract (more or less beautiful) "visemes" from typed text, in order to synchronize lip shapes to synthesized or recorded speech [33]. We are even able to animate a synthetic face using voice input [6] or to re-animate actual videos of human faces, in accordance with recorded audio [7]. [27] go further in the direction of communicative action and generate not just visemes, but also syntactic and semantic facial movements. And the gains are considerable, as "talking heads" with high-quality lip-synching significantly improve the comprehensibility of synthesized speech [22], and the willingness of humans to interact with synthesized speech [25], as well as decrease the need for animators to spend time on these time-consuming and thankless tasks.

Animators also spend an enormous amount of effort on the thankless task of synchronizing body movements to speech, either by intuition, or by using rotoscoping or motion capture. And yet, we still have seen no attempts to automatically specify "gestemes" on the basis of text or to automatically synchronize ("body-synch") those body and face behaviors to synthesized or recorded speech. The task is a natural next step, after the significant existent work that renders communication-like human motion realistic in the absence of speech, or along with text balloons. Researchers have concentrated both on low-level features of movement, and aspects of humans such as intentionality, emotion, and personality. [5] devised a method of interpolating and modifying existing motions to display different expressions. [14] have concentrated on providing a tool for controlling the expressive shape and effort characteristics of gestures. Taking existing gestures as input, their system can change the nature of how a gesture is perceived. [1] have concentrated on realistic emotional expression of the body. [4] and [3] have developed behavioral animation systems to generate animations of multiple creatures with varying personalities and/or intentionality. [8] constructed a system that portrays the gestural interaction between two agents as they pass and greet one another, and in which behavioral parameters were set by personality attribute "sliders." [29] concentrated on the challenge of representing the personality of a synthetic human in how it interacted with real humans, and the specification of coordinated body actions using layers of motions defined relative to a set of periodic signals.

There have also been a smaller number of attempts to synthesize human behaviors specifically in the context of communicative acts. [20] implemented a graphical chat environment that automatically generates still poses in comic book format on the basis of typed text. This very successful system relies on conventions often used in chat room conversations (chat acronyms, emoticons) rather than relying on the linguistic and contextual features of the text itself. And the output of the system depends on our understanding of comic book conventions – as the authors themselves say "characters pointing and waving, which occur relatively infrequently in real life, come off well in comics."

Synthesis of animated communicative behavior has started from an underlying computation-heavy "intention to communicate" [10], a set of natural language instructions [2], or a state machine specifying whether or not the avatar or human participant was speaking, and the direction of the human participant's gaze [15]. However, starting from an intention to communicate is too computation-heavy, and requires the presence of a linguist on staff. Natural language instructions guide the synthetic human's actions, but not its speech. And, while the state of speech is essential, the content of speech must also be addressed in the assignment of nonverbal behaviors.

In the current paper, we describe a toolkit that automatically suggests appropriate gestures, communicative facial expressions, pauses, and intonational contours for an input text, and also provides the synchronization information required to animate the behaviors in conjunction with a character's speech. This layer of analysis is designed to bridge the gap between systems that specify more natural or more expressive movement contours (such as [14], or [28] and systems that suggest personality or emotional realms of expression (such as [3] or [29]).

4. SYSTEM

The BEAT system is built to be modular and user extensible, and to operate in real-time. To this end, it is written in Java, is based on an input-to-output pipeline approach with support for user defined filters and knowledge bases, and uses an XML tagging scheme. Processing is decomposed into modules which operate as XML transducers; each taking tagged text as input and producing tagged text as output. XML provides a natural way to represent information which spans intervals of text, and its use facilitates modularity and extensibility. Each module operates by reading in XML-tagged text (initially representing the text of the character's script only), converting it into a parse tree, manipulating the tree, then re-serializing the tree into XML before passing it to the next module. The various knowledge bases used in the system are also encoded in XML so that they can be easily extended for new applications.

An overview of the system is shown in Figure 1. There are three main processing modules: Language Tagging module, Behavior Generation module and Behavior scheduling module. The stages of XML translation produced by each of these modules are shown in Figure 2. The Behavior Generation module is further divided into a Suggestion module and a Selection module as our approach to the generation process is to first suggest all plausible behaviors and then use user modifiable filters to trim them down to a set appropriate for a particular character. In Figure 1, user definable data structures are indicated with dotted line boxes. We will now discuss each of these components in turn.

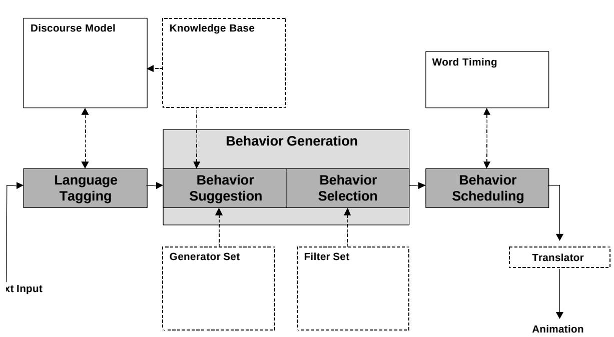

Figure 1. BEAT System Architecture

4.1 Knowledge Base

A knowledge base adds some basic knowledge about the world to what we can understand from the text itself, and therefore allows us to draw inferences from the typed text, and consequently specify the kinds of gestures that should illustrate it, and the kinds of places where emphasis should be placed. Currently, the knowledge base is stored in two XML files, one describing objects and other describing actions. These knowledge bases are seeded with descriptions of generic objects and actions but can easily be extended for particular domains to increase the efficacy of nonverbal behavior assignment.

