A Lightweight Algorithm for Procedural Generation of Emotionally Affected Behavior and Appearance

Yathirajan Manavalan\textsuperscript{1} and Vadim Bulitko\textsuperscript{1} and Marcia Spetch\textsuperscript{2}

Departments of Computing Science\textsuperscript{1} and Psychology\textsuperscript{2}
University of Alberta
Edmonton, Alberta, T6G 2E8, CANADA
{yathi.bm|bulitko|mspetch}@ualberta.ca

Abstract

Displaying believable emotional reactions in virtual characters is required in applications ranging from virtual-reality trainers to video games. Manual scripting is the most frequently used method and enables an arbitrarily high fidelity of the emotions displayed. However, scripting is labor intensive and greatly reduces the scope of emotions displayed and emotionally affected behavior in virtual characters. As a result, only a few virtual characters can display believable emotions and only in pre-scripted encounters. In this paper, we implement and evaluate a lightweight algorithm for procedurally controlling both emotionally affected behavior and emotional appearance of a virtual character. The algorithm is based on two psychological models of emotions: conservation of resources and appraisal. The former component controls emotionally affected behavior of a virtual character whereas the latter generates explicit numeric descriptors of the character’s emotions which can be used to drive the character’s appearance. We implement the algorithm in a simple testbed and compare it to two baseline approaches via a user study. Human participants judged the emotions displayed by the algorithm to be more believable than those of the baselines.

1 Introduction

Emotionally affected behavior of virtual characters in video games and training environments can significantly impact the user’s experience. In movies and books, characters are often remembered by their emotional reactions. We expect James Bond to be confident and suave, Master Yoda to be mysterious and wise, and Sherlock Holmes to be calm and rational. Similarly, the witty and narcissistic antagonist GLaDOS from the Portal video game series (Valve 2007) stays with the player long after the game is over. The emotional reveal by Pagan Min at the climax of Far Cry 4 (Ubisoft 2014) brings a resolution to the player’s struggle during the forty-some-hour campaign. The cool and collected acknowledgment “Moving to position” of the non-playable special-force companions in Tom Clancy’s Ghost Recon 2 (Ubisoft 2004) becomes the strained “Move accomplished!” as the game’s combat heats up. And how can one forget the gleeful “Yest, Kapitan!” of a battle station officer acknowledging a torpedo launch aboard a Russian attack submarine in Sub Command (Strategy First 2001)?

Similarly, virtual reality trainers have long attempted to incorporate emotionally believable characters in their training simulations. In the early nineties, the US Navy IDCTT trainer used live acting recordings to create the emotionally charged atmosphere of damage control aboard a ship under attack (Johnson 1994). In the mid two thousands, an Iraqi police captain Farid, a non-playable character in a virtual reality trainer by the Institute for Creative Technologies, would be offended by the player’s inappropriate chitchat (Solomon et al. 2008).

As the graphical fidelity of virtual characters has increased substantially, the gap between their realistic appearance and unrealistic behaviors has widened. This is in part due to two trends currently seen in the video game industry. First, modern video games are moving towards giving the player more agency. While most commercial video games still do not allow the player to free-type like Façade (Mateas and Stern 2003) or use their voice like The Restaurant Game (Orkin and Roy 2007), more options are becoming available (e.g., the interaction wheel of the Mass Effect series). Second, even the traditionally story-oriented studios such as BioWare are making their games more open-world (Campbell 2014) where the player is actively encouraged to roam massive worlds and interact with hundreds of AI-controlled virtual characters in them (Pramath 2015).

Combined, the two trends result in a progressively larger number of AI-controlled non-playable characters (NPCs) the player can interact with and a larger number of ways to do so. As a result, even with tens of millions of dollars in development budgets, it is impossible to hand-script emotionally affected reactions and appearances of each character in response to each possible way the player may interact with them. Thus, a handful of characters may be fully scripted for the specific ways the player is allowed to interact with them whereas hundreds of others will deliver generic one-line reactions to actions the player takes toward them. The discrepancy breaks immersion and reminds the player that she is playing a game. For instance, Amita, a key story character in Far Cry 4, becomes a generic in-world NPC after the single-player campaign ends and the game transitions to an endless open-world simulator. Previously
talkative and emotional, Amita now idly walks around the
world, cycling through generic animations, mute and oblivi-
ous of her prior interactions with the player. While merely
immersion-breaking in a video game, such lack of realism
may be intolerable in a VR trainer (Traum et al. 2003).

