UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

GAMYGDALA: an Emotion Engine for Games

Popescu, A.; Broekens, J.; van Someren, M.W.

Published in:
IEEE Transactions on Affective Computing

DOI:
10.1109/T-AFFC.2013.24

Link to publication

Citation for published version (APA):
Popescu, A., Broekens, J., & van Someren, M. W. (2014). GAMYGDALA: an Emotion Engine for Games. IEEE Transactions on Affective Computing, 5(1), 32-44. DOI: 10.1109/T-AFFC.2013.24

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (http://dare.uva.nl)

Download date: 18 Sep 2017

GAMYGDALA: An Emotion Engine for Games

Alexandru Popescu, Joost Broekens, and Maarten van Someren

Abstract—In this paper we present GAMYGDALA, an emotional appraisal engine that enables game developers to easily add emotions to their Non-Player Characters (NPC). Our approach proposes a solution that is positioned between event coding of affect, where individual events have predetermined annotated emotional consequences for NPCs, and a full blown cognitive appraisal model. Instead, for an NPC that needs emotions the game developer defines goals and annotates game events with a relation to these goals. Based on this input, GAMYGDALA produces an emotion for that NPC according to the well-known OCC model. In this paper we provide evidence for the following: GAMYGDALA provides black-box Game-AI independent emotion support, is efficient for large numbers of NPCs, and is psychologically grounded.

Index Terms—Computer games, affective computing, psychological model

1 Introduction

Many computer games would be more interesting and entertaining if the Non-Player Characters (NPC) would express emotions and behave emotionally. An NPC in a digital game is a character of which the behavior is controlled by a computer program, not by a player. Often, the player can interact with (e.g., talk to, shoot at, cooperate with) the NPCs in the game. For single player games in particular, NPCs play an important role. Game developers and designers face an important challenge when developing NPC behavior: when a player plays a game for some time, NPC behavior becomes predictable and boring. Making NPC behavior less predictable by introducing "randomness" can be perceived as illogical or inconsistent with the game's plot and internal environment or with the player's assumptions or understanding. Even large productions in genres that benefit from balanced NPC behavior, such as Role-Playing Games like Skyrim [1], struggle with this challenge of balancing between consistency and variation.

Emotional expression and emotional behavior can increase the variation of NPC behavior [2], [3], [4]. However, given a particular state of a game, what is an appropriate emotion to express? In other words, how to calculate plausible emotional states for an NPC? This is the question we address in this paper. We propose a computational model (a specialized AI sub-component, or, emotion engine) that can recognize which emotion fits an NPC at a particular event, in a way that is compatible with current psychological knowledge and with the game's storyline. It is important to be compatible with psychology because this helps to generate emotions that are plausible and understandable. In this paper we propose such an emotion engine. Our engine's workings are grounded in the psychology of emotion but the engine has the simplicity needed to perform efficiently in a game and has been developed to be easy to use by a game developer.

Our emotion engine is called GAMYGDALA, with an eye-wink to the amygdala, the part of the human brain heavily involved in emotion processing. It supports basic and social emotions, enabling game developers to create a wide range of emotional states. It is based on the model of Ortony, Clore and Collins (OCC) [5], which is tried and tested in the field of computational modeling of emotion (e.g. [2], [6], [7]). The novelty of GAMYGDALA is that it is positioned between event coding of affect, where the emotional behavior is coded explicitly for each NPC for each individual event, and a full blown cognitive appraisal model that would include reasoning about events. When using GAMYGDALA, the game developer defines goals for each NPC that needs simulated emotion and annotates game events with how they affect these goals. Based on this input, GAMYGDALA produces in reaction to a game event an emotion that is specific to the goals of each (type of) NPC. Different NPCs can thus react differently to the same situation, and the same situation can elicit different emotions in the same NPC depending on its active goals. Emotional reactions can depend on previous events that changed the goals of an NPC but GAMYGDALA itself does not include memory.

In essence, GAMYGDALA is a "black-box appraisal" engine. GAMYGDALA can be used in a way analogous to physics engines. With little effort and little knowledge of emotional appraisal on the game developer's part, a psychologically plausible simulation of emotion can be included in a game. We argue that this black-box approach is easy to use and generic, like a physics engine. The game can use its own AI and does not need special cognitive processing to benefit from GAMYGDALA. Defining goals and annotating events for use by GAMYGDALA is straightforward, like adding mass and speed properties is also a straightforward process when using a physics engine. Of course knowledge of emotion is still needed if the NPC is to graphcially express it's emotional state.

- *A. Popescu and M. van Someren are with the Informatics Institute, University of Amsterdam, PO Box 94323, Amsterdam, Noord-Holland NL 1090 GH, The Netherlands.*
- *J. Broekens is with Interactive Intelligence Group, TU Delft, Mekelweg 4, Delft, Zuid-Holland NL 2628 CD, The Netherlands.*
  Email: joost.broekens@gmail.com.

Manuscript received 18 Sept. 2012; revised 26 Apr. 2013; 16 Sept. 2013; accepted 25 Sept. 2013.
Recommended for acceptance by G.N. Yannakakis, K. Isbister, A. Paiva, and K. Karpouzis.
For information on obtaining reprints of this article, please send e-mail to: taffc@computer.org, and reference IEEECS Log Number TAFFCSI-2012-09-0087.
Digital Object Identifier no. 10.1109/T-AFFC.2013.24.

1949-3045/14/$31.00 © 2014 IEEE Published by the IEEE Computer Society

IEEE Proof

GAMYGDALA's output, i.e., an NPC's emotional state, can be used in two different ways. First it can be used to *express* emotions, for example the character may look stressed or joyful. The emotional state is visualized by a rendering engine or the voice may change. Second, the emotional state can be used to alter the *behavior* of the character, for example it may run away when scared or start shooting when angry. Now, the emotional state is fed back to the game AI that generates NPC behavior.

GAMYGDALA is designed to be

- *psychologically grounded*: it is based on a tried and tested model of emotion, also with respect to emotion dynamics;
- *modular*: a modular, black-box, game-AI independent appraisal engine, is easy to use and does not require deep knowledge of appraisal theory on the part of the developer or designer; and
- *efficient*: able to run in real-time for large numbers of agents.

2 Related Work

2.1 Emotion Theories

In psychology, there are three main perspectives on emotion [8]. The *discrete* view defines a set of "basic" emotions [9] where each emotion is characterized by feelings, expression and behavior, for example, fear is associated with a negative feeling, a particular facial expression, and fleeing behavior. The *dimensional* view defines an emotion as a point in an *n*-dimensional space (e.g. [10], [11]). There are several different possible dimensions, for example, fear is negative, arousing and submissive. The *componential* view defines an emotion as the result of several evaluative cognitive components, called cognitive appraisals, [5], [12]. For example, fear is the anticipation of something that is incongruent with our goals. As we are interested in simulation of emotion in NPCs, we need to know when and how an emotion results from events in a game and therefore the cognitive appraisal view fits our aim best. We will describe cognitive appraisal in more detail in the following section.

2.2 Cognitive Appraisal Theories

Cognitive appraisal theories assume that an emotion is the result of a cognitive process. A person assesses at all times what is happening in the world around him or her. These events are evaluated in terms of how relevant and conducive these are to that person's and other persons' "goals". The emotion emerges from this process.

Frijda [13] puts at the center of his theory the "concern," which is the disposition of a person to prefer certain states of the world over others. This is related to the concept of "goal" in other theories. This theory does not really deal with if and how opposing emotions cancel each other (for example *joy* and *distress*) or with how multiple emotions combine into an emotional state. Further, it does not attempt to define categories of emotions nor label them using common language.

Scherer [12], [14] formulates a componential theory of emotion that defines a step-based appraisal model. He argues that cognitive appraisal does not happen all at once and that there are steps that influence each other. As a first step, the person evaluates relevance, that is: does the event deserve attention? how unexpected is the event? does it result in pleasure or pain? does it influence the person's goals? Second, the potential implications of the event are evaluated. Third, the person assesses the amount of control over the situation. Can he change the situation? Finally, how does the event relate to the person's standards, social norms and values? Emotion emerges continuously from this appraisal of events.