The object knowledge base contains definitions of classes and instances of objects. Figure 3 shows two example entries. The first defines a new object class *CHARACTER* as a type of person (vs. object or place) with two features: *TYPE*, describing whether the professional is *REAL* or *VIRTUAL*; and *ROLE*, describing the actual profession. Each feature value is also described as being "normal" or "unusual" (e.g., a virtual person would be considered unusual), which is important since people tend to generate iconic gestures for the unusual aspects of objects they describe [34]. Each feature value can also provide a gesture specification which describes the type of hand gesture that should be used to depict it (as described below). The second knowledge base entry defines an object instance and provides values for each feature defined for the class.

The action knowledge base contains associations between domain actions and hand gestures which can depict them. An

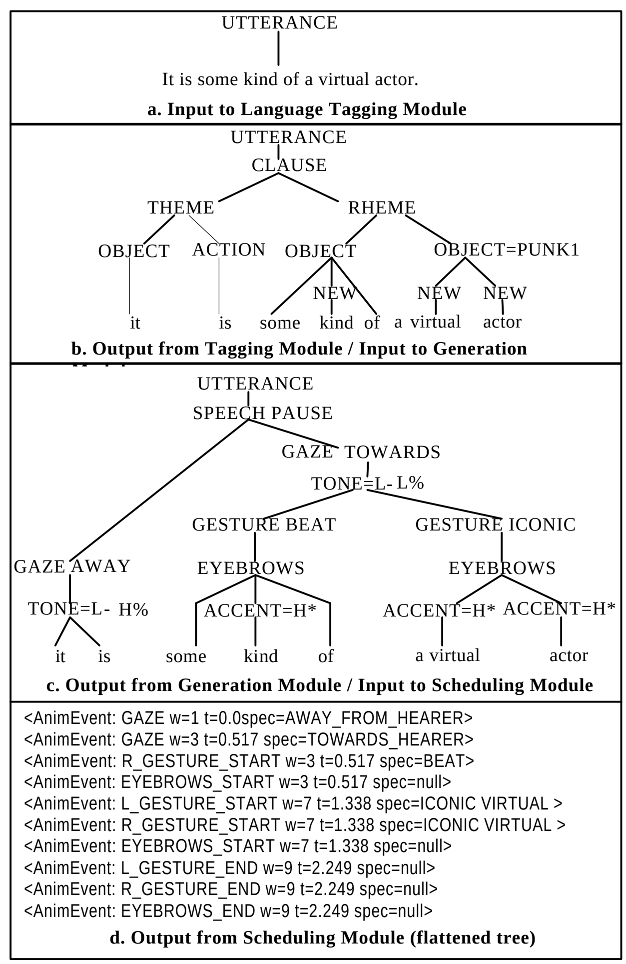

Figure 2. XML Trees Passed Among Modules

example entry is

<ACTION NAME="MOVE" GESTURE="R hand=5, moves from CC towards L …">

which simply associates a particular gesture specification with the verb *to move*.

As mentioned above, the system comes loaded with a generic knowledge base, containing information about some objects and actions, and some common gestures. Gestures are specified using a compositional notation in which hand shapes and arm trajectories for each arm are specified independently. This makes the addition of new gestures easier, since existing trajectories or hand shapes can be re-used.

4.2 Language Tagging

The language module of the Toolbox is responsible for annotating input text with the linguistic and contextual information that allows successful nonverbal behavior assignment and scheduling. The toolkit was constructed so that animators need not concern themselves with linguistic analysis. However, in what follows we briefly describe the few essential fundamental units of analysis used in the system. The language module automatically recognizes and tags each of these units in the text typed by the user. It should be noted that much of what is described in this section is similar to or, in some places identical, to the kind of tagging that allows TTS systems to produce appropriate intonational contours and phrasing along with typed text [17]. Additional annotations are used here, however, to allow not just intonation but also facial display and hand gestures to be generated. And, these annotations will allow not just generation, but also synchronization and scheduling of multiple nonverbal communicative behaviors with speech.

