The Expressive Gaze Model: Using Gaze to Express Emotion

Brent J. Lance ■ *Army Research Laboratory*

Stacy C. Marsella ■ *University of Southern California*

The Expressive Gaze Model is a hierarchical framework for composing simple behaviors into emotionally expressive gaze shifts for virtual characters. Its primary components are the Gaze Warping Transformation, which generates emotionally expressive head and torso movement in a gaze shift, and an eye movement model.

"The eyes are the most important part of an expression, and must be drawn with extreme care. Any jitter or false move ... destroys both communication and believability."[1]—Frank Thomas and Ollie Johnston

Early animators realized that the eyes are an important aspect of the human face regarding the communication and expression of emotion. They also found that properly animating believable, emotionally expressive gaze is extremely difficult. If it's done improperly, viewers will immediately notice, destroying the illusion of life the animators were striving for.

This problem becomes even more difficult with interactive virtual characters—computer-controlled characters in a virtual environment that interact with each other and human users through conversation. Unlike hand-animated characters, interactive virtual characters must function in the virtual environment and interact with human users in real time. This also entails gazing at arbitrary targets as they appear in the environment.

For example, modern video game environments, such as World of Warcraft or Oblivion, have thousands of computer-controlled nonplayer characters with which users can interact. Producing hand-generated animations for each of these characters consumes large amounts of time and money that would often be better spent improving other game aspects.

To address this issue, we present the Expressive Gaze Model (EGM), a method that enables virtual characters to generate believable gaze shifts that communicate a desired emotional expression. We define a believable, emotionally expressive gaze as an eye, head, or torso movement that places the forward vector of the virtual character's eyes on a new target while portraying behaviors that cause a human viewer to attribute a desired emotional expression to that character, in a way that doesn't damage the viewer's suspension of disbelief.

The Expressive Gaze Model

The EGM is a hierarchical framework with two main components: a library of Gaze Warping Transformations (GWTs) and a procedural model of eye movement.

The library is generated from motion capture,[2] procedural animation,[3] or hand animation. Using this library, users can produce gaze shifts portraying any low-level gaze behavior in the library. The eye movement model similarly uses a small number of physical behaviors to generate varieties of eye movement and integrate them with the GWT-based movement.

Eye movement by itself is highly stereotypical and contains little emotional content. It also functions on a different timescale from that of head or torso movement, with different constraints. So, we model eye movement differently from head and torso movement but constrain the relationship between them.

The EGM doesn't determine what the animated character should look at or what the correct emotional response is to a given situation. Instead, it

 Published by the IEEE Computer Society

produces individual gaze shifts to which a human viewer can attribute emotional content.

Figure 1 shows how we use the EGM to produce a gaze shift. First, we perform two motion capture sessions. The first is of a shift that contains physical behaviors, such as a bowed head, explicitly intended to convey emotional information. The other is of an emotionally neutral gaze shift, which contains no physical behaviors beyond those needed to change targets. Both sessions start and end gazing at the same targets as each other. From these, we derive a GWT representing the behaviors performed during the emotionally expressive gaze. Then, we apply that GWT to a new emotionally neutral gaze shift, which can start and end gazing at different targets.

This results in a new partial gaze shift, consisting of head and torso movement that gazes at a new target but displays the expressive behaviors portrayed in the original emotionally expressive gaze shift. Next, the eye movement model automatically layers appropriate eye behavior onto the partial gaze shift. This produces the motion data for an emotionally expressive gaze shift, which we then render. The final result is a gaze shift that displays a desired set of expressive gaze behaviors toward an arbitrary target.

For a look at other research on modeling gaze, see the "Related Work on Modeling Gaze" sidebar.

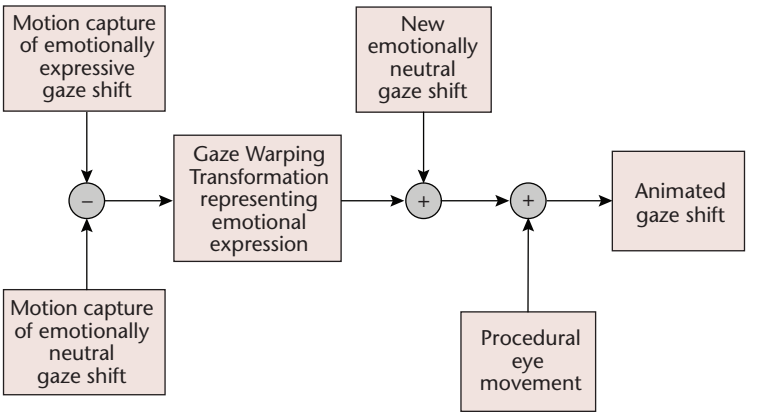

Figure 1. The Expressive Gaze Model (EGM). This image shows the process the EGM uses to generate an emotionally expressive gaze shift.

The Gaze Warping Transformation

The GWT is a lightweight combination of temporal-scaling and spatial-transformation parameters derived from the papers "Motion Warping"[4] and "Emotion from Motion."[5] Basically, it's a representation of the difference between two gaze shifts. By taking any two gaze shifts, we can derive a GWT from the difference between them, such that applying it to one gaze shift produces the other gaze shift.

Data Description

For this work, we recorded the motion data using Ascension's Flock of Birds (FOB) technology. FOB uses the movement of sensors through a magnetic field to measure six degrees of freedom (DOF)—the x, y, and z positions and the roll, pitch, and yaw angles—at 60 Hz. We attached three sensors to an actor, one at the side of the head, one at the base of the neck, and one at the base of the spine.

The FOB collects motion data as a set of time-series data points. Each of the three electromagnetic sensors records a time stamp, the sensor's position in (x, y, z) coordinates, and the sensor's orientation as an Euler angle.

To perform motion capture of gaze shifts, we placed targets around the motion capture area at approximately 10-degree intervals, and the actor gazed from one specified target to another.

We segmented the sets of captured data into individual gaze shifts, which we defined as at most one major head movement, one major torso movement, and one major eye movement. We segmented the data on the basis of the points in time when the head is stationary in the direction with the largest angular displacement. This is because the head usually performs the easiest-to-distinguish and highest-amplitude movements.

Related Work on Modeling Gaze

Although much research has focused on modeling gaze in interactive virtual characters, such as that of Sooha Park Lee and her colleagues,[1] little of it addresses expressing emotion through gaze. However, Atsushi Fukayama and his colleagues demonstrated that a human viewer would attribute emotion and personality to an animated pair of disembodied eyes solely on the basis of the direction and patterns of gaze shifts.[2] In addition, considerable research on animating facial expressions has used the eyes' shape to portray emotions.[3]

Unlike both these approaches, our research doesn't address the gaze target or the shape of the eyes and the immediately surrounding face. Instead, we focus on how to use the rest of the body—the head and torso—to express emotion during a gaze shift and how to integrate the eyes with that movement.