In the theory proposed by OCC [5] the authors define 22 emotions (later 24) that are associated with a number of evaluative components including desirability for the person self, desirability for others, likelihood or praiseworthiness. They define the listed emotions as being emotion types and reject the idea of basic emotions. According to OCC, appraisal of an event consists of three parts, of which only the first is currently used in GAMYGDALA:

- the desirability and consequences of the event in light of the agent's goals (e.g. candy is bad for maintaining health);
- approval/disapproval according to norms, in case the event is an action (e.g. hitting is bad); and
- liking/disliking of the event, in case the event relates to an object that appears in an event (e.g. I like candy).

We decided to use OCC as basis for our emotion engine for the following reasons: it is a well-known and accepted theory of emotions, it is a componential model of emotion that fits the needs of a computational framework, components are generic enough to allow for a wide set of emotions, it accounts for both internal emotions and social relationships which in games are quite important, and most importantly many computational models have been built on it.

A framework that complements the OCC model is that of Pleasure, Arousal and Dominance (PAD) [10], [11], [15]. *Pleasure* defines positiveness versus negativeness, *Arousal* defines alertness versus calmness and *Dominance* defines the feeling of control. For example, *Happiness* is defined as an emotion with high PAD while *Fear* is an emotion with high arousal but low pleasure and dominance. Based on Mehrabian's work [10] GAMYGDALA can derive PAD values from OCC emotions. The game developer can thus also use emotion dimensions.

2.3 Computational Models of Emotion

Computational models of emotion are usually based on psychological emotion theories and attempt to model the theory such that it is usable by artificial agents [2], [6], [16], [17], [18], [19], [20]. For a recent review see [21]. In this section we highlight several models that have had significant influence. A second reason to review these models is because they provide insight to deal with difficult issues in computational modeling of emotion, such as the relation between emotion and mood, the integration of different emotions, emotion dynamics, and the influence of emotion on behavior. These insights have been used for the development of GAMYGDALA.

ACRES [16] is a computational model based on Frijda's theory of appraisal [13]. It has the same components as Frijda's theory, is built around "concerns" and has the same

IEEE Proof

steps for appraisal. ACRES continuously checks if received information is relevant to its concerns. Then, for the relevant parts, it computes the rest of the components: coping variables, urgency, seriousness, difficulty, and so on. It also has a powerful action plan generator that combines actions from a predefined set to form a variety of possible plans.

The "Affective Reasoner" [6] is a computational model of emotion based on the OCC appraisal theory [5]. It models all of the 24 emotion types defined in OCC. The appraisal process is based on goals, standards and preferences and produces one or more emotions. The system then decides which actions the agent takes, based on the resulting emotions. The system also maintains for each agent a set of representations of other agents, that is, what does agent *X* think agent *Y* is feeling right now. This is a feature used to diversify the response. Emotions elicited at a certain time are not combined in the "Affective Reasoner," as the system is focused on the output actions instead. The action generation step eliminates possible behavior that is incongruent with the emotional state before deciding on an action.

"Em" [2] is a computational model based mostly on the OCC theory. Emotion generation is based on a set of predefined rules, such as "when an agent has a goal failure and the goal has importance X, generate an emotion structure of type distress and with intensity X". An emotion has a type and an intensity. Emotions are stored in a hierarchical structure based on the specificity of the emotion: a generic *distress* emotion will be higher in the hierarchy while more specific ones, such as *grief* will be a subtype of *distress*. This makes it possible to infer highlevel emotions from emotions that are subsumed. At this point each subtree will result in an emotion type with a certain intensity based on emotions in the subtree. Emotions are combined using a set of "Emotion Combination Functions." The intensities decay over time as specified by the "Emotion Decay Functions." A "Behavioral Feature Map" maps the resulting emotions to behavioral features and sends them back to the system. Later [22], Reilly specifies how emotions are stored, how their intensity is determined and how emotions are combined. He describes several interesting features, some of which we borrow in GAMYGDALA:

- *Expectedness*. Emotions are triggered by changes in the likelihoods of events and goals. For example, if an event has a likelihood of 0.9 and then it happens, it will trigger a lower intensity emotion than a totally unexpected event, even if at the latest step, they both have likelihood 1.
- *Bias against failure*. Since humans have higher intensity emotions for failure than for success, he introduces desirability and undesirability as two separate variables for each goal.
- *Different decay per emotion type*. This allows for "hope" and "fear" to be treated differently, slower than others when the event is still uncertain and instantly when the event takes place.
- *Combining emotions*. Em uses a logarithmic function to combine and normalize emotions. This ensures, as opposed to a sigmoidal normalizer function, that accumulating emotional intensity is linear for lower values and less additive as values increase.

EMA is a "computational model of appraisal and coping" [7] inspired by the "Affective Reasoner" [6] and as a consequence by the "OCC" [5] appraisal theory. Appraisal is based on several variables that are components of the appraisal process:

- *Relevance*. How significant is an event for the agent.
- *Desirability*. Does the agent want this to happen or not.
- *Likelihood*. Is it a surprise or an expected event.
- *Causal attribution*. Who is responsible for the event.
- *Controllability*. Does the agent have any control over the event.
- *Changeability*. Does the agent have any power to change the outcome (together with *Controllability* these two variables refer to the ability of coping).

EMA also introduces an interesting frame concept that makes certain emotions shift out of focus with time and come back into focus if a similar event happens or the agent just remembers and thinks about the event that caused it. A "mood" structure is used to aggregate multiple emotions by adding up all current emotions by type and then passing the results through a sigmoid function in order to normalize them. The mood also creates a bias towards certain emotions rather than others. EMA includes a planning mechanism that works closely together with the appraisal to identify the best coping strategy and modify resulting emotions accordingly.

MAMID [23] is a computational model that significantly differs from the other models. It is focused not only on modeling emotions but also on differences between agents (traits). It represents individual differences of three different types: cognitive, affective and personality. Also, it models several influences of affect on cognition, which is an interesting basis for emotional influences on game AI.

We reviewed several influential computational models of emotions. The "Affective Reasoner" appears to be the most complete model built on top of the OCC theory. It implements all possible emotions, including social interaction. For this it includes inside each agent a representation of all the other agents that supports the social interaction by allowing agents to speculate about other agents' feelings and goals. Other models also assume (or use) particular AI techniques, reasoning techniques or cognitive architectures. As one of our goals is to make GAMYGDALA independent of game AI, we do not want to make such assumptions. We are aiming for a standalone model of appraisal that can function independently from AI reasoning and expression rendering because that is the only way we can support pluggability into different games. As such, GAMYGDALA computes emotions from goals and beliefs and includes the decay of emotions but does not include the expression of emotion or predetermined behavioral influences. The mechanism used to define goals and beliefs and to compute an emotional state for an NPC is explained in Section 3.

2.4 Affective Gaming

Simulations of emotions can be incorporated in games, which can benefit from that in several ways. Games could recognize emotions in the human player and adapt to this to increase satisfaction [24], similar to how players

IEEE Proof

appreciate human opponents for their ability to socialize [25]. Here we focus on augmenting NPC behavior with emotions. There are several games that simulate a certain level of human-like emotional behavior [26], such as "Creatures" by Cyberlife Technology [27], [28], "The Sims" series by Electronic Arts or "Black & White" by Lionhead Studios [29]. Up to this date there are no emotion engines that can be used in a variety of games as a simple pluggable black-box. Recently, it has been argued that such an approach would be welcome to "support the development of socially complex and affectively realistic games" [4], [8], [30]. In the following paragraphs we critically review earlier computational models of emotion for NPCs created specifically for games.

Baillie-de Byl [26] creates an example game where the agents emulate emotions to be more believable. The model implemented in this game is based on the work by Smith and Ellsworth [31], which proposes six appraisal dimensions: pleasantness, responsibility, certainty, attention, effort, control. For the output the game uses the basic emotions of Ekman [9]: happiness, sadness, anger, fear, disgust, surprise. The implementation is based on a neural network with the value of each appraisal dimension as input and six emotion flags as output where only one is set to *true*. The result is an agent that cannot have complex emotions and cannot have two emotions at the same time. This might work for simple games but is not enough to build a complex NPC.