The largest unit is the UTTERANCE, which is operationalized as an entire paragraph of input. The utterance is broken up into CLAUSEs, each of which is held to represent a proposition. To detect clause boundaries the tagging module looks for punctuation and the placement of verb phrases.

```
<CLASS NAME="CHARACTER" ISA="PERSON">

<FEATURE NAME="TYPE">
<VALUEDESC NAME="REAL" ISNORMAL="TRUE">
<VALUEDESC NAME="VIRTUAL" ISNORMAL="FALSE"
  GESTURE="gesture specification goes here">

<FEATURE NAME="ROLE">
<VALUEDESC NAME="ACTOR" ISNORMAL="TRUE">
<VALUEDESC NAME="ANIMATOR" ISNORMAL="TRUE">

<INSTANCE NAME="PUNK1">
<VALUE FEATURE="ROLE" VALUE="ACTOR">
<VALUE FEATURE="TYPE" VALUE="VIRTUAL">

</CLASS>
```

Figure 3. Example Object Knowledge Base

Clauses are further divided into two smaller units of information structure, a THEME and a RHEME. The former represents the part of the clause that creates a coherent link with a preceding clause and the latter is the part that contributes some new information to the discussion [16]. For example in the mini-dialogue "who is he?" "he is a student", the "he is" part of the second clause is that clause's theme and "student" is the rheme. Identifying the rheme is especially important in the current context since gestural activity is usually found within the rheme of an utterance [9]. The language module uses the location of verb phrases within a clause and information about which words have been seen before in previous clauses to assign information structure, following the heuristics described in [18].

The next to smallest unit is the word phrase, which in the current implementation either describes an ACTION or an OBJECT. These two correspond to the grammatical verb phrase and noun phrase, respectively. Actions and objects are linked to entries in the knowledge base whenever possible, as follows. For actions, the language module uses the verb head of the corresponding verb phrase as the key to look up an action description in the action database. If an exact match for that verb is not found, it is sent to an embedded word ontology module (using WordNet [24]), which creates a set of hypernyms and those are again used to find matching descriptions in the knowledge base. A hypernym of a word is a related, but a more generic -- or broader -- term. In the case of verbs, one can say that a certain verb is a specific way of accomplishing the hypernym of that verb. For example "walking" is a way of "moving", so the latter is a hypernym of the former. Expanding the search for an action in the action database using hypernyms makes it possible to find and use any descriptions that may be available for a super-class of that action. The database therefore doesn't have to describe all possible actions, but can focus on high-level action categories. When an action description match is found, a description identifier is added to the ACTION tag.

For objects, the module uses the noun head as well as any accompanying adjectives to find a unique instance of that object in the object database. If it finds a matching instance, it adds the unique identifier of that instance to the OBJECT tag.

The smallest units that the language module handles are the words themselves. The tagger uses the EngLite parser from Conexor (www.conexor.fi to supply word categories and lemmas for each word. It also keeps track of all previously mentioned words and marks each incoming noun, verb, adverb or adjective as NEW if it has not been seen before. This "word newness" helps to determine which words should be emphasized by the addition of intonation, eyebrow motion or hand gesture [18].

Words can also stand in contrast to other words (for example "I went to buy **red** apples but all they had were **green** ones"), a property often marked with hand gesture and intonation and therefore important to label. The language module currently labels contrasting adjectives by using WordNet to supply information about which words might be synonyms and which might be antonyms to one another [18]. Each word in a contrast pair is tagged with the CONTRAST tag.

In sum, the language tags that are currently implemented are:

- Clause
- Theme and rheme
- Word newness

- Contrast
- Objects and actions

4.3 Behavior Suggestion

The Behavior Suggestion module operates on the XML trees produced by the Language Tagging module (such as the one shown in Figure 2b) by augmenting them with suggestions for appropriate nonverbal behavior. This augmentation is intended to be liberal and all-inclusive; any nonverbal behavior that is possibly appropriate is suggested independent of any other. The resulting over-generated behaviors will be filtered down in the next stage of processing to the final set to be animated. This independence of behavior suggestions allows filters to be defined for different personality types, situations, and scenes.

Behavior suggestion proceeds by applying each of an extensible set of nonverbal behavior generators to all nodes in the XML tree which meet criteria specified by each generator. When the criteria are completely satisfied a suggestion is added to the appropriate node. The pseudocode for the generator which suggests beat gestures is shown in Figure 4 (behavior generators are actually implemented in Java).

```
FOR each RHEME node in the tree
  IF the RHEME node contains at least
        one NEW node
  THEN Suggest a BEAT to coincide
        with the OBJECT phrase
```

Figure 4. Example Behavior Generator

This pseudocode states that beat gestures are appropriate during the description of objects (noun phrases), but only when those objects are part of the rheme (new information) and contain new words.

Behavior suggestions are specified with a tree node (defining the time interval they are active for), priority (used for conflict resolution), required animation degrees-of-freedom, and any specific information needed to render them (e.g., gesture specification). Suggestions also specify whether they can *co-articulate*, i.e., occur during other behaviors which use the same degrees of freedom. For example, beat gestures can co-articulate with other gestures through the addition of a relative hand displacement [10].

The current set of behavior generators implemented in the toolkit includes the following:

4.3.1 Beat GestureGenerator

Beats, or formless handwaves, are a "default" gesture, in that they are used when no additional form information is available to generate a more specific kind of gesture, and they account for roughly 50% of the naturally occuring gestures observed in most contexts [23]. Thus, they are typically redundantly generated when other types of gestures are appropriate, but they are given a low priority relative to other types of gestures so that they will only be selected when no other gestures are available. Like all gestures that occur during speech, beats occur primarily during the introduction of new material (rheme).