References

1. S.P. Lee, J.B. Badler, and N.I. Badler, "Eyes Alive," *ACM Trans. Graphics*, vol. 21, no. 3, 2002, pp. 637–644.
2. A. Fukayama et al., "Messages Embedded in Gaze of Interface Agents—Impression Management with Agent's Gaze," *Proc. 2002 SIGCHI Conf. Human Factors in Computing Systems* (SIGCHI 02), ACM Press, 2002, pp. 41–48.
3. S.M. Platt and N.I. Badler, "Animating Facial Expressions," *ACM Siggraph Computer Graphics*, vol. 15, no. 3, 1981, p. 252.

Motion Curves and Keyframes

We captured the motion data from Ascension's Flock of Birds technology as a set of motion curves in a Biovision Hierarchy (BVH) format. In this representation, each degree of freedom (DOF) of the animated character has its own 2D motion curve, with the animation's frame number as the *x*-axis and the DOF's magnitude as the *y*-axis. To derive Gaze Warping Transformations (GWTs) from these motion curves, we first derived the keyframe representation of the motion data.

Keyframes can be viewed as a sparse representation of motion curves. A motion curve's keyframes are a subset of the animation frames in the motion curve such that interpolating between the keyframes can reconstruct the original motion.

To find a gaze shift's keyframes, we aligned that shift with a "stereotypical" gaze shift. We found the stereotypical gaze shift by averaging 33 emotionally neutral gaze shifts that were scaled to the same length of time and transformed to begin at the same initial state. We then placed keyframes on this gaze shift to minimize the least-squared error between the original movement and a cubic-spline interpolation across the keyframes. We used six keyframes for each gaze shift, but modeling more complex gaze behavior might require more keyframes, even up to the full set of collected data.

To align the gaze shifts, we used an algorithm (derived from the paper "Emotion from Motion"[1]) that aligns the curves of two animations or motion capture sessions on the basis of the ratio of movement that has occurred by a specific frame to that which occurs throughout the entire curve. To find the values for alignment, we used

$$\hat{f}(t) = \frac{\sum_{\tau=0}^{t} \left|\mathbf{v}(\tau)\right|}{\sum_{\tau=0}^{t_{end}} \left|\mathbf{v}(\tau)\right|}, \quad (1)$$

where $\mathbf{v}(\tau)$ is the head's 3D velocity vector. We used this vector to approximate the movement of the entire body because the head movement is usually of higher amplitude than that of the torso during a gaze shift.

For each animation frame, we used this formula to calculate the frame alignment value, or how much of the movement in that gaze shift has occurred by that frame. For example, 0 percent of the total movement in the gaze shift will have occurred by the first frame, whereas 100 percent of the total movement will have occurred by the last frame. Each frame in between receives a monotonically increasing value between 0 and 1.

Deriving the GWT

The GWT is a point-to-point transformation that transforms one gaze shift's keyframes into another gaze shift's keyframes. Consequently, each gaze shift's keyframes must align. (For a discussion of keyframes and alignment, see the sidebar "Motion Curves and Keyframes.") If they don't align, a transformation derived from a specific point in time in one gaze shift might be applied to a noncorresponding point in time in another gaze shift, leading to visible artifacts in the resulting animations.

Given the motion data from the emotionally expressive and emotionally neutral gaze shifts, we derive the GWT by first obtaining the two gaze shifts' keyframes. We then find a set of point-to-point warping parameters[4] that converts the keyframes of the motion curve representing each DOF in the emotionally neutral animation into the keyframes of the motion curve for the corresponding DOF in the emotionally expressive movement.

Algorithm 1 shows the GWT derivation algorithm. (For all the algorithms mentioned in this article, see the related sidebar at the end of this article.) For a mathematical depiction of the GWT, see the sidebar "Mathematically Defining the Gaze Warping Transformation."

Applying the GWT

The process of applying the GWT is similar, beginning with motion data of an emotionally neutral gaze shift. We obtain this shift's keyframes (as described in the sidebar "Motion Curves and Keyframes") and apply the GWT transformation parameters to each of them, as shown in Algorithm 2 and the sidebar "Mathematically Defining the Gaze Warping Transformation." Finally, we use cubic interpolation to produce motion curves from the resulting transformed keyframes and then use these motion curves to drive an animated character's gaze.

The Eye Movement Model

The EGM's eye movement model[6] is based primarily on the visual-neuroscience literature. This model ensures that our characters perform realistic eye movement during animated gaze shifts.

Types of Eye Movement

R. John Leigh and David Zee's overview of visual neuroscience identified several functional classes of eye movement.[7] Those relevant to this research are

- saccades, which are rapid, highly stereotyped eye movements toward a specific target, and
- vestibular and optokinetic movements, which maintain visual fixation during head movement.

We determined the keyframes by choosing those frames with the same alignment value as the keyframes on the standard gaze shift (see Algorithm 7 in the "Algorithms" sidebar at the end of the main article). For example, if a keyframe from the standard gaze shift had a 0.2 alignment value, we selected the frames in other gaze shifts with a 0.2 value as the corresponding keyframes. Once we had the set of keyframes from both the emotionally expressive and the emotionally neutral gaze shifts, we derived the GWT.

Given this set of keyframes based on the head orientation, we obtained the keyframes for every other DOF by simply selecting the frames that occur at the same time as those in this set of keyframes.

Reference

1. K. Amaya, A. Bruderlin, and T. Calvert, "Emotion from Motion," *Proc. Conf. Graphics Interface 1996*, Canadian Information Processing Soc., 1996, pp. 222–229.

In addition, Leigh and Zee discussed how head and eye movement can be combined through saccades and the *vestibulo-occular reflex* (VOR). So, our model employs saccades and VOR movement to implement combined eye-head saccades.

We represent eye movement with two DOF representing the eyes' horizontal and vertical orientation. Both orientations range from +45 to −45 degrees, with the horizontal increasing to the right and the vertical increasing upward. In both cases, 0 degrees is straight ahead. Although the mechanical limits of human eye movement are closer to ±55 degrees, evidence shows that neural limits saturate eye movement starting at ±45 degrees. The same motion curve drives both eyes, meaning that the eyes maintain a parallel orientation. Adding the eyes' ability to converge on a target complicates the problem but would increase the generated gaze shifts' realism.

Our representation of the gaze targets consists of a local (pitch and yaw) angular coordinate system that represents the horizontal and vertical rotations that must be applied to the neck joint for the character's head to directly face the target. The origin centers on the character's neck joint; (0, 0) represents a target directly in front of the character's face.