Carlisle proposed a simple framework [32] used to create emotional agents for games. This framework implements *personality* using the OCEAN model of personality [33], which has the potential to give each agent a unique personality that will affect the behavior later. As far as *mood* goes, this framework uses a 1D representation along the pleasure-displeasure axis where positive values represent pleasure. The *appraisal* part is built on a simplified version of the OCC model, which does not elicit the specific emotion types defined by OCC. Instead, it focuses on the types of emotions (related to events, agents or objects) and defines a single component for each of them: desirability for events, and attraction for agents and objects. This offers advantages in computation times but reduces the amount of available emotional consequences and behavior. The downside of this approach is that many specific emotions, such as Gratitude or Pity, cannot be generated at all. For most of the games this is enough but possibly more complex social emotions are needed for some games.

Research at the University of Bath [34] aims at a system to create procedural side-quests for role-playing games. Even if it is not based on any established psychological model, it is an important step towards specific socially capable agents. Their model is based on behavior trees that use information from a set of priorities (goals) and memory (beliefs/events) to decide on the next quest to be proposed to the player. The emotional state is not maintained but inferred from goals and beliefs.

Researchers at the University of California at Santa Cruz developed a "social simulation," called Prom Week [35], where agents emulate emotions and social interaction between teenagers during the last week before the Prom. The main focus of this game are relationships, including making and breaking friendships that may or may not include the player. The scenarios have predefined goals and the player wins by achieving the goal before the week is over. It is an interesting project but strongly focused on social networking, and the large set of rules needed for the social simulation makes this approach difficult to use in a generic way for emotion simulation.

All in all there has been great progress in the field but the game industry still lacks an AI independent, easily pluggable emotion engine. A fundamental reason for this seems to be that most attempts integrate the emotional appraisal with the AI reasoning. This limits generic use of the emotion engine, because if the game AI is not compatible with the AI used in the emotion engine, the engine can not be used. Further, simpler approaches support only basic emotions, while complex ones support personality, mood and social reasoning. We think a solution in the middle is needed for game development: support for a reasonable set of emotions (including, e.g., social emotions), but a simple straightforward implementation of appraisal. We aim for a stand-alone emotional appraisal engine that implements most of the OCC emotions with low computational cost so that it is scalable for a large number of agents in real time. Our approach, analogous to a physics engine, should support rapid development of emotions for NPCs. Further, developers should be able to use it without knowledge of appraisal theory.

3 GAMYGDALA: An Emotion Engine for Games

In this section we introduce GAMYGDALA, its assumptions, its mechanisms of appraisal, how it computes the emotion, and what input it needs to do so. GAMYGDALA is based on the OCC model, for reasons mentioned earlier. OCC (and all other appraisal theories) assume that an agent has some form of goals, concerns, or a set of preferred states, and a mechanism to perceive the world and act in that world. A fundamental assumption for being able to use GAMYGDALA in a game is therefore that the game can be viewed as events (see [19] for a discussion), that goals can be defined for the NPCs that need emotion simulation and that the relevance of events for the goals is specified.

Here we argue that GAMYGDALA is indeed easy to use, AI independent and easily pluggable. We will show that in most games goals exist or can be easily defined. In general, each NPC in a game has a purpose, otherwise it would not be in the game or its behavior would be incomprehensible to the player. Each NPC in a game can be viewed as an autonomous agent: within the frame of its purpose and possibility for action, it perceives the game state (input) and chooses alternative behaviors (plans, actions) that are intended to serve its purpose. Its goal is a desired state. An example is a soldier that wants to kill the player. Upon killing the player, the end state has been reached. An NPC can have multiple goals. For example a soldier that wants to stop the player from entering a building can have the goals *player distance is large* and *player is killed*. Note that it does not really matter if the game itself uses goals or reasoning with these goals to generate

IEEE Proof

behavior. The goals are defined for the emotion engine. The only thing that is then needed is to annotate (some) game events with goal congruence (how good or bad is this event for which goal). This is the essence of how GAMYGDALA works and supports game AI independence.

To use GAMYGDALA, a game designer needs to explicitly define one or more goals for each NPC (or NPC type, when all NPCs of a type have the same goals). The game designer can now define which (type of) events are relevant to each goal, and specify this in GAMYGDALA. The only essential modification to a game is that events delivered to the game AI controlling the NPC's behavior are also delivered to GAMYGDALA. GAMYGDALA emotionally appraises the event and outputs the most plausible emotion(s) based on the OCC model. This way of using an emotion engine is analogous to how physics engines are used. This approach is AI independent. Nowhere a link to the AI system is needed for emotion simulation, even though some AI mechanisms might interface more naturally with GAMYGDALA including BDI-based goal-oriented reasoning and planning.

GAMYGDALA is a richer model for generating emotions than event coding. In event coding, each event needs to include an emotional consequence for each of the NPCs involved. The advantage of using GAMYGDALA is that the game designer gets plausible emotions without having to design them and without having to specify them for each event. Yet GAMYGDALA avoids the computational costs of a full-blown emotional AI engine that also includes reasoning and planning. AI specific functionality such as decision making and action selection is not included in GAMYGDALA. Also, emotion expression functionality is not included, as this is related to specifics of the game format, style, story or graphics. This means that GAMYGDALA keeps track of the goals of each character as well as their beliefs representing events happening in the game. Further, it appraises the current situation, manages the affective state and associated dynamics. GAMYGDALA uses a componential representation of emotion based on OCC that it can translate to a dimensional representation based on Pleasure-Arousal-Dominance upon request. [10], [11], [15]. A game engine can thus obtain from GAMYGDALA both emotion intensities (e.g. "fear level") and affect ("degree of dominance").

3.1 Game World

The term "Game World" refers to the virtual world where the game takes place and in which the player (gamer) is immersed while playing. A game consists of a sequence of events where a new event is caused by the user or the game engine. The game world is usually populated with NPCs. For example, in the classic game "Pac-Man" [36], the game world is a labyrinth, the player is represented by PacMan, a character that has to survive while collecting all dots in the labyrinth. The ghosts, running around the labyrinth and trying to block pacman from finishing the level are the NPCs of this world. GAMYGDALA finds emotions for the NPCs, in this case the ghost. The game developer can use this to add expression or emotional behavior to the ghost. We now specify GAMYDALA's input in detail.

TABLE 1
Examples of Defined Goals

Owner	Goal Name	Utility	Explanation
knight	kill monster	1	the knight wants to kill the monster
knight	self die	-1	the knight wants to preserve his own life (not die)
knight	princess die	-1	the knight wants to keep the princess alive
knight	find gold	0.5	the knight wants to find treasures

3.2 Goals

Goals in GAMYGDALA can be states of the game that an NPC wants achieved (active pursuit and passive goals) or states that an NPC wants maintained (maintenance goals). The system does not differentiate between active pursuit and passive goals, as it is a black-box oblivious to the NPC's plans and OCC does not really differentiate between the two goal types in terms of different emotion outcome. Each goal is represented using the same structure:

- *name*: identifier;
- *owner*: the NPC that has this as a goal; and
- *utility*: the value the NPC attributes to this goal becoming true ($\in [-1, 1]$ where a negative value means the NPC does not want this to happen).

Some examples of goals are shown in Table 1. The value of the utility is proportional to the level of desire for that specific goal. Utility values of 1 or -1 should be used for major goals, for example, the outcome of the game. Everything in between is less important. For example, finding gold is not as important as staying alive, so failure to gather gold will not generate the same level of distress for the NPC as having his own life threatened. A goal is achieved if it is true in the current event of the game.

3.3 Beliefs

Beliefs are annotated events. Each emotionally relevant event is annotated with:

- *name*: identifier used to define and update a belief;
- *likelihood*: the likelihood that this information is true ($\in [0, 1]$) where 0 means the belief is disconfirmed and 1 means it is confirmed;
- *causal agent*: the agent responsible for this event (can be an NPC, the player or empty if it is an "Act of God" or irrelevant); and
- *affected goals*: a table of goal identifiers with congruences between goals and beliefs. Congruence is a number $\in [-1, 1]$ where negative values mean this belief is blocking the goal and positive values mean this belief facilitates the goal.