4.3.2 Surprising Feature Iconic Gesture Generator

A study of individuals describing house floor plans showed that gestures representing some feature not described in accompanying speech were used 80% of the time during the description of house features which were "surprising" or unusual in some way, [34]. Following these results, this generator determines if any of the OBJECTS identified by the Tagger within the RHEME have unusual features (based on information in the object knowledge base), and for each generates an iconic (representational) gesture based on the gesture specification defined on the unusual feature value in the knowledge base.

4.3.3 Action Iconic Gesture Generator

This generator determines if there are any actions (verb phrase roots) occurring within the RHEME for which gestural descriptions are available in the action knowledge base. For each such action, an iconic gesture is suggested with the gesture specification used from the knowledge base.

4.3.4 Contrast Gesture Generator

The tagger identifies objects which contrast with other nearby objects (e.g., "Are you *a good witch* or *a bad witch*?"). Such objects (even if they occur within a THEME) are typically marked with either beats or a "contrastive gesture" if there are exactly two such objects being contrasted (gestures literally of the form "on the one hand…on the other hand") [11]. This generator suggests beats for contrast items unless there are exactly two items being contrasted, in which case the special contrast gesture is suggested.

4.3.5 Eyebrow Flash Generator

Raising of eyebrows can also be used to signal the introduction of new material [27]. This generator suggests raising the character's eyebrows during the description of OBJECTs within the RHEME.

4.3.6 Gaze Generator

[12] studied the relationship between eye gaze, theme/rheme, and turn-taking, and used these results to define an algorithm for controlling the gaze behavior of a conversational character. The gaze generator implements this algorithm is shown in Figure 5.

```
FOR each THEME
  IF at beginning of utterance OR
                         70% of the time
    Suggest Gazing AWAY from user
FOR each RHEME
  If at end of utterance OR 73% of the time
    Suggest Gazing TOWARDS the user
```

Figure 5. Algorithm for controlling conversational gaze

4.3.7 Intonation Generator

The intonation generator implements three different strategies for controlling a Text-To-Speech (TTS) engine. The first strategy assigns accents and boundary tones based on a theme-rheme analysis, as described by [30] and shown in Figure 6.

```
Within THEME:
   Suggest L+H* accent for NEW objects
   Suggest LH% boundary tone at end of THEME
Within RHEME:
   Suggest H* accent on NEW objects
   Suggest LL% boundary tone at end of RHEME
```

Figure 6. Algorithm for accent and boundary tone generation

The second intonation strategy suggests H* accents for all CONTRAST objects identified by the Tagger, following [30]. The final intonation strategy simply suggests TTS pauses at CLAUSE boundaries.

4.4 Behavior Selection

The Behavior Selection module analyzes the tree that now contains many, potentially incompatible, gesture suggestions, and reduces these suggestions down to the set that will actually be used in the animation. The selection process utilizes an extensible set of filters which are applied to the tree in turn, each of which can delete behavior suggestions which do not meet its criteria. In general, filters can reflect the personalities, affective state and energy level of characters by regulating how much nonverbal behavior they exhibit. Currently, two filter strategies are implemented: conflict resolution and priority threshold.

4.4.1 Conflict Resolution Filter

The conflict resolution filter detects all nonverbal behavior suggestion conflicts (those which physically cannot co-occur) and resolves the conflicts by deleting the suggestions with lower priorities. Conflicts are detected by determining, for each animation degree-of-freedom, the suggestions which co-occur and require that degree-of-freedom, even if specified at different levels of the XML tree. For each pair of such conflicting suggestions (in decreasing order of priority) the one with lower priority is deleted *unless* the two can be co-articulated (e.g., a beat gesture on top of an iconic gesture).

4.4.2 Priority Threshold Filter

The priority threshold filter simply removes all behavior suggestions whose priority falls below a user-specified threshold.

4.5 Behavior Scheduling and Animation

The last module in the XML pipeline converts its input tree into a set of instructions which can be executed by an animation system, or edited by an animator prior to rendering. In general, there are two ways to achieve synchronization between a character animation subsystem and a subsystem for producing the character's speech (either through a TTS engine or from recorded audio samples). The first is to obtain estimates of word and phoneme timings and construct an animation schedule prior to execution (see Figure 7). The second approach is to assume the availability of real-time events from a TTS engine--generated while the TTS is actually producing audio--and compile a set of event-triggered rules to govern the generation of the nonverbal behavior. The first approach must be used for recorded-audio-based animation or TTS engines such as Festival [32], while the second must be used with TTS engines such as Microsoft's Whistler [19]. We have used both approaches in our systems, and the current toolkit is capable of producing both kinds of animation schedules, but we will focus our discussion here on absolute-time-based scheduling with a TTS engine such as Festival.

The first step in time-based scheduling is to extract only the text and intonation commands from the XML tree, translate these into a format for the TTS engine, and issue a request for word and phoneme timings. In our implementation, the TTS runs as a separate process. Thus part of the scheduling can continue while these timings are being computed.