Figures 2 through 5 show how we represent eye movements for each type of gaze shift. The gaze-

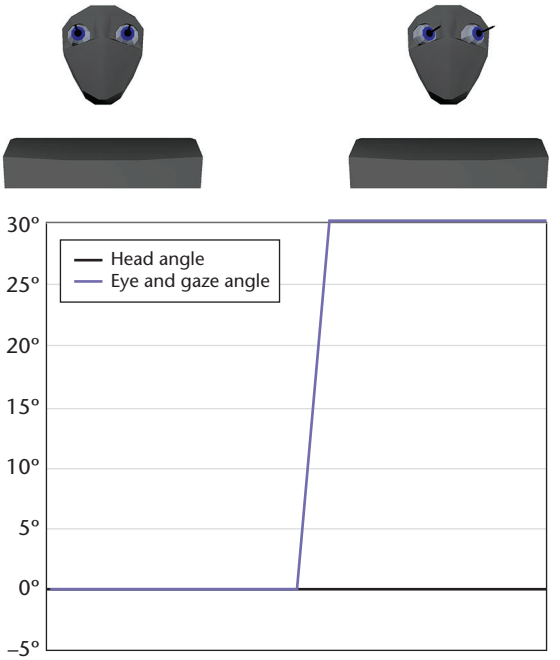

Figure 2. A stereotypical saccade. Motion curves for a 30-degree saccade showing head, eye, and combined eye and head gaze as generated by our model.

angle curve represents the gaze's target, combining eye and head orientation. The eye-angle curve represents the angle of the eyes within their orbit. The head-angle curve represents the head's angle relative to its initial position.

Saccades

Figure 2 shows the motion curves for a stereotypical saccade in 1D and an animated figure showing the saccade's beginning and ending. Saccadic movements' size, speed, and duration are closely related. As the amplitude increases, so do the speed and duration. This relationship is called the *main sequence relationship* and defines ranges for standard saccades. In a normal human, saccade duration is linearly related to the amplitude for eye movements under 50 degrees. A saccade will rarely be longer than 100 ms, approximately four to five animation frames. Eye movement outside these ranges is either nonsaccadic or an abnormal saccade, often symptomatic of pathology.[7]

Accordingly, we represent saccadic movement (see Algorithm 3) as a rotation to the desired target, with additional considerations. We approximate the main sequence relationship as a linear relation between the amplitude and the number of frames the saccade takes to execute. For each 10 degrees of amplitude, we add one intermediate frame, limiting us to approximately four to five frames. Then, linear interpolation between the start and end positions determines the eye's intermediate orientation across these frames. So, velocity and duration aren't

Mathematically Defining the Gaze Warping Transformation

We derive the Gaze Warping Transformation (GWT) from the difference between the keyframes of two animation curves, $u(t)$ and $v(t)$, respectively defined as the sets of (*frame*, *value*) pairs (ut_i, u_i) and (vt_i, v_i).[1] We represent the GWT as an $m * n$ set of (c_i, b_i) pairs, where m is the number of degrees of freedom (DOF) in the animated body and n is the number of keyframes in the animation. Each (c_i, b_i) pair then represents the difference between the keyframes (ut_i, u_i) and (vt_i, v_i) (see Algorithm 1 in the "Algorithms" sidebar).

In each pair of the GWT, c_i is a time-scaling factor that represents the temporal difference between two keyframes, whereas b_i is a spatial-offset parameter that represents the difference between the spatial values of the two keyframes. We calculate c_i from the keyframes of u and v (see Figure A):

$$c_i = \frac{ut_i - ut_{i-1}}{vt_i - vt_{i-1}}.$$

We calculate b_i from the spatial values of u and v (see Figure A):

$$b_i = u_i - v_i.$$

Because the GWT is a point-wise transformation of the keyframes, we perform these operations for each keyframe of every motion curve of the gaze shift.

Similarly, given the keyframes of an animation curve $v(t)$, with m DOF and n keyframes, a GWT consisting of an $m * n$ sized set of (c_i, b_i) pairs can transform $v(t)$ into a new gaze shift $u(t)$ (see Algorithm 2). For each keyframe, we use c_i to derive the temporal interval between two adjacent keyframes ut_i and ut_{i-1} of u, on the basis of the interval between the corresponding keyframes vt_i and vt_{i-1} of v:

$$ut_i = ut_{i-1} + c_i(vt_i - vt_{i-1}).$$

We then use b_i to determine the amplitude of keyframe u_i of u from the corresponding keyframe v_i of v:

$$u_i = b_i + v_i.$$

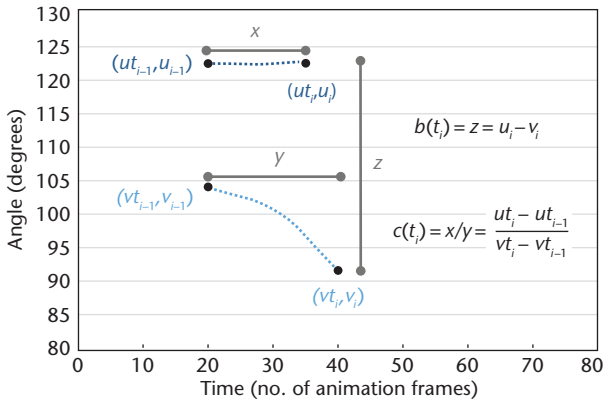

Figure A. Calculating parameters *c* and *b*, which are the temporal and spatial difference between two adjacent keyframes (ut_i, u_i) and (vt_i, v_i).

Reference

1. B. Lance and S.C. Marsella, "Emotionally Expressive Head and Body Movement during Gaze Shifts," *Intelligent Virtual Agents*, LNCS 4722, 2007, pp. 72–85.

directly controlled but are implicitly determined by the amplitude, obeying a relationship similar to the main sequence relationship in a human. The saccade generation algorithm assumes that the head is stationary during the saccade. For combined head and eye movement, the eye movement model uses the eye-head saccade, which we describe later.

The VOR

Figure 3 shows the motion curve for a head movement with the VOR and an animated character demonstrating the VOR movement. Because the VOR is nonsaccadic, it's not subject to the main sequence relationship, allowing slower eye rotation to match the head rotation. We implement VOR eye movement by counter-rotating the eyes to the head (see Algorithm 4).

Eye-Head Combined Movement

We model two similar versions of the combined eye-head saccade (see Figures 4 and 5), although the same algorithm (see Algorithm 5) can generate both movements.