The event likelihood gives the game developer the possibility to implement concepts such as rumors (event was not witnessed but the NPC has heard of it) or credibility (the NPC does not entirely believe the information because it was delivered by another unknown NPC). When a new

IEEE Proof

TABLE 2
Examples of Incoming Beliefs

Belief Name	princess attacked by monster	found magic sword
Likelihood	0.5	1
Causal Agent	monster	knight
Affected Goals	princess die	kill monster
Congruence	0.8	0.5

event occurs it produces a corresponding belief for the NPC's involved.

3.4 Emotions

Emotion simulation is based on the OCC model of appraisal [5]. Currently 16 of the 24 OCC emotions are supported. GAMYGDALA defines two types of emotion: internal emotions and social emotions (emotions that are directed at another NPC). Each NPC p has one internal emotional state and for every other NPC it knows a social emotional state (e.g., angry at q_1, happy for q_2). Before we detail the appraisal mechanism, we first explain belief desirability and goal likelihood.

3.4.1 Desirability

The desirability of a belief b for a goal g depends on the utility of g and the congruence between b and g:

$$desirability(b, g, p) \leftarrow congruence(b, g) * utility(g). \quad (1)$$

In the example above, the desirability for the "*princess attacked by monster*" event, as far as the *knight* is concerned, would be $0.8 * (-1) = -0.8$ so for this NPC it is quite an undesirable event. If we also define the "*kill princess*" goal for the monster but with a utility 1 then the system would also calculate the desirability for the monster $0.8 * 1 = 0.8$ for which the event is very desirable. This shows how the same belief can have a very different desirability for two different NPCs due to different goal utilities.

3.4.2 Goal Likelihood

The likelihood of the goal is maintained internally and updated every time when a belief that affects that specific goal is received. The initial value is *Unknown*. The goal likelihood is defined by:

$$likelihood(g) \leftarrow (congruence(b, g) * likelihood(b) + 1)/2. \quad (2)$$

Here $likelihood(g)$ is the likelihood that the goal g is realized after observing belief b, with $congruence(b, g)$ and $likelihood(b)$ as defined above. In the previous example, the likelihood of the "*princess die*" goal will be $(0.8 * 0.5 + 1)/2 = 0.7$. Every time the goal likelihood is updated, the old value is saved into an internal variable so we are able to also know the *change* in goal likelihood for later calculations. This difference will be referred to as $\Delta likelihood(g)$.

3.4.3 Internal Emotions

This contains all emotions that do not relate to other NPCs. These are:

TABLE 3
Internal Emotions Appraisal Mechanism

Emotion	Eliciting condition
hope	$(des(b, g, self) > 0, L(g) < 1, \Delta L(g) > 0) \vee$ $(des(b, g, self) < 0, L(g) > 0, \Delta L(g) < 0)$
fear	$(des(b, g, self) < 0, L(g) < 1, \Delta L(g) > 0) \vee$ $(des(b, g, self) > 0, L(g) > 0, \Delta L(g) < 0)$
joy	$(des(b, g, self) > 0, L(g) = 1) \vee$ $(des(b, g, self) < 0, L(g) = 0)$
distress	$(des(b, g, self) < 0, L(g) = 1) \vee$ $(des(b, g, self) > 0, L(g) = 0)$
satisfaction	$des(b, g, self) > 0, L(g) = 1, \lvert\Delta L(g)\rvert < 0.5$
fearsConfirmed	$des(b, g, self) < 0, L(g) = 1, \lvert\Delta L(g)\rvert < 0.5$
disappointment	$des(b, g, self) > 0, L(g) = 0, \lvert\Delta L(g)\rvert > 0.5$
relief	$des(b, g, self) < 0, L(b) = 0, \lvert\Delta L(g)\rvert > 0.5$

- *hope*: a desirable uncertain goal increases in likelihood of success or an undesirable uncertain goal decreases in likelihood of success;
- *fear*: an undesirable uncertain goal increases in likelihood of success or a desirable uncertain goal decreases in likelihood of success;
- *joy*: a desirable goal succeeds or an undesirable goal fails;
- *distress*: an undesirable goal succeeds or a desirable goal fails;
- *satisfaction*: a desirable goal was expected to succeed and it actually does succeed;
- *fearsConfirmed*: an undesirable goal was expected to succeed and it actually does succeed;
- *disappointment*: a desirable goal was expected to succeed and it fails; and
- *relief*: an undesirable goal was expected to succeed and it fails.

For emotions such as *fear* we need the notion of uncertainty [5]. We consider a goal or belief to be uncertain if it has a likelihood between 0 and 1 exclusive. For example, if the likelihood of an undesirable event is 0.99 then *fear* will have a high intensity. If the likelihood becomes 1, *distress* and *fearsConfirmed* are generated. Additionally, for confirmation-based emotions like *disappointment* and *relief* we need to define when a situation is reversed or not. Currently reversal is defined as an arbitrary threshold for $\Delta likelihood(g)$. If $|\Delta likelihood(g)| > 0.5$ we assume the situation has reversed.

Table 3 shows the eliciting conditions for each emotion, where L is the likelihood and des is desirability. When GAMYGDALA's appraisal process is invoked, for each emotion meeting it's eliciting condition the intensity is computed as:

$$intensity(e) \leftarrow |des(b, g, self) * \Delta L(g)|. \quad (3)$$

3.4.4 Social Emotions

Social emotions have another NPC as target, although the target can also be the NPC self, the human player or characters controlled by the player. They relate to an event in which one NPC influences the other NPC's goals. These emotions are:

- *anger*: an undesirable event is caused by another NPC;
- *guilt*: this NPC causes an undesirable event for a liked NPC;

IEEE Proof

TABLE 4
Social Emotions Appraisal Mechanism

Emotion	Eliciting condition
anger	$des(b,g,self)<0, NPC(b)\neq self$
guilt	$des(b,g,q\neq self)<0, NPC(b)=self, like(self,q)>0$
gratitude	$des(b,g,self)>0, NPC(b)\neq self$
gratif.	$des(b,g,q\neq self)>0, NPC(b)=self, like(self,q)>0$
happyFor	$des(b,g,q\neq self)>0, like(self,q)>0$
pity	$des(b,g,q\neq self)<0, like(self,q)>0$
gloating	$des(b,g,q\neq self)<0, like(self,q)<0$
resentm.	$des(b,g,q\neq self)>0, like(self,q)<0$

- *gratitude*: a desirable event is caused by another NPC
- *gratification*: this NPC causes a desirable event for a liked NPC;
- *happyFor*: a desirable event happens to a liked NPC;
- *pity*: an undesirable event happens to a liked NPC;
- *gloating*: an undesirable event happens to a disliked NPC; and
- *resentment*: a desirable event happens to a disliked NPC;
- *like*: this is calculated based on all events caused by that specific NPC (it is neutral = 0 if unknown).

Importantly, the terms *desirable* and *undesirable* always refer to the desirability calculated from the goal owner's point of view. Table 4 shows the eliciting conditions for each emotion. When this condition is met, the intensity is computed as:

$$intensity(e) \leftarrow |des(b,g,q\neq self) * \Delta L(g) * like(self,q)|, \quad (4)$$

except for *anger* and *gratitude* that follow the intensity calculation as defined for internal emotions.

$NPC(b)$ is the NPC that caused belief b and $like(p,q)$ is how much NPC p likes NPC q. Of course p and q can also be Player Controlled characters. The $like(p,q)$ relationship is influenced by the desirability of events caused by q as seen through the eyes of p. An event caused by q that facilitates a desirable goal for p, increases the value of $like(p,q)$. Likewise, if the goal is blocked then the value of $like(p,q)$ decreases. As such, relationships grow over time, and do not need to be symmetrical. Consequently, social emotions (except anger and gratitude) exist only when a relationship exists, i.e., $|like(p,q)|>0$. This relation is configured, by setting $like(p,q)$ to a value, or develops over time as explained above. The benefit of this is that social emotions are tractable, the downside is that actions detrimental to q do not trigger empathy in p, unless there exists a relation between p and q, such that $|like(p,q)|>0$.