The next step in the scheduling process is to extract all of the (non-intonation) nonverbal behavior suggestions from the tree, translate them into an intermediate form of animation command, and order them by word index into a linear animation proto-schedule.

Once the word and phoneme timings become available, the proto-schedule can be instantiated by mapping the word indices into execution times (relative to the start of the schedule). The schedule can then also be augmented with facial animation commands to lip-sync the phonemes returned from the TTS engine. Figure 8. shows a fragment of an animation schedule at this stage of compilation.

```
<VISEME  time=0.0 spec="A">
<GAZE word=1 time=0.0 spec=AWAY_FROM_HEARER>
<VISEME  time=0.24 spec="E">
<VISEME  time=0.314 spec="A">
<VISEME  time=0.364 spec="TH">
<VISEME  time=0.453 spec="E">
<GAZE word=3 time=0.517 spec=TOWARDS_HEARER>
<R_GESTURE_START word=3 time=0.517 spec=BEAT>
<EYEBROWS_START word=3 time=0.517>
```

Figure 8. Example Abstract Animation Schedule Fragment

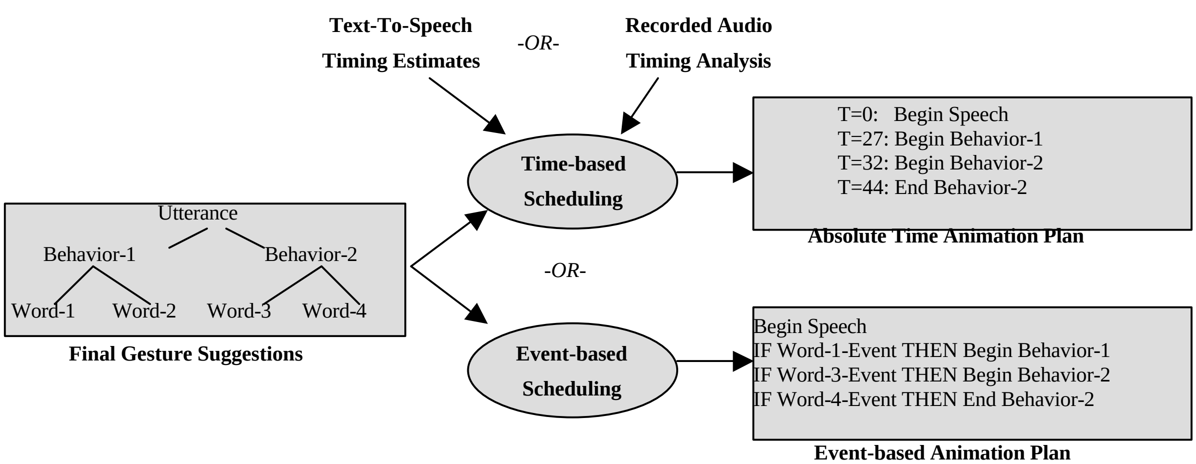

Figure 7. Scheduling Process

The final stage of scheduling involves compiling the abstract animation schedule into a set of legal commands for whichever animation subsystem is being used. This final compilation step has also been modularized in the toolkit. In addition to simply translating commands it must concern itself with issues such as enabling, initializing and disabling different animation subsystem features, gesture approach, duration and relax times (the abstract schedule specifies only the peak time at start of phrase and the end of phrase relax time), and any time offsets between the speech production and animation subsystems.

Our current compilation target is a humanoid animation system we have developed called Pantomime [13]. Pantomime animates one or more VRML-defined characters (adhering to the H-ANIM standard [31]) using a variety of motor skill modules, and resolves any remaining conflicts in character degrees-of-freedom. Pantomime can receive an animation schedule for the character, with the schedules specifying motor skills to be executed at specific times relative to the start of the schedule. Hand and arm commands are treated specially, however, in that complete motions for each hand and arm are computed prior to the start of the schedule. As a result, motions through all specified keyframe positions can be spline-smoothed for more natural looking behavior. Overlayed onto all commanded motion is a tailorable amount of Perlin noise on each character joint [28], and idle motor skills (such as eye blinking) to provide a more life-like character. Pantomime renders the final set of character joint angles using OpenInventor.

4.6 EXTENSIBILITY

As described in the introduction, BEAT has been designed to fit into a number of existent animation systems, or to exist as a layer between lower-level expressive features of motion and higher-level specification of personality or emotion. It has also been designed to be extensible in several significant ways. First, new entries can easily be made in the knowledge base to add new hand gestures to correspond to domain object features and actions. Second, the range of nonverbal behaviors, and the strategies for generating them, can easily be modified by defining new behavior suggestion generators. Behavior suggestion filters can also be tailored to the behavior of a particular character in a particular situation, or to a particular animator's style. Animation module compilers can be swapped in for different target animation subsystems. Finally, entire modules can be easily re-implemented (for example, as new techniques for text analysis become available) simply by adhering to the XML interfaces.