For either version, we first generate head and torso movement using GWTs. Then, the eye movement is automatically layered on the head movement using the representation of both saccades and the VOR. For eye-head saccades of less than 45 degrees, we determine the eye position by generating a stereotypical saccade to the target once the head has turned more than 1 degree away from its starting location (see Figure 4a). Once the eyes have reached their target, the VOR will keep them on target as the head slowly turns toward it.[7] The images above the graph in Figure 4a show the character at the beginning of the movement, just after the saccade has occurred and the VOR is taking control of the eye movement, and at the end of the gaze shift.

We model an eye-head saccade of greater than ±45 degrees slightly differently (see Figure 5a). In this case, we determine the eyes' position by performing a saccade to 45 degrees once the head has

turned more than 1 degree away from its starting location. The eyes will then remain in that orientation, relative to the head, until the head has rotated enough that the eyes are on target. At that time, the VOR takes effect, and the eyes remain on target until the head movement ends. The images above the graph in Figure 5a show the gaze's initial position, the gaze at the end of the 45-degree saccade, and the gaze shift's terminal position. Although slight differences exist between the motion curves provided here and those from actual humans (see Figures 4b and 5b), these differences are unlikely to be visible to an observer.

Implementation

Our view of an emotionally expressive gaze is that of a mapping between

- a space of physical behaviors that can be performed while gazing and
- a space of possible emotions that can be attributed to a gaze shift displaying those behaviors.

We've used the EGM to examine three ways to explore these spaces.

First, we used the EGM to explore only the gaze behavior, demonstrating its capability to perform nonemotional gaze shifts. Second, we explored a point-to-point mapping between emotion and behavior by having an actor portray specific emotional states through gaze shifts. Finally, we examined a more piecemeal approach to generating gazes, composing low-level gaze behaviors and seeing how gaze shifts displaying these compositions portray emotion.

Generating Nonemotional Gazes

To develop a GWT-based library for portraying nonemotional gaze shifts, we used motion data consisting of an actor facing straight ahead and then turning to gaze at targets until the actor's face was oriented toward them. We placed targets around the actor horizontally and vertically at 10-degree intervals from −90 degrees to 90 degrees, with 0 degrees being directly in front of the actor. This resulted in data for 37 gaze shifts: 18 horizontal gaze shifts, 18 vertical gaze shifts, and one capture of the actor gazing at the target directly in front.

To obtain the library from this data, we derived the GWTs between each of the 36 horizontal and vertical gaze shifts and the stationary straight-ahead gaze. Then, to generate a gaze shift to a desired target, we used our gaze generation algorithm (see Algorithm 6). In this algorithm, we first calculate the distance the head must rotate from its initial position to achieve the gaze target. We then find the GWTs with the closest magnitude to this distance for the pitch and yaw axes.

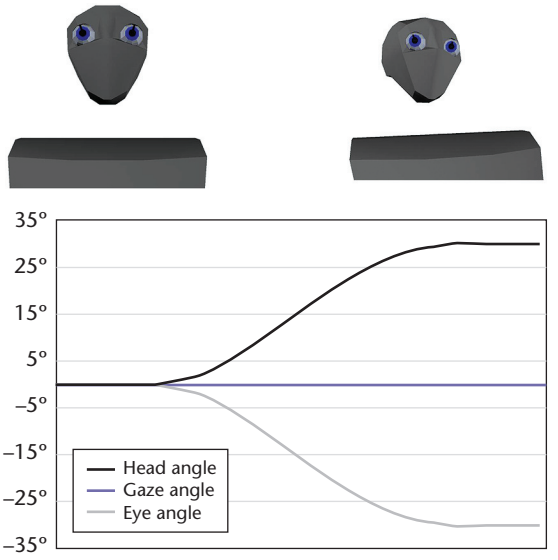

Figure 3. The vestibulo-ocular reflex (VOR). Motion curves for a 30-degree head movement with VOR showing head, eye, and combined eye and head gaze as generated by our model.

We then scale these GWTs so that they'll rotate the head the desired amount, and we apply them to the stationary straight-ahead gaze. We interpolate the resulting keyframes and apply inverse kinematics to the resulting motion curve. Finally, we layer eye movements on these motion curves, which then drive an animated character.

Generating Emotional Gazes

Using the nonverbal-behavior research literature, animation literature, and acting literature, we defined a set of emotions and a set of behaviors possibly portrayed by these emotions. We asked our actor to gaze from a target directly in front of her to one 60 degrees to her right, while performing these behaviors in a manner that expressed the desired emotion. We derived GWTs from these gaze shifts and layered them onto neutral gaze shifts.

Although we performed no formal human-subject evaluation, the methodology replicated the performed behaviors in gaze shifts to new targets. However, the head's vertical orientation (pitch) is a strong emotional signal that interfered with other behaviors being displayed. Keeping the pitch within an approximately ±10-degree band would minimize its interference with the desired emotional signal.

Additionally, this method only lets the EGM generate gazes displaying individual emotional states. It also requires motion capture data for each emotional state, some of which might be more difficult for an actor to portray than others. Finally,

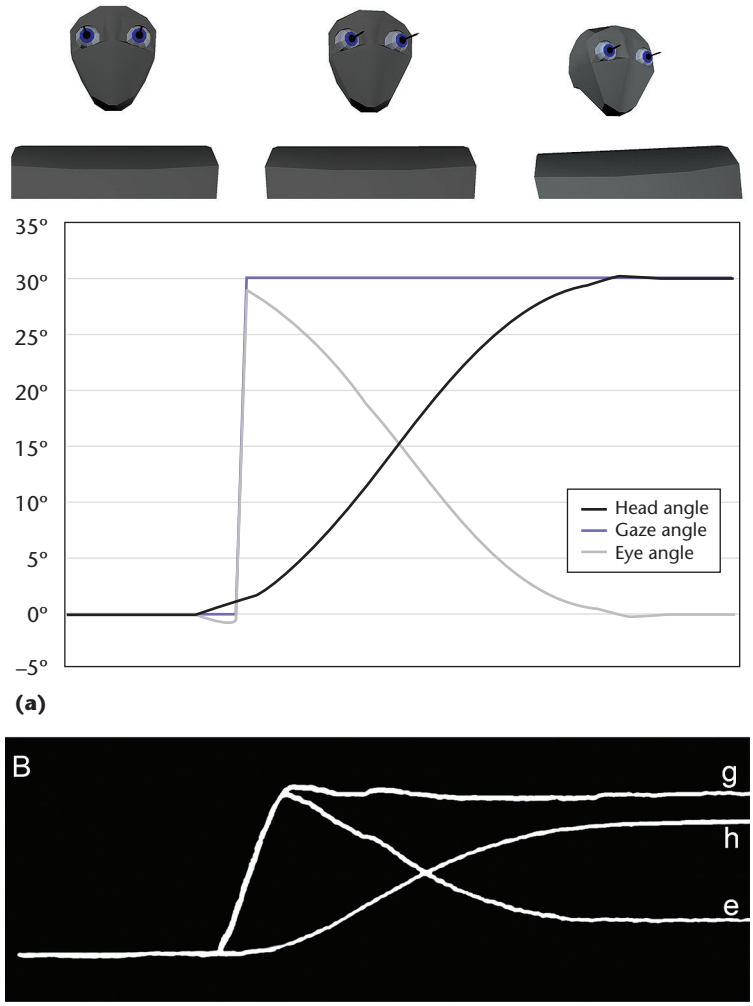

Figure 4. An eye-head saccade within the eye motor limit. (a) Motion curves for the character's head, eyes, and combined eye and head gaze as generated by our model. (b) Motion curves for the head (h), eyes (e), and combined eye and head gaze (g) for an actual human.[8] (Figure 4b source: Taylor & Francis Group, used with permission.)