TABLE 5
The Octants of Temperament Space in PAD [15]

Temperament	Pleasure	Arousal	Dominance
Exuberant	+	+	+
Bored	-	-	-
Dependent	+	+	-
Disdainful	-	-	+
Relaxed	+	-	+
Anxious	-	+	-
Docile	+	-	-
Hostile	-	+	+

TABLE 6
The Axes of Temperament Space in PAD []

Axes	Extremes	Formula
Exuberance	Exuberant vs. Bored	$(P+A+D)$
Dependency	Dependent vs. Disdainful	$(P+A-D)$
Relaxation	Relaxed vs. Anxious	$(P-A+D)$
Docility	Docile vs. Hostile	$(P-A-D)$

3.4.5 Pleasure-Arousal-Dominance

In addition to the OCC model, GAMYGDALA computes the values for PAD. These can be derived from the OCC emotions following [10] and are multiplied by the intensity of that emotion. All OCC emotions are combined into an overall PAD value using the formula:

$$PAD \leftarrow 0.1 \times log_2\left(\sum_e 2^{10\times PAD(e)\times intensity(e)}\right), \quad (5)$$

where $PAD(e)$ is the PAD score for each specific emotion, $intensity(e)$ is the intensity of each emotion and PAD is the PAD representation of the emotional state.

We also implemented four temperamental axes as defined in [15], (see Table 6).

Each negative extreme is the opposite of its positive counterpart, for example, *bored* is $-(P+A+D)=(-P-A-D)$. The weight that is assigned to each axis is $\sqrt{3}$, so for example *exuberance* is defined as $\sqrt{3}*\frac{P+A+D}{3}=0.577*(P+A+D)$. The implementation in our case is a property of the NPC that returns the PAD representation of the NPC's emotional state on demand. Of course one is free to use any combination of P, A and D from this output should not all three be needed.

3.4.6 Emotion Management and Dynamics

Each NPC is associated with:

- a list of *goals* the NPC knows about, owned by anyone;
- an array of real values representing the *default emotional state* initialized when the NPC is created;
- an array of real values representing the current *emotional state;*
- an array of real values representing the *default social stance* initialized when the NPC is created;
- an array of real values representing the current *social stance* to each known NPC; and
- (optional) an *emotional decay function* and a *social stance decay function.*

The *default emotional state*, the *default social stance* and the *decay functions* are used by the game designer to create individual differences in emotion dynamics between NPCs. They represent an NPC's personality by defining that NPC's default (social) emotion (e.g., an angry guy) and how the intensity of all emotions decay. The NPC's *emotional state* is initialized to the *default emotional state*, it changes due to events that affect it and decays back towards the *default emotional state*. A similar thing happens to the *social emotions*: when a new NPC is encountered the *social stance* is set to the *default social stance*. Events change the *social stance* as explained previously, and it decays to

IEEE Proof

Fig. 1. Screenshot from the RPG implementation.

Fig. 2. Screenshot from the RTS implementation.

the NPC's *default social stance*. Appraisals related to internal emotions are combined into the agent's current emotional state using Reilly's logarithmic function. Appraisals related to social emotions are combined into the agent's current social stance towards particular other NPCs. For details we refer to equation (5) from [22]. This has the desirable properties of not being strictly additive, using all emotions (and not only the strongest), and being at least as intense as the most intense component. GAMYGDALA thus closely follows the OCC and PAD models in computing the emotional state with as little additional assumptions as possible.

4 Examples and Evaluation

In this section we illustrate how GAMYGDALA is used in combination with games and game AI. The goal is to illustrate how it can be integrated into games. The examples are a simple arcade game, a role playing game (RPG), a real-time strategy game (RTS), and a first-person shooter (FPS). We selected these four cases to represent different game genres with different types of NPC involvement [26], [37], [38]. For these cases we also explain how emotions generated by GAMYGDALA could be used to modify game AI that controls the NPCs. To support our efficiency and scalability claim, we present a large-scale multi-NPC simulation.

4.1 Arcade Game: Pac-Man (1980)

Pac-Man enjoyed a huge success when it was released. The game generated endless sequels, spin-offs and clones and is a representative game for action arcade games. Pac-Man is the NPC handled by the player who has to collect all dots in a labyrinth but stay away from the enemies (ghosts) who try to eat him and thus hinder achieving the goal. There are also some *power dots* that give Pac-Man the ability to "eat" the enemies and send them back to their base location. This case is a basic example of how straightforward it is to instrument an arcade game with simple emotions using GAMYGDALA, analogous to the approach shown in [19].

The goals of the ghost are modeled as *"get eaten"* with utility -1 and *"catch pacman"* with utility 0.6 (we assume that being eaten is of higher importance than catching Pac-Man). If a state occurs in which Pac-Man gets close to the ghost then this creates a corresponding belief that affects the likelihood that the goal *"catch pacman"* will be achieved. Now, if Pac-Man gets close to the ghost this will increase the likelihood that this goal will be satisfied and this causes the emotion *hope* in the ghost. Proximity of a powered-up Pac-Man also increases the likelihood of the goal *"get eaten,"* resulting in a second emotion *fear*.

These emotions can now be used for expression or they may affect the behavior of the ghost. This example illustrates that GAMYGDALA can be incorporated into Pac-Man by associating goals with the NPC "ghost" and adding information on which states have an effect on achieving the goal or increase the likelihood. GAMYGDALA will then infer the appropriate emotions.

4.2 Role-Playing Game

Classical RPGs involve a storyline with quests in which the player character can interact with various friendly characters but also fight or compete against enemies. RPGs have the potential for an infinite variety of stories and situations, including NPC character variation and thus the use of GAMYGDALA as a way to vary the emotional reactions of NPCs due to game events. We implemented a simple RPG consisting of "the village" where people are neutral to our character, "the hero," and could grow to like him or not, and "the forest" where the character fights enemies. The story is that the village is surrounded by monsters and the people are too scared to go outside. This case serves to show how complex emotions can evolve over time, and how social emotions can be modeled.

We give two examples of in-game situations that produce complex emotions and we will explain the mechanics behind them.

Example 1. Relief. *A villager is relieved when a dangerous situation disappears*

Relief is a two-step emotion. First an event or belief causes Fear (in this case that the village is surrounded) and when

IEEE Proof

the cause of this disappears the new event or belief will bring *Relief* with an intensity proportional to that of *Fear*. Listing 1 shows the trace of the system.

Listing 1. Output for example "Relief"

```
adding goal: village destroyed, utility = -0.9,
    owner = self
adding belief: village surrounded, likelihood =
    0.6

Decaying...
Updating...
Recalculating...
recalculating belief: village surrounded
affected goal: village destroyed, valence = 1
desirability: -0.9 <- 1 x -0.9
goal likelihood: 0.8 <- (1 x 0.6 + 1) / 2
delta goal likelihood: 0.8 <- 0.8
emotion intensity: 0.72 <- abs(-0.9 x 0.8)
adding FEAR: 0.72

adding belief: village surrounded, likelihood = 0
Decaying...
Updating...
Recalculating...
recalculating belief: village surrounded
affected goal: village destroyed, valence = 1
desirability: -0.9 <- 1 x -0.9
goal likelihood: 0.5 <- (1 x 0 + 1) / 2
delta goal likelihood: -0.3 <- 0.5 - 0.8
emotion intensity: 0.27 <- abs(-0.9 x -0.3)
adding JOY: 0.27
adding RELIEF: 0.27

Current Internal State: Joy: 0.27; Fear: 0.36;
    Relief: 0.27
```

Example 2. Pride (Gratification). *The blacksmith was proud of saving the village by providing it with weapons.* This example illustrates events that cause a social emotion.

Listing 2. Code for example "Pride"

```
EmoBrain brain = new EmoBrain();
brain.Goal("to live", 0.7f);
brain.Belief("provides house", 1f, "village");
brain.AffectsGoal("provides house", "live", 1f);
brain.Update();
brain.Goal("village destroyed", -1f, "village");
brain.Belief("village is unarmed", 0.7f);
brain.AffectsGoal("village is unarmed", "village
    destroyed", 1f);
brain.Update();
brain.Belief("provide weapons", 1f, "self");
brain.AffectsGoal("provide weapons", "village
    destroyed", -1);
brain.Update();
```

We first "create" an initial relationship between the blacksmith and the village caused by the blacksmith's need to live somewhere and the village providing that place. This makes the blacksmith like the village. Alternatively, one could simply configure this relationship, or let it grow in a more natural manner over time. After the first step, *Joy* is generated because the blacksmith is pleased to live in a house and *Gratitude* towards the village because it is the responsible agent. The second step brings the belief that the village is in grave danger because it is lacking the weapons needed to defend itself. This step triggers *Pity* as something bad is happening to an NPC that the blacksmith likes, i.e., the village. As a consequence, when the blacksmith is able to help the village by providing the necessary weapons, two new emotions appear: *HappyFor* as a liked NPC has something good happening to it and *Gratification* as the blacksmith feels proud of helping an NPC that he likes. As an additional remark, if we were to model the village with its own emotional brain at the same time, then after the second step it would generate *Fear* and at the end *Relief*, *Joy* and *Gratitude* towards the blacksmith. This example illustrates the complex emotional relationships between NPCs that can be achieved with GAMYGDALA, where an NPC could be a character or a more abstract agent, such as a village.