One additional kind of flexibility to the system derives from the ability to override the output from any of the modules simply by including appropriate tags in the original text input. For example, an animator could force a character to raise its eyebrows on a particular word simply by including the relevant EYEBROWS tag wrapped around the word in question, and this tag will be passed through the Tagger, Generation and Selection modules and compiled into the appropriate animation commands by the Scheduler.

5. EXAMPLE ANIMATION

To demonstrate how the system works, in this section we walk through a couple of example utterances. The full animated example can be found on the accompanying video tape.

As a first example, we trace what happens when BEAT receives as input the two subsequent sentences "It is some kind of a virtual actor" and "You just have to type in some text, and the actor is able to talk and move by itself". Lets look at each sentence in turn.

The language tagging module processes the input first, and generates an XML tree, tagged with relevant language information as described in section 4.1. The output of the language tagger is shown in Figure 2b. Of particular interest in Sentence 1 is the classification of "a virtual actor" as an object and the ability of the system to give it the unique identifier PUNK1. This is because when looking for the object in the knowledge base, it found under a user-defined type CHARACTER, an instance of an ACTOR that in fact is of the virtual type, this was the only instance matching on this attribute, so the instance name PUNK1 was copied into the value of ID in the object tag.

When the behavior generator receives the XML tree from the language tagger, it applies generator rules to annotate the tree with appropriate behaviors as described in section 4.3. Beats are suggested for the object "some kind of" and the object "a virtual actor" (previously identified as PUNK1) because these objects are inside a rheme and contain new words. Eyebrow raising is also suggested for these same objects and intonational accents are suggested for all the new lexical items (words) contained in those two objects (i.e. "kind", "virtual" and "actor"). Eye gaze behavior and intonational boundary tones are suggested based on the division into theme and rheme. Of particular interest is the suggestion for an iconic gesture to accompany PUNK1. This suggestion was generated because, upon examining the database entry for PUNK1, the generator found that one of its attributes, namely the type, did not hold a value within a typical range. That is, the value 'virtual' was not considered a typical actor type. The form suggested for the gesture is retrieved from the database entry for the value virtual; in this way the gesture highlights the surprising feature of the object.

When the behavior selection module receives the suggestions from the generator module, it notices that both a beat and an iconic gesture were suggested for PUNK1. Using the rule of gesture class priority (beats being the lowest class in the gesture family), the module filters out the beat and leaves in the iconic. No further conflicts are noticed and no further filters have been included in this example. The resulting tree is shown in Figure 2c.

Lastly the behavior scheduling module compiles the XML tree, including all suggestions not filtered out, into an action plan ready for execution by an animation engine as described in section 4.4. The final schedule (without viseme codes) is shown in Figure 2d.

The second sentence is processed in much the same way. Part of the output of the behavior generator is shown in Figure 9. Two particular situations that arise with this sentence are of note. The first is that the action, "to type in", is identified by the language module because an action description for typing is found in the action database. Therefore the gesture suggestion module can suggest the use of an iconic gesture description, because the action occurs within a rheme. See Figure 10. for a snapshot of the generated "typing" gesture. The second one is that although PUNK1 ("the actor") was identified again, no gesture was suggested for this object at this time because it is located inside a theme as opposed to a rheme part of the clause.

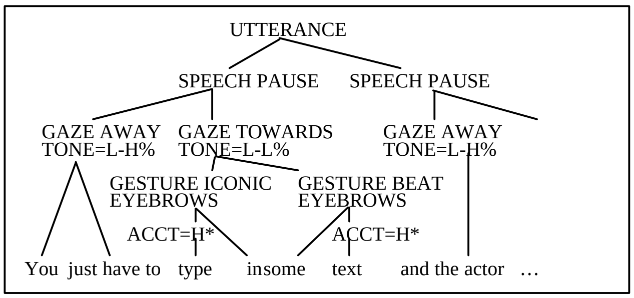

Figure 9. Part of the output XML tree for first example

Figure 10. "You just have to type in some text…"