it doesn't predict the attribution of emotion to behaviors outside these specific emotional states.

Generating Behavior-Based Combinatorial Gazes

As an alternative to point-to-point mapping between gaze behavior and emotion, we tried to map between a set of low-level gaze behaviors derived from the nonverbal-behavior literature and a space of emotions that these behaviors could portray. To do this, we first collected motion data for eight expressive gaze behaviors:

- head tilted upwards,
- head unbowed,
- head bowed,
- torso unbowed,
- torso bowed,
- faster velocity,
- moderate velocity, and
- slower velocity.

Although we could model many additional gaze behaviors using the GWT (for example, wider variations of posture), this limited set gave us a starting point.

For each head and torso behavior, we collected data from three movements. In these movements, the actor first gazed at the target directly in front and then rotated the gaze toward a target 30 degrees to the right. The first movement transitioned from a neutral posture to a posture displaying the behavior. The second movement began and ended with the behavior being displayed. The third movement started with the behavior being displayed and ended in a neutral posture.

This let us develop characters that can "enter" an emotional state by performing a gaze shift that transitions from an emotionally neutral shift to one displaying emotional behavior. The character can then perform multiple gaze shifts within that emotion. Finally, the character can "exit" the emotion and transfer back to an emotionally neutral state. For example, collecting a bowed-head behavior required four movements, during each of which the actor gazed from one target to the other. During the first movement, the actor began with an unbowed head and ended with it bowed. For the second, the actor began and ended the movement with a bowed head, keeping it bowed the entire time. For the third movement, the actor began with a bowed head and ended with it unbowed. Finally, the actor performed a neutral gaze shift with the head unbowed. From this motion data, we derived GWTs and produced animations from them.

Using these animations, we performed an empirical study to develop the mapping between emotion and behavior.[9] The results show that these low-level gaze behaviors, when annotated with emotional values and composed in accordance with those values, display the composed emotions within certain constraints. Besides providing the mapping, the study demonstrated the GWT's utility as a research tool beyond generating animations and pointed out future research areas.

All the animations used in our study, as well as many other animations produced using the EGM, are at http://people.ict.usc.edu/~blance/AnimationGallery/AnimationGallery.html.

The Current EGM Approach's Benefits and Limitations

The GWT isn't fundamentally necessary to the EGM. The library of gaze movements could consist

of other representations of motion styles, such as those that M. Alex O. Vasilescu presented.[10] However, we use this GWT representation because it provides several benefits.

Behavior Layering

As a representation of emotionally expressive gaze behavior, GWTs can be used to transform gaze shifts to arbitrary targets, layering the desired behaviors on top of the new gaze shift. This lets us generate gaze shifts expressing the desired behavior to arbitrary targets while requiring motion capture of only the gaze shift expressing the desired behavior.

Compositionality

We can also compose gaze behaviors by applying multiple GWTs representing different behaviors to a single gaze shift. In this way, we can generate a wider variety of gaze shifts with a smaller library of motion data. We can also explore the relationship between gaze behavior and the attribution of emotion to gaze.[9]

However, not all behavior compositions result in expressive gaze shifts that display all the composed behaviors, and not all behavior compositions predictably portray emotion. Developing a set of expressive, easily composable gaze behaviors, similar to the MPG-4 Facial Animation Parameters standard, and mapping between them and emotional expression remain interesting research areas.

Additional Advantages

In addition, because GWT construction doesn't depend on anatomy, we can add joints and DOFs to the GWT representation as needed. GWTs also don't depend on animation data representation. This lets us determine them from or apply them to different sources of animation data, such as motion capture, handcrafted animation, or even procedural animation, as long as we can obtain the keyframes or motion curves from the animation system.

Limitations

However, because we based the GWT on simple geometric transformations, our approach has several limitations. The most immediate is that animations generated using simple techniques such as this might have artifacts that require inverse kinematics to address.[4] In addition, the GWT currently doesn't represent periodic behavior, such as a nodding head, nor does it represent patterns of multiple gaze shifts.

The eye movement model also includes some assumptions that need addressing. The primary assumption is that the target doesn't move during the generation of a gaze shift.