Listing 3. Output for example "Pride"

```
adding goal: to live, utility = 0.7, owner = self
adding belief: provides house, likelihood = 1,
    agent = village

Decaying...
Updating...
Recalculating...
recalculating belief: provides house
affected goal: to live, valence = 1
desirability: 0.7 <- 1 x 0.7
goal likelihood: 1 <- (1 x 1 + 1) / 2
delta goal likelihood: 1 <- 1
emotion intensity: 0.7 <- abs(0.7 x 1)
adding JOY: 0.7
adding GRATITUDE: 0.7 towards village
adding goal: village destroyed, utility = -1,
    owner = village
adding belief: village is unarmed, likelihood =
    0.7

Decaying...
Updating...
Recalculating...
recalculating belief: village is unarmed
affected goal: village destroyed, valence = 1
desirability: -1 <- 1 x -1
goal likelihood: 0.85 <- (1 x 0.7 + 1) / 2
delta goal likelihood: 0.85 <- 0.85
emotion intensity: 0.85 <- abs(-1 x 0.85)
adding PITY: 0.85 towards village
adding belief: provide weapons, likelihood = 1,
    agent = self

Decaying...
Updating...
Recalculating...
recalculating belief: provide weapons
affected goal: village destroyed, valence = -1
desirability: 1 <- -1 x -1
goal likelihood: 0 <- (-1 x 1 + 1) / 2
delta goal likelihood: -0.85 <- 0 - 0.85
emotion intensity: 0.85 <- abs(1 x -0.85)
adding GRATIFICATION: 0.85 towards self
adding HAPPY-FOR: 0.85 towards village

Current Internal State: Joy: 0.18
Relationships:
        village -> Gratitude: 0.18; Gratification
            : 0.85; HappyFor: 0.85; Pity: 0.43
```

To illustrate how such an emotional instrumentation can be integrated in the gameplay of an RPG, consider the following. Let's say there is a quest in which one of the villagers, Clark, tells a story about a ring he lost outside of the

IEEE Proof

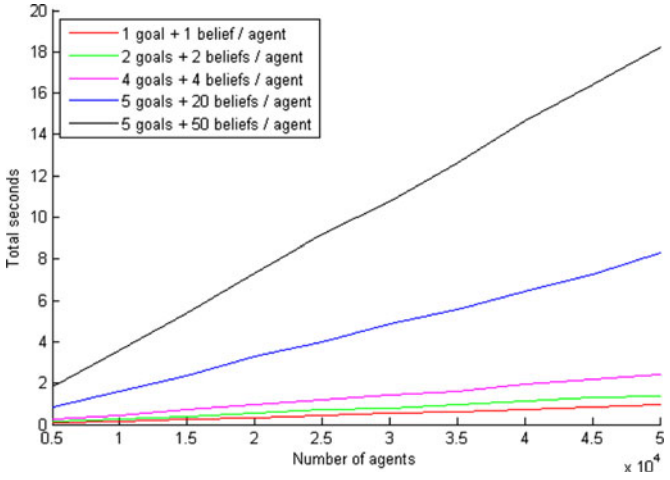

Fig. 3. Timing various populations of NPCs.

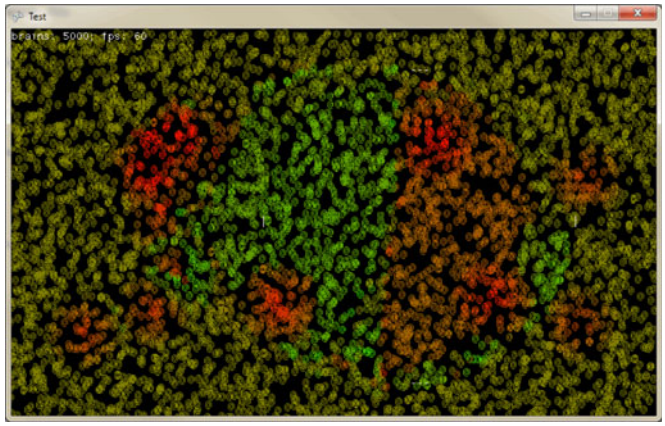

Fig. 4. Performance test on a population of NPCs. Negative pleasure in red, positive pleasure in green.

village (NPCs tend to loose there jewels quite often). Clark has an initial goal *have_item* with utility 1. When the player accepts the quest, belief *accept_quest* (congruency *have_item*= 0.2; likelihood = 1) is added, giving Clark *Hope* and *Gratitude*. When the player actually returns the item, belief *receive_item* (congruency *have_item* = 1; likelihood = 1) is added, giving Clark feelings of *Joy* and *Satisfaction*. If Clark has a wife, Alice, and assuming Alice likes Clark, Alice will now feel *Happy for* Clark and *Gratitude* towards the player. Concerning emotional influences on the storyline, the interaction with NPCs can easily include the emotion in, for example, the dialog trees, a special case of behavior trees [39]. Emotions can be used to select the path through the dialog tree or even to alter certain sentences. For example, when Alice feels *Gratitude* towards the player, this can be a condition to insert an extra line such as "I am so grateful you helped my partner," or even open up new storyline options such as "Now that you helped us, I trust you, would you do me a favor....." Of course this is all possible using other mechanisms, but the interesting thing is that the player can now choose in what way he/she will try to gain enough gratitude from Alice, without having to script all these possibilities explicitly.

4.3 Real-Time Strategy Game

A real-time strategy game usually contains high level management of units and resources. The player is the commander of an army and gives orders to his troops. We developed a simple RTS in which two factions (*Human* and *Orc*) each have three types of buildings and two types of units that can be trained using one of the buildings. When a building is destroyed, it stays as ruins and becomes nonfunctional. Ruins can be rebuilt using a unit but doing so costs gold and takes a few seconds. This case aims to show how emotions generated by a simple GAMYGDALA instrumentation can be used to impact RTS NPC behavior in a psychologically plausible way. In our example, 1) individual fleeing units might trigger mass fleeing of all AI-controlled units through the generation of fear, and, 2) high arousal produces errors in planning.

The initial setup is easy. All AI-controlled units (NPC) initially have the following two goals: *die* with utility -1 and *win* with utility 1, assuming winning and not dying are of equal importance. No initial beliefs are generated and no differences in the default emotional state. While playing, beliefs are added to the NPCs in the game, such as the belief *wounded* affecting goal (die) with likelihood $1 - health$ and conduciveness 1. This generates *Fear* when units are wounded. Furthermore, if NPCs start fleeing, the belief *fleeing* is added to all NPCs affecting the goal *win* negatively with a likelihood proportional to the number of fleeing units. When a player-controlled building is destroyed, the *win* goal is affected positively by a belief *enemy buildings down* that has a probability equal to the proportion of destroyed player-controlled buildings. All of these beliefs generate emotions such as *Fear*, *Hope*, and even *Anger* towards the player and his/her units if they would also be specified as NPCs.