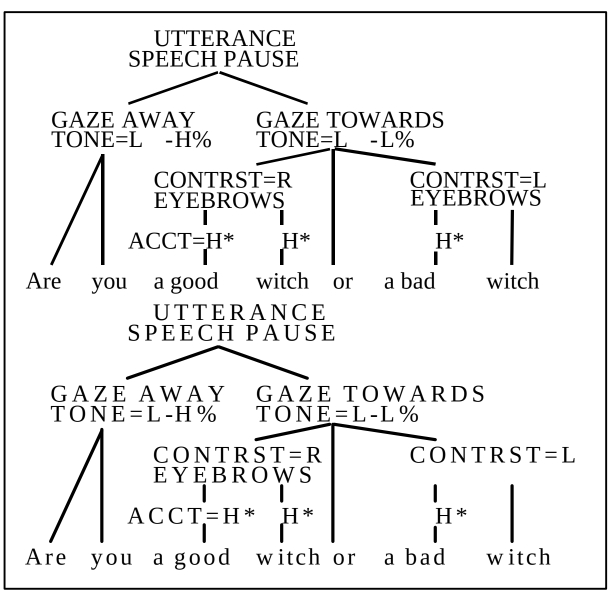

Figure 11. The output XML tree for second example

As an example of a different kind of a nonverbal behavior assignment, let's look at how the system processes the sentence "Are you a good witch or a bad witch?". The output of the behavior generation module is shown in Figure 11. As well as suggesting the typical behaviors seen in the previous examples, here the language tagger has identified two contrasting adjectives in the same clause, "good" and" bad." They have been assigned to the same contrast group. When the gesture suggestion module receives the tagged text, generation rules suggest a contrast gesture on the "a good witch" object and on the "a bad witch" object. Furthermore, the shape suggested for these contrast gestures is a right hand pose for the first object and a left hand pose for the second object since there are exactly two members of this contrast group. When filtering, the gesture selection module notices that the contrasting gestures were scheduled to peak at exactly the same moment as a couple of hand beats. The beats are filtered out using the gesture class priority rule, deciding that contrasting gestures are more important than beats. See Figure 12. for a snapshot of the contrast gesture.

Figure 12. "Are you a good witch or a bad witch?"

6. CONCLUSIONS / FUTURE WORK

The BEAT toolkit is the first of a new generation (the *beat generation*) of animation tool that extracts actual linguistic and contextual information from text in order to suggest correlated gestures, eye gaze, and other nonverbal behaviors, and to synchronize those behaviors to one another. For those animators who wish to maintain the most control over output, BEAT can be seen as a kind of "snap-to-grid" for communicative actions: if animators input text, and a set of eye, face, head and hand behaviors for phrases, the system will correctly align the behaviors to one another, and send the timings to an animation system. For animators who wish to concentrate on higher level concerns such as personality, or lower level concerns such as motion characteristics, BEAT takes care of the middle level of animation: choosing how nonverbal behaviors can best convey the message of typed text, and scheduling them.

While the automated specification of nonverbal behavior demonstrated here is no doubt inferior to rotoscoping, motion capture, or the skilled eye of a trained animator, it may be adequate for many purposes. Certainly, this kind of automated specification improves over the hand-animated associations between language and nonverbal behavior used in many current web-based agents, or other autonomous systems. It also provides a first pass at the desired behaviors in those cases where manual improvement can follow up. The system is meant to suggest a baseline that without any tweaking will at least appear plausible, but it invites the input of an animator at any stage to affect the final output.

Future work includes more extensive automatic linguistic tagging and additional inferencing, relying further on WordNet or even on a database of common sense knowledge, such as Cyc [21]. In addition further work is needed on the notion of the gesture ontology, including some basic spatial configuration gesture elements. As it stands, hand gestures cannot be assembled out of smaller gestural parts, nor can they be shortened. When gesture descriptions are read from the knowledge base, they are currently placed in the animation schedule unchanged. The Behavior Scheduler makes sure the stroke of the gesture aligns with the correct word, but does not attempt to stretch out the rest of the gesture, for instance to span a whole phrase that needs to be illustrated. Similarly, it does not attempt to slow down or pause speech to accommodate a complex gesture, a phenomenon observed in people. Finally, additional nonverbal behaviors should be added: wrinkles of the forehead, smiles, ear wiggling. The system will also benefit from a visual interface that displays a manipulatable timeline where either the scheduled events themselves can be moved around or the rules behind them modified.

In the meantime, we hope to have demonstrated that the animator's toolbox can be enhanced by the knowledge about gesture and other nonverbal behaviors, turntaking, and linguistic structure that are incorporated and (literally) embodied in the Behavior Expression Animation Toolkit.

7. REFERENCES

[1] Amaya, K., Bruderlin, A., and Calvert, T., Emotion from motion. *Proc. Graphics Interface'96*, pp. 222-229, , 1996.

[2] Badler, N., Bindiganavale, R., Allbeck, J., Schuler, W., Zhao, L., and Palmer., M., Parameterized Action Representation for Virtual Human Agents., in *Embodied Conversational Agents*, J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, Eds. Cambridge, MA: MIT Press, 2000, pp. 256-284.

[3] Becheiraz, P. and Thalmann, D., A Behavioral Animation System for Autonomous Actors personified by Emotions, *Proc. of the1st Workshop on Embodied Conversational Characters,* 57-65, 1998.

[4] Blumberg, B. and Galyean, T. A., Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments. *Proc. SIGGRAPH '95*, pp. 47-54, Los Angeles, CA, 1995.

[5] Bodenheimer, B., Rose, C., and Cohen, M., Verbs and Adverbs: Multidimensional Motion Interpolation, *IEEE Computer Graphics and Applications*, vol. 18 (5), pp. 32-40, 1998.

[6] Brand, M., Voice Puppetry. *Proc. SIGGRAPH '99*, pp. 21-28, Los Angeles CA, 1999.

[7] Bregler, C., Covell, M., and Slaney, M., Video Rewrite: driving visual speech with audio. *Proc. SIGGRAPH '97*, pp. 353-360, Los Angeles, CA, 1997.

[8] Calvert, T., Composition of realistic animation sequences for multiple human figures, in *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*, N. Badler, B. Barsky, and D. Zeltzer, Eds. San Mateo, CA: Morgan-Kaufmann, pp. 35-50, 1991.