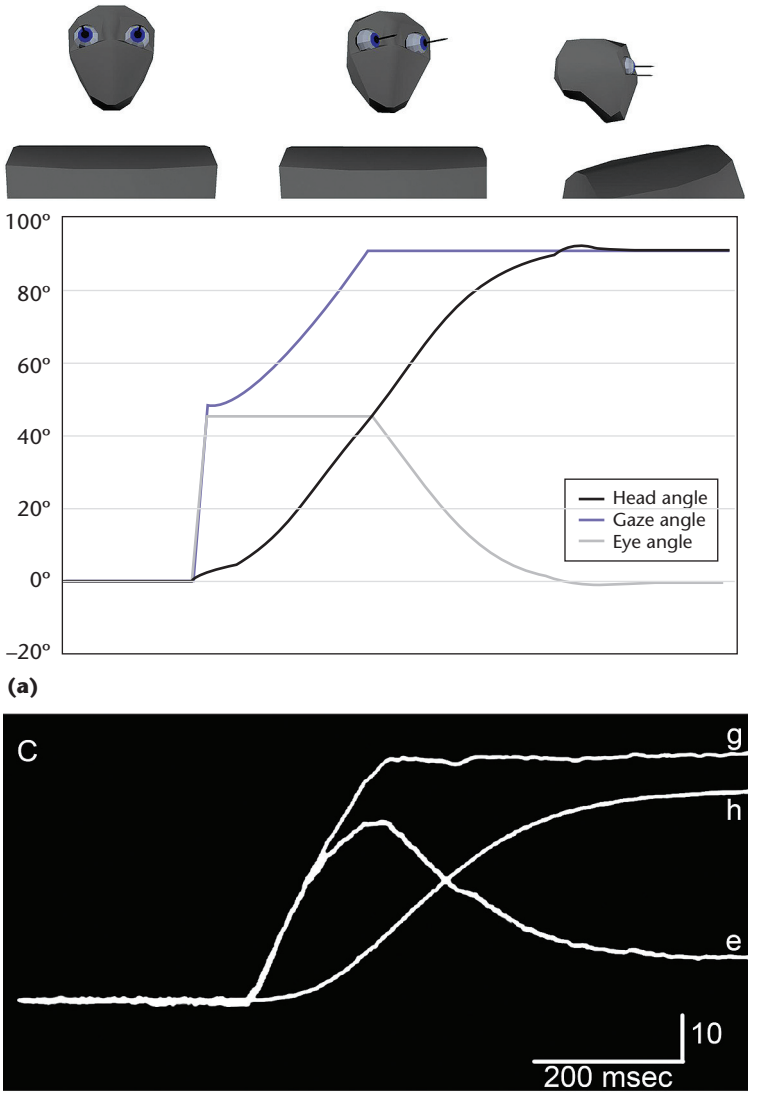

Figure 5. An eye-head saccade beyond the eye motor limit. (a) Motion curves for the character's head, eyes, and combined eye and head gaze as generated by our model. (b) Motion curves for the head (h), eyes (e), and combined eye and head gaze (g) for an actual human.[8] (Figure 5b source: Taylor & Francis Group, used with permission.)

However, we can overcome these limitations by basing the EGM in a robust character animation system.[3]

An Example Application

We could apply our approach to a computer-controlled nonplayer character in an interactive role-playing game. Consider a character with a simple emotion model consisting of two states: neutral and sad. The player interacts with the character through selecting responses in a dialogue tree, which can push the character between emotional states.

The character needs a gaze model that lets the character look at targets in the environment. The character also needs one emotional gaze behavior—specifically, a bowed head—with three collected

Algorithms

```
Input:
  //Motion curves for an emotional gaze shift
  //One curve for each degree of freedom:
    Emotional[DOF][frames],
  //Motion curves for a nonemotional gaze shift
  //One curve for each degree of freedom:
    Neutral[DOF][frames],
Output:
  //Gaze Warping Transformation:
    GWT[DOF][keyframes][c,b];

//Obtain keyframes from both movements
//as described in block 1
emotionKeys[DOF][keyframes][frame,value]=
  keys(Emotional);
neutralKeys[DOF][keyframes][frame,value]=
  keys(Neutral);

//calculate GWT spatial offset parameter "b"
//and assemble GWT from "b" and "c" parameters
FOR EACH DOF:D
  FOR EACH keyframe:i
    IF i==0
      GWT[D][i][c]=1;
    ELSE
      GWT[D][i][c]=(emotionKeys[D][i][frame]-
        emotionKeys[D][i-1][frame])/
        (neutralKeys[D][i][frame]-
        neutralKeys[D][i-1][frame]);
    END IF
    GWT[D][i][b]=emotionKeys[D][i][value]-
      neutralKeys[D][i][value];
  END FOR
END FOR

RETURN GWT
```

Algorithm 1. The Gaze Warping Transformation (GWT) derivation algorithm. Given motion data of two gaze shifts, this algorithm extracts the parameters to convert the keyframes of one of these gaze shifts to the keyframes of the other.

```
Input:
  //GWT Derived from an emotional gaze shift
  //Two parameters (c,b) for each keyframe
  //in each degree of freedom:
    GWT[DOF][keyframes][c,b],
  //Motion curves for a nonemotional gaze shift
  //One curve for each degree of freedom:
    Neutral[DOF][frames],
Output:
  //Motion curves describing head and torso
  //movement for an emotional gaze shift:
    PartialGaze[DOF][frames];

//Obtain keyframes from nonemotional gaze
//As described in block 1
neutralKeys[DOF][keyframes][frame,value]=keys
  (Neutral);

//Apply the GWTs to the keyframes from the
//nonemotional gaze
FOR EACH DOF:D
  FOR EACH keyframe:i
    //Apply GWT scaling parameter "c"
    IF i==0
      GazeKeys[D][i][frame]=1;
    ELSE
    GazeKeys[D][i][frame]=neutralKeys[D][i][frame]
      *GWT[D][i][c];
    END IF
    //Apply GWT spatial offset parameter "b"
    GazeKeys[D][i][value]=neutralKeys[D][i][value]
      +GWT[D][i][b];
  END FOR
END FOR

RETURN PartialGaze
```

Algorithm 2. The GWT application algorithm. Given a GWT and a gaze shift, this algorithm will generate a new gaze shift that displays the behaviors represented in the GWT.

```
Input:
  //Orientation motion curves for the head:
    Head[roll,yaw,pitch][frames],
  //Orientation of the gaze target in head-local
  //rotation coordinates:
    Target[yaw,pitch],
  //Number of frame to begin Saccade:
    saccadeTime;
  //Current Eye Orientation:
    Eye[yaw,pitch];
Output:
  //Eye Rotation motion curves:
    EyeOutput[yaw,pitch][frames];

//Calculate the distance the eyes have
//to rotate from their initial position to
//achieve the target
eyeTarget[yaw]=Target[yaw]-Eye[yaw];
eyeTarget[pitch]=Target[pitch]-Eye[pitch];

//Calculate length of time to perform saccade.
//One frame for each 10 degrees of eye rotation.
saccadeLength=distance(Eye,eyeTarget)/10;
saccadeEnd=saccadeTime+saccadeLength;

//Produce the output motion curves
FOR EACH frame:t
  //if the frame is before the saccade
  //keep eye gazing at original target
  IF t<saccadeTime
    EyeOutput[yaw][t]=Eye[yaw];
    EyeOutput[pitch][t]=Eye[pitch];

  //if the frame is after the saccade
  //keep eye gazing at final target
  ELSE IF t>=saccadeTime+saccadeLength
    EyeOutput[yaw][t]=eyeTarget[yaw];
    EyeOutput[pitch][t]=eyeTarget[pitch];

  //if the frame is during the saccade, linearly
  //interpolate between (saccade beginning,eye
  //position) and (saccade end,target)
  ELSE
    EyeOutput[yaw][t]=
      lerp((saccadeTime,Eye[yaw]),
           (saccadeEnd,eyeTarget[yaw]),
            t-saccadeTime);
    EyeOutput[pitch][t]=
      lerp((saccadeTime,Eye[pitch]),
           (saccadeEnd,eyeTarget[pitch]),
            t-saccadeTime);
  END IF
END FOR

RETURN EyeOutput
```