Listing 4. Output for the RTS game example

```
adding goal: die, utility = -1, owner = self
adding goal: win, utility = 1, owner = self
adding belief: wounded, likelihood = 0.7

Decaying...
Updating...
Recalculating...
recalculating belief: wounded
affected goal: die, valence = 1
desirability: -1 <- 1 x -1
goal likelihood: 0.85 <- (1 x 0.7 + 1) / 2
delta goal likelihood: 0.85 <- 0.85
emotion intensity: 0.85 <- abs(-1 x 0.85)
adding FEAR: 0.85
adding belief: enemy buildings down, likelihood =
    0.5

Decaying...
Updating...
Recalculating...
recalculating belief: enemy buildings down
affected goal: win, valence = 1
desirability: 1 <- 1 x 1
goal likelihood: 0.75 <- (1 x 0.5 + 1) / 2
delta goal likelihood: 0.5625 <- 0.75 - 0.1875
emotion intensity: 0.5625 <- abs(1 x 0.5625)
adding HOPE: 0.5625

Current Internal State: Hope: 0.56; Fear: 0.69
```

NPC emotions in the RTS example still need to be used in the game. To influence the NPC behavior based on the

IEEE Proof

emotions we propose two methods. First, we modified the Finite State Machine used to control the unit. Finite State Machines [37], [40] are simple structures used for controlling behavior and are very popular in games because of their implementation simplicity, scalability, and ability to generate complex behaviors. Our initial implementation has only two states: *Idle* and *Attack*. To show the effects of emotions we added a third one, *Flee*. The transition was very simple: when the amount of *Fear* would go above a threshold, the unit would enter the *Flee* state.

The second method is to modify the A* pathfinding algorithm [41] so that troops would lose precision when the emotions were running high. For this case we used the *Arousal* component of the PAD structure. In essence we added a random factor to the weight of each node in the path, so that aroused NPCs will have more erratic behavior (see next case for more detail).

Listing 5. Output for the FPS game example

```
adding goal: die, utility = -1, owner = self
adding goal: kill player, utility = 1, owner =
    self
adding belief: wounded, likelihood = 0.7, agent
     player

Decaying...
Updating...
Recalculating...
recalculating belief: wounded
affected goal: die, valence = 1
desirability: -1 <- 1 x -1
goal likelihood: 0.85 <- (1 x 0.7 + 1) / 2
delta goal likelihood: 0.85 <- 0.85
emotion intensity: 0.85 <- abs(-1 x 0.85)
adding FEAR: 0.85
adding ANGER: 0.85 towards player
adding belief: player shot, likelihood = 0.5

Decaying...
Updating...
Recalculating...
recalculating belief: player shot
affected goal: kill player, valence = 1
desirability: 1 <- 1 x 1
goal likelihood: 0.75 <- (1 x 0.5 + 1) / 2
delta goal likelihood: 0.75 <- 0.75
emotion intensity: 0.75 <- abs(1 x 0.75)
adding HOPE: 0.75

Current Internal State: Hope: 0.75; Fear: 0.77
Relationships:
        player -> Anger: 0.77
```

4.4 First-Person Shooter Game

First-Person shooters are games where the player experiences action with a first-person camera positioned at the level of the eyes and that involve shooting weapons and direct combat. We developed a simple FPS in which NPCs are controlled in a similar manner to the RTS example: they have a very simple decision making system akin to a finite state machine and they use A* for pathfinding. We have implemented the same emulation of fear as in the RTS when their life decreases. The goal for this case is to show in a different game genre than RTS how emotions generated by a simple GAMYGDALA instrumentation can impact NPC behavior.

We give the NPCs the goals to avoid ''die'' and to achieve ''win''. By defining the responsibility of the inflicted wound as belonging to the player the NPC has a social emotion: *Anger* towards the player. This adds to the current *Arousal*, which can be used to modify the behavior of the NPC.

The A* algorithm [41] is a graph search algorithm that finds the shortest path in a graph. Because in games, speed is more important than precision, A* uses a heuristic function that estimates the future steps approximatively so that there is no need to calculate all possible paths. Arousal taken from GAMYGDALA influences A* pathfinding precision by adding a random factor that is proportional to the *Arousal* level of the NPC to the estimation of the distance to the goal. This will produce erratic behavior by an aroused NPC, analogous to real-life: tense and nervous individuals make irrational and unpredictable choices. In addition to this, arousal causes the NPCs to have less shooting precision. As a consequence, when they get hurt arousal increases and they automatically start missing some shots.

5 Computation Speed

As GAMYGDALA is targeted at games involving a potentially large amount of NPCs present, it is important that it scales well. We performed two stress tests: timing populations of emotional NPCs of various sizes, and, running a 60 FPS (frames per second) simulation of 5,000 NPCs with which the user can interact by injecting beliefs using mouse clicks.

For the first experiment we run tests varying the following parameters: a population of size n, a number of goals g for each NPC in the population, and b beliefs for each NPC. GAMYGDALA's emotion calculation was applied to all NPCs after adding all beliefs. The generated goals and beliefs have random parameters and the beliefs affect a randomly chosen goal for that specific NPC . We ran five tests, three simple ones: with one goal and one belief per NPC, two goals and two beliefs per NPC and four goals and four beliefs per NPC, and two stress tests with five goals per NPC and 20 or 50 beliefs per NPC. Fig. 3 shows a comparison between the three cases.

Our tests ran on a 3 GHz processor in a single thread. The y-axis in Fig. 3 represents the total amount of time needed to calculate the corresponding emotions as well as do the belief updates. GAMYGDALA allows simulation of up to 35,000 NPCs, 2 goals and beliefs per NPC , with a total computing time of less than one second. This computing time includes 70,000 belief additions (2 per NPC), and a final emotion computation (appraisal). Of course a second of appraisal is not possible in a real game, but one can easily use standard load spreading techniques across frames, so that not all NPCs need updates every frame. When the number of goals and beliefs increases to a more realistic amount (goals and 20 beliefs per NPC), GAMYGDALA can still compute the appraisal for 5,000 NPCs (a total of 25,000 goals and 100,000 beliefs) in just 0.816 seconds.

As a second test we built a sandbox environment (using Microsoft'sXNA Framework). The world is a rectangular area with 5,000 NPCs present. The user influences them using the mouse: left clicking an area would send positive beliefs (beliefs that facilitate their goals) to the NPCs in that

IEEE Proof

area and right clicking an area would send negative beliefs (beliefs that block their goals). As an outcome, the NPCs become agitated (speed increases) if the *Arousal* is high, or calm if it is low and their color would change between red and green to show the level of *Pleasure*. In Fig. 4 a screenshot is shown (red is negative, green is positive). This simulation is representative of a user interacting with a local set of NPCs, such as in an RPG. In this case each emotional wave only hits a part of the population of 5,000 NPCs and the simulation easily keeps up a frame-rate of about 60FPS.

The experiments show that in these applications the computational costs are quite acceptable and scale well with the number of NPCs in the game. When a new events appears and it is relevant for an emotion in an NPC then approximately the same computations take place. The speed of processing a new event therefore grows linearly with the number of NPCs times the number of beliefs affected by that event (see also Fig. 3). The example above shows the effect on speed of increasing the number of agents and beliefs. This suggests that for many current games our approach is fast enough.

6 Conclusion

We have introduced GAMYGDALA, an emotion engine for games based on OCC [5]. Our approach proposes a solution that is positioned between event coding of affect, where each individual event has a predetermined annotated emotional consequence for each NPC, and a full blown cognitive appraisal model. Instead, for each (type of) NPC that needs simulated emotion the game developer specifies goals, and annotates game events with a relation to these goals. Based on this input, GAMYGDALA finds an emotion for that NPC according to the well-known OCC model. We have shown that GAMYGDALA is psychologically grounded, provides black-box Game-AI independent emotion support, and that it is efficient for large numbers of NPCs. We have done so by 1) specifying in detail how our computational mechanisms closely follow the OCC model, 2) presenting four examples that show that GAMYGDALA is indeed game AI (and game genre) independent, and 3) presenting results of a large scale simulation of thousands of NPCs showing acceptable performance. Because GAMYGDALA is a black-box emotional appraisal module, the game developer and game programmer do not need to be emotional appraisal experts to use it. Of course, a game developer needs to instrument the game to interface with GAMYGDALA. However, we have explained that this process is analogous to using a physics engine: provide the essentials and the engine will do the hard stuff. We conclude that the goal-based event annotation approach used in GAMYGDALA is suitable for the development of emotional NPCs in games. In this paper we have not shown that emotions generated with GAMYGDALA actually add to the level of fun of a game. This is a next step. Another direction for further work is to refine and test the mechanisms for modeling individual differences, for example using more elaborate dynamics of emotion. Questions such as, how should possibly conflicting emotions be combined, and, what are appropriate decay functions for a gaming setting, should be investigated. A relatively straightforward extension of GAMYGDALA is to include norms and object liking (the two other parts of OCC).