[9] Cassell, J., Nudge, Nudge, Wink, Wink: Elements of Face-to-Face Conversation for Embodied Conversational Agents, in *Embodied Conversational Agents*, J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, Eds. Cambridge: MIT Press, pp. 1-27, 2000.

[10] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., and Stone, M., Animated Conversation: Rule-Based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversational Agents. *Proc. Siggraph '94*, pp. 413-420, Orlando, 1994.

[11] Cassell, J. and Prevost, S., Distribution of Semantic Features Across Speech and Gesture by Humans and Computers. *Proc. Workshop on the Integration of Gesture in Language and Speech,* pp. 253-270, Newark, DE, 1996.

[12] Cassell, J., Torres, O., and Prevost, S., Turn Taking vs. Discourse Structure: How Best to Model Multimodal Conversation, in *Machine Conversations*, Y. Wilks, Ed. The Hague: Kluwer, pp. 143-154, 1999.

[13] Chang, J., Action Scheduling in Humanoid Conversational Agents, M.S. Thesis in Electrical Engineering and Computer Science. Cambridge, MA: MIT, 1998.

[14] Chi, D., Costa, M., Zhao, L., and Badler, N., The EMOTE model for effort and shape. *Proc. SIGGRAPH '00*, pp. 173-182, New Orleans LA, 2000.

[15] Colburn, A., Cohen, M. F., and Drucker, S., The Role of Eye Gaze in Avatar Mediated Conversational Interfaces, *MSR-TR-2000-81*. Microsoft Research, 2000.

[16] Halliday, M. A. K., *Explorations in the Functions of Language*. London: Edward Arnold, 1973.

[17] Hirschberg, J., Accent and Discourse Context: Assigning Pitch Accent in Synthetic Speech. *Proc. AAAI 90*, pp. 952-957, 1990.

[18] Hiyakumoto, L., Prevost, S., and Cassell, J., Semantic and Discourse Information for Text-to-Speech Intonation. *Proc. ACL Workshop on Concept-to-Speech Generation,* Madrid, 1997.

[19] Huang, X., Acero, A., Adcock, J., Hon, H.-W., Goldsmith, J., Liu, J., and Plumpe, M., Whistler: A Trainable Text-to-Speech System. *Proc. 4th Int'l. Conf. on Spoken Language Processing (ICSLP '96)*, pp. 2387-2390, Piscataway, NJ, 1996.

[20] Kurlander, D., Skelly, T., and Salesin, D., Comic Chat, *Proc. of SIGGRAPH '96*, pp. 225-236, 1996.

[21] Lenat, D. B. and Guha, R. V., *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Reading, MA: Addison Wesley, 1990.

[22] Massaro, D. W., *Perceiving Talking Faces: From Speech Perception to a Behavioral Principle*. Cambridge, MA: MIT Press, 1987.

[23] McNeill, D., *Hand and Mind: What Gestures Reveal about Thought*. Chicago, IL/London, UK: The University of Chicago Press, 1992.

[24] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K., Introduction to Wordnet: An On-line Lexical Database, 1993.

[25] Nagao, K. and Takeuchi, A., Speech Dialogue with Facial Displays: Multimodal Human-Computer Conversation. *Proc. ACL-94*, pp. 102-109., , 1994.

[26] Pearce, A., Wyvill, B., Wyvill, G., and Hill, D., Speech and expression: a computer solution to face animation. *Proc. Graphics Interface*, pp. 136-140, 1986.

[27] Pelachaud, C., Badler, N., and Steedman, M., Generating Facial Expressions for Speech, *Cognitive Science*, 20(1), pp. 1–46, 1994.

[28] Perlin, K., Noise, Hypertexture, Antialiasing and Gesture, in *Texturing and Modeling, A Procedural Approach*, D. Ebert, Ed. Cambridge, MA: AP Professional, 1994.

[29] Perlin, K. and Goldberg, A., Improv: A System for Scripting Interactive Actors in Virtual Worlds, *Proceedings of SIGGRAPH '96*, pp. 205-216, 1996.

[30] Prevost, S. and Steedman, M., Specifying intonation from context for speech synthesis, *Speech Communication*, vol. 15, pp. 139-153, 1994.

[31] Roehl, B., Specification for a Standard Humanoid, Version 1.1, H. A. W. Group, Ed. http://ece.uwaterloo.ca/~h-anim/spec1.1/, 1999.

[32] Taylor, P., Black, A., and Caley, R., The architecture of the Festival Speech Synthesis System. *Proc. 3rd ESCA Workshop on Speech Synthesis*, pp. 147-151, Jenolan Caves, Australia, 1998.

[33] Waters, K. and Levergood, T., An Automatic Lip-Synchronization Algorithm for Synthetic Faces. *Proc. of the 2nd ACM international conference on Multimedia*, pp. 149-156, San Francisco CA, 1994.

[34] Yan, H., Paired Speech and Gesture Generation in Embodied Conversational Agents, M.S. thesis in the Media Lab. Cambridge, MA: MIT, 2000.