We propose to address the content bottleneck by procedu-
really generating both emotionally affected actions and emo-
tional appearance descriptors of computer-controlled char-
acters in video games and virtual reality trainers. Proce-
dural generation is a promising approach with advances
made in level generation (Valchyanov and Brown 2012),
NPC population (Booth 2009) and gameplay difficulty ad-
justment (Hunicke and Chapman 2004). Furthermore, work
exists on visualizing emotions via procedurally animated fa-
cial expressions (Nvidia 2013) and body language (Desai
and Szafron 2012). In this paper we implement and evaluate
a recent algorithm that can procedurally generate emotion-
ally charged NPC actions and explicit emotional descriptors
for visualization. While procedural models of emotional re-
actions have been explored over the past few years, this al-
gorithm is intentionally lightweight for an easy integration
in video games and trainers, especially running on mobile
hardware and consoles.

The rest of the paper is organized as follows. We formu-
late the problem in Section 2 and follow with a discussion of
related work in Section 3. We present details of our imple-
mentation of the algorithm in Section 4. We then present the
results from a user study comparing the algorithm against
two baseline models in Section 5. We discuss the limitations
of the algorithm and possible future work Section 6.

2 Problem Formulation

Our problem is to procedurally generate NPC emotions that
can be used to drive the NPC appearance and behavior. The
criterion of success is whether such displays of emotions are
deemed believable by a human observer. As more and more
games are moving towards the open-world model with hun-
dreds to thousands of AI-controlled characters, an ideal so-
lution to the problem will be computationally lightweight,
allowing a number of NPCs to be driven simultaneously
on modest hardware such as game consoles. Additionally,
the procedural methods should minimize character author-
ing time and, ideally, be accessible to a non-programmer.

3 Related Work

There have been three primary approaches to achieving
emotionally believable artificial characters in video games
and other settings. The first is manual scripting, where
the designers script the actions, facial expressions and dialogue
of a virtual character for every possible in-game interac-
ction. As a result, the designers have complete control on
how the virtual character is going to behave and emote in
each scenario. This can yield realistic movie-like experience
but requires a substantial amount of time and effort on the
part of the designers. Consequently, most video games and
virtual training environments necessarily limit the number
of characters the player/trainee can interact with and/or the
scope of the interactions available. For instance, BioWare’s

Mass Effect series (Bioware 2007) constrains the player’s
actions with a dialogue wheel and quicktime events and
only fleshes out the primary characters in specific encoun-
ters. Other characters tend to utter generic one-liners and
behave less believably when the player attempts to interact
with them. Even the primary characters exhibit emotionally
unbelievable behavior beyond pre-scripted encounters: the
implausibly repetitive “Wrex. Shepard.” dialogue from Mass
Effect has become an Internet meme (Linguica 2008).

The second approach replaces hand-scripted emotional
NPC responses with a procedural model that takes in the con-
text and the player’s actions and outputs a numerical de-
scriptor of the NPC emotions. A few representatives of this
approach use an appraisal model of emotions which postu-
lates that emotions arise from the character’s appraisal of
the current state with respect to the character’s goals (Scherer,
Schorr, and Johnstone 2001). For instance, when achieving
a desirable goal is uncertain the character may experience
hope. A looming undesirable state elicits fear and so on. A
well known computational implementation of this approach
is EMotion and Adaptation (EMA) (Gratch and Marsella
2001; 2004a; 2004b; Marsella and Gratch 2009) which can
be expensive to run and daunting for game developers to im-
plement at its full scale. A lightweight subset of EMA has
been proposed (Bulitko et al. 2008) but it can only produce
emotional descriptors (e.g., the amount of fear an NPC is
feeling) and not the related coping actions.