References

[1] Bethesda, *Skyrim*, Zenimax, 2011.

[2] W.S.N. Reilly, "Believable Social and Emotional Agents," PhD dissertation, School of Computer Science, Carnegie Mellon Univ., 1996.

[3] C. Bartneck, "Integrating the OCC Model of Emotions in Embodied Characters," *Proc. Workshop Virtual Conversational Characters: Applications, Methods, and Research Challenges*, 2002.

[4] E. Hudlicka, "Affective Game Engines: Motivation and Requirements," *Proc. Fourth Int'l Conf. Foundations of Digital Games*, pp. 299-306, 2009.

[5] A. Ortony, G.L. Clore, and A. Collins, *The Cognitive Structure of Emotions*, vol. 7, Cambridge Univ. Press, 1988.

[6] C.D. Elliott, "The Affective Reasoner: A Process Model of Emotions in a Multi-Agent System," PhD dissertation, Northwestern Univ., 1992.

[7] J. Gratch and S. Marsella, "A Domain-Independent Framework for Modeling Emotion," *J. Cognitive Systems Research*, vol. 5, pp. 269-306, 2004.

[8] E. Hudlicka, "Guidelines for Designing Computational Models of Emotions," *Int'l J. Synthetic Emotions*, vol. 2, no. 1, pp. 26-79, 2011.

[9] P. Ekman, "An Argument for Basic Emotions," *Cognition & Emotion*, vol. 6, no. 3/4, pp. 169-200, May 1992.

[10] A. Mehrabian, *Basic Dimensions for a General Psychological Theory*. OGH Publishers, 1980.

[11] A. Mehrabian, "Framework for a Comprehensive Description and Measurement of Emotional States," *Genetic, Social, and General Psychology Monographs*, vol. 121, no. 3, pp. 339-361, Aug. 1995.

[12] K.R. Scherer, "Appraisal Considered as a Process of Multilevel Sequential Checking," *Appraisal Processes in Emotion: Theory, Methods, Research*, pp. 92-120, Oxford Univ. Press, 2001.

[13] N.H. Frijda, *The Emotions*. Series Studies in Emotion and Social Interaction Cambridge Univ. Press, 1986.

[14] K.R. Scherer, "Toward a Dynamic Theory of Emotion: The Component Process Model of Affective States," *Geneva Studies in Emotion and Comm.*, vol. 1, pp. 1-98, 1987.

[15] A. Mehrabian, "Pleasure-Arousal-Dominance: A General Framework for Describing and Measuring Individual Differences in Temperament," *Current Psychology*, vol. 14, no. 4, pp. 261-292, Dec. 1996.

[16] N.H. Frijda and J. Swagerman, "Can Computers Feel? Theory and Design of an Emotional System," *Cognition & Emotion*, vol. 1, no. 3, pp. 235-257, 1987.

[17] Y. Guoliang, W. Zhiliang, W. Guojiang, and C. Fengjun, "Affective Computing Model Based on Emotional Psychology," *Proc. Second Int'l Conf. Advances in Natural Computation*, 2006.

[18] K.-L. Liu, "Affective Computing for Computer Games: Bots with Emotions," MSc thesis, Nat'l Taiwan Univ. of Science and Technology, 2007.

[19] J. Broekens and D. DeGroot, "Scalable and Flexibel Appraisal Models for Virtual Agents," *Proc. Fifth Game-On Int'l Conf.*, pp. 208-215, 2004.

[20] M. Ochs, D. Sadek, and C. Pelachaud, "A Formal Model of Emotions for an Empathic Rational Dialog Agent," *Autonomous Agents and Multi-Agent Systems*, vol. 24, no. 3, pp. 410-440, May 2012.

[21] S. Marsella, J. Gratch, and P. Petta, "Computational Models of Emotion," *A Blueprint for Affective Computing: A Sourcebook and Manual*, Oxford Univ. Press, 2010.

[22] W.S.N. Reilly, *Modeling What Happens between Emotional Antecedents and Emotional Consequents*, Austrian Soc. for Cybernetic Studies, 2006.

[23] E. Hudlicka, "This Time with Feeling: Integrated Model of Trait and State Effects on Cognition and Behavior," *Applied Artificial Intelligence*, vol. 16, no. 7/8, pp. 611-641, 2002.

[24] G. N. Yannakakis and J. Hallam, "Real-Time Game Adaptation for Optimizing Player Satisfaction," *IEEE Trans. Computational Intelligence and AI in Games*, vol. 1, no. 2, pp. 121-133, June 2009.

[25] P. Sweetser, D. Johnson, J. Sweetser, and J. Wiles, "Creating Engaging Artificial Characters for Games," *Proc. Second Int'l Conf. Entertainment Computing*, Carnegie Mellon University, pp. 1-8, 2003.

IEEE PROOF

[26] P. Baillie de-Byl, *Programming Believable Characters for Computer Games*. Charles River Media, Charles River Media, Inc., 2004.
[27] S. Grand, D. Cliff, and A. Malhotra, "Creatures: Artificial Life Autonomous Software Agents for Home Entertainment," *Proc. First Int'l Conf. Autonomous Agents*, W.L. Johnson, ed., 1997.
[28] S. Grand, *Creation: Life and How to Make It*. Harvard Univ. Press, 2000.
[29] J. Wexler, "Artificial Intelligence in Games: A Look at the Smarts behind Lionhead Studio's "Black and White" and Where It Can Go and Will Go in the Future," Univ. of Rochester, 2002.
[30] E. Hudlicka and J. Broekens, "Foundations for Modelling Emotions in Game Characters: Modelling Emotion Effects on Cognition," *Proc. IEEE Third Int'l Conf. Affective Computing and Intelligent Interaction (ACII)*, 2009.
[31] C. A. Smith and P. C. Ellsworth, "Attitudes and Social Cognition," *J. Personality and Social Psychology*, vol. 48, no. 4, pp. 813-838, 1985.
[32] P. Carlisle, "A Framework for Emotional Digital Actors," *Game Programming Gems 8*, pp. 312-322, Cengage Learning, 2010.
[33] R.R. McCrae and P.T. Costa, "Toward a New Generation of Personality Theories: Theoretical Contexts for the Five-Factor Model," *The Five-Factor Model of Personality: Theoretical Perspectives*, pp. 51-87, Guilford Press, 1996.
[34] J. Grey and J.J. Bryson, "Procedural Quests: A Focus for Agent Interaction in Role-Playing-Games," *Proc. AISB Symp.: AI & Games*, pp. 3-10, 2009.
[35] J. McCoy, M. Treanor, B. Samuel, M. Mateas, and N. Wardrip-Fruina, "Prom Week: Social Physics as Gameplay," *Proc. Sixth Int'l Conf. Foundations of Digital Games*, pp. 319-321, 2011.
[36] T. Watani, *Pac-Man*, Namco, 1980.
[37] B. Schwab, *AI Game Engine Programming*. Charles River Media, 2004.
[38] D. Mark, *Behavioral Mathematics for Game AI*. Charles River Media, 2009.
[39] A. Champandard, "Behavior Trees for Next-Gen Game AI," *Proc. Game Developers Conf., Audio Lecture*, 2007.
[40] D. Fu and R. Houlette, *The Ultimate Guide to FSMs in Games*, vol. 2, pp. 283-302, Charles River Media, 2004.
[41] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100-107, Feb. 1968.

Alexandru Popescu received the MSc degree in artificial intelligence from the University of Amsterdam, and is currently working as a software developer.

Joost Broekens received the MSc degree in computer science from Delft University and the PhD degree in computer science from the University of Leiden on computational modeling of emotion in relation to reinforcement learning. He is an assistant professor at Delft University and teaches affective computing. His research interests include reinforcement learning, affective computing, computational models of affective processes, human-technology interaction, and gaming research.

Maarten van Someren teaches artificial intelligence at the University of Amsterdam. His research interests include machine learning, in particular applications, semi-supervised and transfer learning, and foundations of induction. When the opportunity arises, he combines this with models of human cognition.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

IEEE Proof

Queries to the Author

Q1. Please cite the Table 2 at an appropriate place in the text.
Q2. Please cite Figs. 1 and 2 at an appropriate place in the text.

IEEE Proof