Algorithm 3. The saccade generation algorithm. Given the initial orientation, gaze target orientation, and time to begin the gaze shift, this algorithm generates a saccade to look from the initial position to the target.

```
Input:
  //Orientation motion curves for the head:
    Head[roll,pitch,yaw][frames],
  //Current Eye Orientation:
    Eye[yaw,pitch];
Output:
  //Eye Rotation motion curves:
    EyeOutput[yaw,pitch][frames];

//initialize Output
EyeOutput[yaw][0]=Eye[yaw];
EyeOutput[pitch][0]=Eye[pitch];

//Rotate the eye in the opposite direction of the
//head, keeping the eye gazing at the same target
FOR EACH frame:t
  IF t>0
    EyeOutput[yaw][t]=EyeOutput[yaw][t-1]-
      (Head[yaw][t]-Head[yaw][t-1]);
    EyeOutput[pitch][t]=EyeOutput[pitch][t-1]-
      (Head[pitch][t]-Head[pitch][t-1]);
  END IF
END FOR

RETURN EyeOutput
```

Algorithm 4. The VOR generation algorithm. Given animation curves for head movement and initial eye orientation, this algorithm generates an eye movement curve displaying the VOR.

```
Input:
  //Orientation motion curves for the head:
    Head[roll,pitch,yaw][frames],
  //Orientation of the gaze target in head-local
  //rotation coordinates:
    Target[yaw,pitch],
  //Current Eye Orientation:
    Eye[yaw,pitch];
Output:
  //Eye Rotation motion curves:
    EyeOutput[yaw,pitch][frames];

//Calculate the distance the eyes have
//to rotate from their initial position to
//achieve the target
eyeTarget[yaw]=Target[yaw]-Eye[yaw];
eyeTarget[pitch]=Target[pitch]-Eye[pitch];

//Calculate length of time to perform saccade.
saccadeLength=distance(Eye,eyeTarget)/10;

//Calculate frame number when the head has rotated
//1 degree from initial position, to begin the saccade
WHILE HeadRotation<1
  currentFrame+=1;
  HeadRotation=abs(distance(
    Head[currentFrame],Head[0]));
END WHILE
saccadeBegin=currentFrame;
saccadeEnd=saccadeBegin+saccadeLength;
currentFrame=saccadeEnd;

//Use VOR to keep eyes on initial target before
//saccade begins
EyeOutput[yaw,pitch][0 to saccadeBegin]=
  VOR(Head[][0 to saccadeBegin],Eye);

//generate saccade
EyeOutput[yaw,pitch][saccadeBegin to saccadeEnd]=
  saccade(Head[][saccadeBegin to saccadeEnd],
          Target,saccadeBegin,Eye);

//After the saccade, if the eyes are not on target
//then keep the eyes at the maximum. The head
//will rotate until the eyes are on target.
WHILE EyeOutput[currentFrame]!=eyeTarget
  EyeOutput[yaw][currentFrame]=45;
  EyeOutput[pitch][currentFrame]=45;
  //Update the distance the eyes have to rotate
  //to achieve the target
  eyeTarget[yaw]=(Target[yaw]-
    Head[yaw][currentFrame])-
    EyeOutput[yaw][currentFrame-1];
  eyeTarget[pitch]=(Target[pitch]-
    Head[pitch][currentFrame])-
    EyeOutput[pitch][currentFrame-1];
  currentFrame++;
END WHILE
onTarget=currentFrame;

//Use VOR to keep eyes on final target until
//Head finishes movement
EyeOutput[yaw,pitch][onTarget to lastFrame]=
  VOR(Head[][onTarget to lastFrame],
    Eye[][onTarget]);

RETURN EyeOutput
```

Algorithm 5. The combined eye-head saccade generation algorithm. Given the animation curves for head movement, initial eye orientation, and gaze target orientation, this algorithm generates eye movement for a combined eye-head saccade.

Algorithms (cont.)

```
Input:
  //Head orientation before the gaze shift:
    Head[yaw,pitch],
  //Eye orientation before the gaze shift:
    Eye[yaw,pitch],
  //Target orientation, in relation to the head
    Target[yaw,pitch],
  //Keyframes for straight-ahead gaze
    Ahead[DOF][keyframes][frame,value],
  //Gaze action: Saccade, VOR, or combined
  //Head/eye Gaze
    Action;
Output:
  //Motion curves for a gaze shift, including
  //head, torso, and eyes:
    Gaze[DOF][frames];

//determine how far the head has to rotate
Rotation[pitch]=Target[pitch]-Head[pitch];
Rotation[yaw]=Target[yaw]-Head[yaw];

//Select the nearest GWTs from the GWT library
PitchGWT=nearestNeighbor(Rotation[pitch],
                         GWTPitchLib);
YawGWT=nearestNeighbor(Rotation[yaw],GWTYawLib);

//Determine how much the GWTs have to be scaled
//by calculating the ratio between what the head
//needs to rotate to achieve the target, and how
//much the GWT will actually rotate the head
PitchScale=PitchGWT[HeadPitch][lastKeyframe]/
  Rotation[Pitch];
YawScale=YawGWT[HeadYaw][lastKeyframe]/
  Rotation[Yaw];

//Scale spatial offset parameter "b" in all
//degrees of freedom in the GWTs
FOR EACH DOF:D
  FOR EACH keyframe:i
    pitchGWT[D][i][b]*=PitchScale;
    yawGWT[D][i][b]*=YawScale;
  END FOR
END FOR

//Apply the GWTs to the straight-ahead gaze
GazeKeys=ApplyGWT(pitchGWT,Ahead);
GazeKeys=ApplyGWT(pitchGWT,GazeKeys);

//Interpolate keyframes using a cubic spline
PartialGaze=cubicInterpolate(GazeKeys);

//Add saccadic, VOR, or combined head/eye movement
//to the partial gaze shift based on Action input
Gaze=EyeModel(PartialGaze,Eye,Target,Action);

RETURN Gaze
```