The third approach forgoes procedural generation of emo-
tion descriptors and instead generates emotion-related ac-
tions. For instance, an NPC fearing for its life may run
away, causing the player to attribute the emotion of fear to
it (Isla 2005). The psychological theory of resource con-
servation (Hobfoll 1989) suggests that character actions
stem from protecting and gaining resources that the charac-
ter deems valuable. Computational implementations of this
approach (e.g., COR-E (Campano et al. 2013) and Thes-
pian (Si, Marsella, and Pynadath 2005)) have been pro-
posed. While computationally lightweight and less daunting
to embed into a game, COR-E does not generate explict
emotional descriptors and therefore is unable to drive
facial (MOVA 2009; Nvidia 2013) and non-facial (Desai
and Szafron 2012) NPC emotional displays or modulate the NPC
voice (Black and Lenzo 2003).

4 Implementing ACORE

In this paper we adopt a recent combination of COR-E and
EMA known as ACORE (Manavalan and Bulitko 2014).
Similarly to COR-E, ACORE uses a representation of re-
sources valuable to the NPC. This allows ACORE to gener-
ate actions that the player can interpret as the NPC’s displays
of emotions (e.g., fleeing from danger may be attributed to
fear). Like CEMA and unlike COR-E, ACORE also uses an
appraisal model of emotions to compute explicit emotion de-
scriptors. For instance, if a valuable resource is threatened,
ACORE will generate a numeric value for the emotion of
fear which can be used to drive the NPC appearance and
even color a display of the protective action.

To make this paper more self-contained, we will now al-
gorithmically describe ACORE. In-game non-playable char-
characters, NPCs, are controlled by Algorithm 1. Lines 1 through 3 initialize the NPCs’ data. Then, as long as the game is not over (line 5), we model each NPC’s action selection by invoking the resource conservation module ACOREaction (line 7, detailed in Section 4.2). We then compute NPC’s emotional descriptors by invoking the appraisal module ACOREemotion (line 8, detailed in Section 4.3). Resource values are updated using the actions of this and other NPCs (line 9). Emotion descriptor values decay over time (line 10).

Algorithm 1: ACORE

\[
\text{for each NPC do} \\
\quad \text{initialize: resources } \tilde{r}_i, \text{ resource weights } \bar{w}_i \\
\quad \text{initialize } \tilde{e}_t = (0, 0, 0, 0) \\
\quad t \leftarrow 1 \\
\text{while game running do} \\
\quad \text{for each NPC do} \\
\quad \quad \text{act: } a_t \leftarrow \text{ACORE}_{\text{action}}(\tilde{r}_t, \bar{w}_t, A_t) \\
\quad \quad \text{display emotions: } \tilde{e}_{t+1} \leftarrow \text{ACORE}_{\text{emotion}}(\tilde{r}_t, \bar{w}_t, \tilde{e}_t) \\
\quad \quad \text{update resources: } \tilde{r}_{t+1} \text{ from } \tilde{r}_t, a_t \text{ and other NPCs’ actions} \\
\quad \quad \text{decay the emotion values: } \tilde{e}_{t+1} \leftarrow \tilde{e}_{t+1}/2 \\
\quad t \leftarrow t + 1 
\]

4.1 Resources

ACORE associates a set of valuable resources (e.g., health, reputation, gold) with each NPC. Mathematically, each NPC has a vector of resources, which at time \( t \) is denoted by \( \tilde{r}_t = (r^1_t, \ldots, r^N_t) \) where each \( r^i_t \in \mathbb{R} \) is a scalar representing the value of \( i \)-th resource at time \( t \). The NPC assigns different importance or value to each resource and this is represented by an \( N \)-dimensional weight vector \( \bar{w}_t = (w^1_t, \ldots, w^N_t) \in (0, 1]^N \). A higher weight indicates higher importance of the resource to the NPC. Generally speaking, the weights may change over time but remain in \((0, 1]\).

To illustrate we will walk through the algorithm on a concrete example. Consider an NPC standing in line of three other NPCs, waiting to purchase a product (Figure 2). Each of the NPCs has three resources: health, reputation and rank, which is inversely proportional to the NPC’s place in line. Assume that at some time \( t \), the resources of our NPC are \( \tilde{r}_t = (1, 1, 0.5) \). Suppose the NPC values the health resource at \( 0.26 \), the reputation resource at \( 0.17 \) and the rank resource at \( 0.44 \): \( \bar{w}_t = (0.26, 0.17, 0.44) \). We will now run the NPC step-by-step through ACORE.

4.2 Computing NPC Actions

Each NPC has a set of actions (called behaviors in COR-E) available to it at time \( t \), denoted by \( A_t \). The actions can be primitive or complex, consisting of a control policy. Each action can affect the resources the NPC holds (e.g., getting into a fight may negatively affect the NPC’s health). NPCs select actions that are expected to increase their cumulative weighted resource value (lines 2 and 3 of Algorithm 2).

The value of an action is the sum of the resource value deltas weighted by the resource weights, conditional on the action. In our example, an NPC standing in line has two actions available for it: \( A_t = \{ a^\text{pass}, a^\text{wait} \} \). By taking the \( a^\text{pass} \) action, the NPC will attempt to pass the NPC standing in front of it, thus improving its rank in the line but possibly losing reputation or even health (if the NPC being passed physically protests the pass). Suppose that by passing the NPC expects its resources to change from \( \tilde{r}_t = (1, 1, 0.5) \) to \( E[\tilde{r}_{t+1}|a^\text{pass}] = (0.95, 0.6, 1) \). Weighting the expected delta \( E[\tilde{r}_{t+1}|a^\text{pass}] - \tilde{r}_t \) by \( \bar{w}_t = (0.26, 0.17, 0.44) \) the NPC computes the value of passing as \( V(a^\text{pass}) = 0.139 \) (line 2).

An alternative action is to wait in line which does not immediately change the resource vector. Hence \( V(a^\text{wait}) \approx 0 \). Then, in line 3, the NPC will select the action with the maximum value: \( a_t = a^\text{pass} \). Note that a different set of resource weights (e.g., \( \bar{w}_t = (0.5, 0.5, 0) \) for an NPC that cares about its health and reputation but not about its place in the line) would have made the value of the passing action negative and led to the NPC selecting the wait action instead.

4.3 Computing NPC Emotions

ACORE simplifies the emotion model of EMA (Gratch and Marsella 2001; 2004a) by assuming that each NPC has only a single goal: to improve its weighted resource values. The appraisal process is thus limited to considering changes in the resource vector. In line with CEMA (Bulitko et al. 2008), ACORE models only four emotions: hope, joy, fear and distress. At time \( t \), the NPC represents its emotional state as a four-dimensional vector \( \tilde{e}_t = (e^\text{hope}_t, e^\text{joy}_t, e^\text{stress}_t, e^\text{fear}_t) \) where each \( e^i_t \) represents the intensity of the corresponding emotion. The initial emotion vector \( \tilde{e}_1 = (0, 0, 0, 0) \), set in line 3 of Algorithm 1.

The intensity of each emotion with respect to a resource is computed as the product of the NPC’s desirability \( \alpha \) of the change in the resource caused by the action selected by ACOREaction (Section 4.2) and the certainty of the change \( \beta \). The desirability is the product of the resource weight and the expected change in the value of a resource caused by the action (line 3, Algorithm 3). The certainty of the change is the probability of the change in the direction specified by \( \alpha \) (lines 5 and 7). A desirability of zero leads to no changes to the agent’s emotions.

The total intensity of each emotion is the sum of its intensity for each resource \( i \) (the loop in line 2). Which emotion gets the update depends on the relation between the desirability and certainty. Desirable but uncertain changes in a resource contribute to hope (lines 10 and 11), desirable and certain changes contribute to joy (lines 8 and 9). Likewise, undesirable resource changes contribute to fear (when they...
are uncertain, lines 14 and 15) or distress (when they are certain, lines 12 and 13).