Algorithm 6. The gaze generation algorithm. Given the initial orientation and gaze target orientation, this algorithm generates an appropriate gaze shift to the target.

```
Input:
  //Motion curves for the gaze shift to obtain
  //keyframes from. One curve of length frames for
  //each degree of freedom:
    Gaze[DOF][frames];
  //Stereotypical Keyframe Alignment Values. One
  //alignment value for each keyframe:
    SKAV[keyframes];
Output:
  //Keyframes of the motion curve. One
  //(framenumber,value) pair for each keyframe for
  //each degree of freedom
    Keys[DOF][keyframes][key,value]

//Extract the head rotation curves from the gaze
Head[roll,pitch,yaw][frames]=
Gaze[HeadRoll,HeadPitch,HeadYaw][frames];

FOR EACH frame:t
  FOR EACH of roll,pitch,yaw:D
    //Calculate framewise velocity
    Vf[D][t]=(Head[D][t-1]+Head[D][t+1])/2;
  END FOR
  //Calculate overall velocity magnitude per frame
  V[t] = sqrt(square(Vf[roll][t])+
    square(Vf[pitch][t])+square(Vf[yaw][t]));
END FOR

//Calculate Denominator
Denom = sum(V[1 to length(V)]);

//Calculate Frame Alignment Values based on
//formula above
FOR each frame:t
  FAV[t]=sum(V[1 to t])/Denom;
END FOR

FOR EACH keyframe:i
  //Find frame number with the alignment
  //value closest to SKAV[i]
  FrameNumber[i]=nearestNeighbor(FAV,SKAV[i]);
END FOR

FOR EACH DOF:D
  FOR EACH keyframe:i
    Keyframes[D][i][frame]=FrameNumber[i];
    Keyframes[D][i][value]=
      Gaze[D][FrameNumber[i]]
  END FOR
END FOR

RETURN Keyframes;
```

Algorithm 7. The keyframe selection algorithm. Given the motion curves representing a gaze shift, this algorithm obtains the keyframes for that gaze shift.

GWTs for that behavior. The first bows the head during a gaze shift, the second keeps the head bowed during the gaze shift, and the third raises the head back to neutral.

Suppose as the player moves through the dialog tree, he or she selects a response that pushes the character into the "sad" state. The next gaze shift uses the first GWT to bow the character's head while the character gazes. As the player continues interacting with the character, the character's head continues to be bowed. If new players approach, the character also gazes at them with a bowed head.

When the user selects a response that pushes the character back to the neutral emotional state, the third GWT raises the character's head back to neutral on the next gaze shift. While the character

is in the neutral state, no GWTs are applied to the character's further gaze shifts.

For additional information on applying the EGM to interactive virtual humans, see "Real-Time Expressive Gaze Animation for Virtual Humans."[3]

We still have much work to do on the EGM. We'd like to integrate it with eye shape and other facial-expression components, such as those that Stephen Platt and Norman Badler described.[11] We'd also like to integrate it with a model based on expressing emotion through the gaze direction, such as Atsushi Fukayama and his colleagues described.[12] The EGM would also benefit from incorporating additional gaze movements such as the vergence movements that move the eyes independently to focus on a target, and from incorporating control over the character's pupil size.

Of course, gaze isn't the only signifier of emotion. Besides other aspects of the character's nonverbal behavior, contextual factors such as the environment, the character model, the relationship between individual characters or between characters and the user, and even the relative position between characters can affect the gaze's expressive content. We explored the EGM in a very decontextualized virtual environment, and it's unclear how these contextual factors affect expressive gaze.

Finally, we'll continue to use the EGM to further explore the composition of low-level gaze behaviors into expressive gaze shifts.

Acknowledgments

We thank Bosco Tjan, Karen Liu, Skip Rizzo, W. Lewis Johnson, Mei Si, Sharon Carnicke, Marcus Thiebaux, Andrew Marshall, Jina Lee, and Tiffany Cole for their invaluable assistance in this research. This research was supported partly by an appointment to the US Army Research Laboratory Postdoctoral Fellowship Program administered by Oak Ridge Associated Universities through a contract with the US Army Research Laboratory, and partly by the US Army Research, Development, and Engineering Command. The content doesn't necessarily reflect the US government's position or policy, and no official endorsement should be inferred.

References

1. F. Thomas and O. Johnston, *Disney Animation: The Illusion of Life*, Abbeville Press, 1981.
2. B. Lance and S.C. Marsella, "Emotionally Expressive Head and Body Movement during Gaze Shifts," *Intelligent Virtual Agents*, LNCS 4722, 2007, pp. 72–85.
3. M. Thiebaux, B. Lance, and S.C. Marsella, "Real-Time Expressive Gaze Animation for Virtual Humans," *Proc. 8th Int'l Conf. Autonomous Agents and Multiagent Systems* (AAMAS 09), vol. 1, Int'l Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 321–328.
4. A. Witkin and Z. Popovic, "Motion Warping," *Proc. Siggraph*, ACM Press, 1995, pp. 105–108.
5. K. Amaya, A. Bruderlin, and T. Calvert, "Emotion from Motion," *Proc. Conf. Graphics Interface 1996*, Canadian Information Processing Soc., 1996, pp. 222–229.
6. B. Lance and S.C. Marsella, "A Model of Gaze for the Purpose of Emotional Expression in Virtual Embodied Agents," *Proc. Autonomous Agents and Multi-agent Systems* (AAMAS 08), Int'l Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 199–206.
7. R.J. Leigh and D.S. Zee, *The Neurology of Eye Movements*, Oxford Univ. Press, 2006.
8. T. Uemura, Y. Arai, and C. Shimazaki, "Eye-Head Coordination during Lateral Gaze in Normal Subjects," *Acta Oto-Laryngologica*, vol. 90, no. 1, 1980, pp. 191–198.
9. B. Lance and S.C. Marsella, "Glances, Glares, and Glowering: How Should a Virtual Human Express Emotion through Gaze?" *J. Autonomous Agents and Multi-agent Systems*, vol. 20, no. 1, 2009, pp. 50–69.
10. M.A.O. Vasilescu, "Human Motion Signatures: Analysis, Synthesis, Recognition," *Proc. 2002 Int'l Conf. Pattern Recognition* (ICPR 02), IEEE CS Press, 2002, pp. 456–460.
11. S.M. Platt and N.I. Badler, "Animating Facial Expressions," *ACM Siggraph Computer Graphics*, vol. 15, no. 3, 1981, p. 252.
12. A. Fukayama et al., "Messages Embedded in Gaze of Interface Agents—Impression Management with Agent's Gaze," *Proc. 2002 SIGCHI Conf. Human Factors in Computing Systems* (SIGCHI 02), ACM Press, 2002, pp. 41–48.

Brent J. Lance *is a computer scientist at the Army Research Laboratory at the Aberdeen Proving Grounds. His research interests are virtual reality and affective computing. Lance has a PhD in computer science from the University of Southern California. Contact him at blance@ict.usc.edu.*

Stacy C. Marsella *is a research associate professor of computer science at the University of Southern California and the Associate Director for Social Simulation Research and codirector for the Computational Emotion Group, both at the USC Institute for Creative Technologies. He heads projects on virtual humans, social simulation, emotions, and theory of mind. Marsella has a PhD in computer science from Rutgers University. Contact him at marsella@ict.usc.edu; www.ict.usc.edu/~marsella.*