**Algorithm 3: ACORE\textsuperscript{emotion}**

```
inputs : current resources \(\hat{r}_t\), resource weights \(\hat{w}_t\), selected action \(a_t\), current emotion \(e_t\)
output : emotion \(e_{t+1}\) = \((e_{t+1}^{\text{joy}}, e_{t+1}^{\text{hope}}, e_{t+1}^{\text{distress}}, e_{t+1}^{\text{fear}})\)
1 start with the current emotion \(e_t\)
2 for resource index \(i = 1, \ldots, N\) do
3     compute desirability: \(\alpha = w_t^i \cdot (E[r_{t+1}^i | a_t] - r_t^i]\)
4     if \(\alpha > 0\) then
5         compute certainty: \(\beta = Pr(r_{t+1}^i > r_t^i | a_t)\)
6     else
7         compute certainty: \(\beta = 1 - Pr(r_{t+1}^i > r_t^i | a_t)\)
8     if \(\alpha > 0 \&\ & \beta = 1\) then
9         compute joy: \(e_{t+1}^{\text{joy}} = e_t^{\text{joy}} + \alpha\)
10        else if \(\alpha > 0 \&\ & \beta < 1\) then
11            compute joy: \(e_{t+1}^{\text{joy}} = e_t^{\text{joy}} + \alpha \cdot \beta\)
12            if \(\alpha < 0 \&\ & \beta = 1\) then
13                compute distress: \(e_{t+1}^{\text{distress}} = e_t^{\text{distress}} - \alpha\)
14            else if \(\alpha < 0 \&\ & \beta < 1\) then
15                compute fear: \(e_{t+1}^{\text{fear}} = e_t^{\text{fear}} - \alpha \cdot \beta\)
```

We will now illustrate the operation of ACORE\textsuperscript{emotion} with specific numbers, continuing our example from the previous section. Starting with the current emotion values \(e_t\) (line 1), our NPC computes an update to its new emotional state \(e_{t+1}\) for each resource in the loop in line 2. Suppose the current value of the NPC’s health is \(r_t^{\text{health}} = 1\). Under the action \(a_t = \text{pass}\) selected by ACORE\textsuperscript{action} in Section 4.2, the health resource is expected to decrease \((E[r_{t+1}^{\text{health}} | a_{\text{pass}}] = 0.95)\) due to a possible physical opposition from the NPC being passed. The decrease in health in undesirable because health is positively weighted \((w = 0.26).\) Thus, \(\alpha = w \cdot (E[r_{t+1}^{\text{health}} | a_{\text{pass}}] - r_t^{\text{health}}) = 0.26 \cdot (0.95 - 1) = -0.013.\)

The certainty \(\beta\) of the undesirable health change is then computed as \(Pr(r_{t+1}^{\text{health}} < r_t^{\text{health}} | a_{\text{pass}})\) (line 7) which, generally speaking, depends on the action the NPC being passed will take. We model this in two stages. At the first stage we assume that our NPC has not yet observed the actions of other NPCs and thus uses a prior for the certainty values. We compute the resulting changes to the emotion model which can then be visualized with the NPC’s appearance. In stage two, the NPC has observed the actions of the affected NPCs and updates its emotion model accordingly. The emotional appearance can then be visualized once again.

For the health resource before the NPC is able to observe actions of the NPC being passed, it uses the prior \(\beta = 0.95.\)

With this value, the conditions in line 14 will be satisfied and the intensity of emotion fear will be updated as \(e_{t+1}^{\text{fear}} = e_t^{\text{fear}} - \alpha \cdot \beta = e_t^{\text{fear}} - (-0.013) \cdot 0.95 = e_t^{\text{fear}} + 0.0123\) in line 15. In other words, our NPC is now slightly more afraid of the prospects of losing some of its health.

The for loop in Algorithm 3 will then consider the next resource, reputation. The reputation is also predicted to decrease under the pass action: \(E[r_{t+1}^{\text{reputation}} | a_{\text{pass}}] = 0.6\) because cutting in front of other people in line is disreputable thing to do. Our NPC cares about its reputation \((w = 0.17)\) and so \(\alpha = w \cdot (E[r_{t+1}^{\text{reputation}} | a_{\text{pass}}] - r_t^{\text{reputation}}) = 0.17 \cdot (0.6 - 1) = -0.068.\) The prior probability of this undesirable loss of reputation is \(\beta = 0.95\) which adds an extra \(- \alpha \cdot \beta = (-(-0.068)) \cdot 0.95 = 0.0646\) to the emotion of fear.

The final resource is the rank in line whose current value is \(r_t^{\text{rank}} = 0.5\) (i.e., the NPC is second in line). Under the pass action, the new value is expected to be \(E[r_{t+1}^{\text{rank}} | a_{\text{pass}}] = 1.\) Our NPC cares about its rank \((w = 0.44)\) which means that \(\alpha = w \cdot (E[r_{t+1}^{\text{rank}} | a_{\text{pass}}] - r_t^{\text{rank}}) = 0.44 \cdot (1 - 0.5) = 0.22.\)

The prior probability of this desirable gain of rank is \(\beta = 0.95\) which adds \(\alpha \cdot \beta = 0.22 \cdot 0.95 = 0.209\) to the emotion of hope (lines 10 and 11). Thus, at the end of the first stage, the NPC’s fear is increased by \(0.0123 + 0.0646 = 0.0769\) and the hope is increased by \(0.209.\) The new emotion vector is then visualized via the NPC appearance.

Suppose then the NPC being passed protests the pass. Thus, in stage two, the passing NPC observes the protest and updates its \(\alpha\) and \(\beta\) values. Retrieving the values from Table 1, ACORE computes the new increase of fear as \(- \alpha \cdot \beta = (-(-0.013)) \cdot 0.5 = 0.0065.\)

In stage two, the change in the reputation resource remains the same leading to \(\alpha = 0.17.\) However, since the NPC is now aware of its pass action being protested, the expectation that the reputation will decrease is reduced to \(\beta = 0.5.\) Thus, the stage-two increase in fear due to possible change in reputation is \(- \alpha \cdot \beta = (-(-0.068)) \cdot 0.5 = 0.034.\)

For the rank resource, the expected value of the resource change remains the same as in stage one which leads to \(\alpha = 0.22.\) The certainty is now reduced to \(\beta = 0.5.\) Thus at stage two the NPC’s hope is increased by \(\alpha \cdot \beta = 0.22 \cdot 0.5 = 0.11.\)

Thus, at the end of the second stage, the NPC’s the NPC’s fear is increased by \(0.0065 + 0.034 = 0.0405\) and its hope is increased by \(0.11.\) The new emotion vector is visualized. It should be noted that the \(\beta\) values for certainty of change in Table 1 are for the moment in time the actions are taken. Once the outcome of an action is known the \(\beta\) of a change can be 1. For instance, once an NPC has successfully passed another NPC, it will experience joy since its position in line (the rank resource) has improved.

5 Evaluation

We evaluated our implementation of ACORE with a user study in which we compared ACORE with COR-E and a random emotion model as baselines.

5.1 Experimental Testbed

Adapting the scenario from COR-E studies (Campano et al. 2013) we considered a line of six people waiting to purchase a video game (Figure 2). Each person in line is represented by an NPC controlled by ACORE and has three resources: health, reputation and rank. Each person has two basic actions available to them at each time step: pass the person in
### Table 1: Domain dynamics.

| Resource $r_t$ | Action $a_t$ | Expected resource change $E[r_{t+1}|a_t] - r_t$ | The other NPC’s action | Certainty of change $\beta$ |
|---------------|-------------|---------------------------------|----------------------|------------------|
| health        | pass        | $-0.05$                         | wait                 | 0.95             |
| reputation    | pass        | $-0.4$                          | wait                 | 0.95             |
| rank          | pass        | $1/(1/r_{t}^{rank} - 1) - r_{t}^{rank}$ | wait                 | 0.95             |
| health        | pass        | $-0.05$                         | protest              | 0.5              |
| reputation    | pass        | $-0.4$                          | protest              | 0.5              |
| rank          | pass        | $1/(1/r_{t}^{rank} - 1) - r_{t}^{rank}$ | protest              | 0.5              |
| health        | protest     | $-0.05$                         | pass                 | 0.5              |
| reputation    | protest     | $-0.1$                          | pass                 | 0.5              |
| rank          | protest     | $1/(1/r_{t}^{rank} + 1) - r_{t}^{rank}$ | pass                 | 0.5              |
| health        | wait        | $0$                             | wait                 | 0.95             |
| reputation    | wait        | $0$                             | wait                 | 0.95             |
| rank          | wait        | $0$                             | wait                 | 0.95             |
| health        | wait        | $0$                             | pass                 | 0.5              |
| reputation    | wait        | $0$                             | pass                 | 0.5              |
| rank          | wait        | $1/(1/r_{t}^{rank} + 1) - r_{t}^{rank}$ | pass                 | 0.5              |

front of them (if he/she exists) or wait in line. Additionally, a person being passed can choose to protest the passing action or allow it to happen (i.e., continuing waiting in line).

At every time step, the person at the head of the line purchases the video game and leaves the line. The simulation was stopped when the line becomes empty. Visually, each person in line was represented with a photograph showing their facial expression (only the highest intensity emotion was shown; Figure 3), their name and the three resources. Health was visualized with a bar underneath the image. The reputation was shown by the color of frame around their portrait. The rank was shown by the position of the person in line (Figure 2). Additionally, people in line uttered one-line remarks shown as text above their heads (Table 2).

There were eighteen individuals (Figure 1) from which a line of six was randomly populated (without repetition).

### 5.2 The Experiment

For the user study we recruited 94 participants (30 males, 64 females; mean age 20). For their participation, each participant received a partial credit in an undergraduate psychology course. After a briefing and signing a consent form, each participant was exposed to each of the following conditions.

**Experimental condition** (E) presented a participant with a line of people whose actions and appearances were controlled by the ACORE algorithm.

**Control condition** (C) was identical to E except the appearances were not controlled. Thus each person in line maintained their neutral expression throughout the simula-

Figure 1: Neutral expressions of the 18 individuals whose portraits were used in the user study.

Figure 3: An individual showing hope (top left), joy (top right), fear (bottom left) and distress (bottom right).
Table 2: One-line utterances.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the head of the line</td>
<td>Can I get a copy of Destiny?</td>
</tr>
<tr>
<td>Being passed</td>
<td>Stop!</td>
</tr>
<tr>
<td></td>
<td>Where are you going?</td>
</tr>
<tr>
<td></td>
<td>You shall not pass!</td>
</tr>
<tr>
<td>Having been just passed</td>
<td>Not fair!</td>
</tr>
<tr>
<td></td>
<td>I will get back at you!</td>
</tr>
</tbody>
</table>

The participants found the experimental condition to be more believable than the control or the random condition. Thus, explicitly computing (via an appraisal model) and visualizing (via facial expressions) emotions appears to add to the believability our control condition that approximated the previous algorithm (COR-E).

Interestingly, the participants rated the random condition as more believable than the control condition which appears to suggest that even random facial expressions, unrelated to the actions and utterings of people in the simulation, are better than a constant neutral expression.

6 Current Limitations and Future Work

Our implementation of ACORE used ad hoc hand-coded values for expected resource changes as well as the associated certainty values. They had not been validated and may have had a negative impact on the overall believability of our ACORE implementation. Future work will generate such values procedurally via stochastic simulation.

Second, we used static images for the facial expressions which were photographs of actors acting out the emotions. Future work will investigate the effectiveness of procedurally generated facial expressions (MOVA 2009; Nvidia 2013), driven by ACORE.

Finally, our simulations were non-interactive as the participants watched the simulation unfold on a computer screen. We are currently working on incorporating ACORE into a video game where it will control actions and appearances of NPCs interacting with the player’s avatar.

7 Conclusions

This paper presented the first implementation and evaluation of ACORE, a recent lightweight computational model of emotions. The results of a user study suggest that ACORE can increase believability of AI-controlled virtual characters relative to the previous model, COR-E.

8 Acknowledgments

We would like to gratefully acknowledge the funding provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), the time volunteered by the actors of Die Nasty: The Live Improvised Soap Opera, the valuable feedback provided by members of the IRCL Research Group and the help volunteered by Mathew Fritze in developing the experimental setup.
References


