Module Interface Specification for EMgine: A Computational Model of Emotion for Enhancing Non-Player Character Believability in Games

Geneva M. Smith and Dr. Jacques Carette

Version 0.1 (March 24, 2023)

Geneva M. Smith and Dr. Jacques Carette
Dept. of Computing and Software—McMaster University

Revision History

Date	Version	Notes
March 24, 2023	0.1	• Initial document with the Emotion Intensity Component (Emotion Intensity Type [M1] and Emotion State Type [M3] Modules) based on SRS Version 1.5 and MG Version 1.5

Contents

Geneva M. Smith and Dr. Jacques Carette
Dept. of Computing and Software—McMaster University

1 Symbols, Abbreviations, and Acronyms

For EMgine's other symbols, abbreviations, and acronyms, see the:

- Software Requirement Specification (SRS) at https://github.com/GenevaS/EMgine/blob/main/docs/SRS/EMgine_SRS.pdf, and
- Module Guide (MG) at https://github.com/GenevaS/EMgine/blob/main/docs/Design/MG/EMgine_MG.pdf.

Text	Description
ADT	Abstract Data Type
Seq.	Sequence

2 Introduction

This document details the Module Interface Specifications for EMgine, a Computational Model of Emotion (CME) for Non-Player Characters (NPCs) to enhance their believability, with the goal of improving long-term player engagement. EMgine is for *emotion generation*, accepting user-defined information from a game environment to determines what emotion and intensity a NPC is "experiencing". How the emotion is expressed and what other effects it could have on game entities is left for game designers/developers to decide.

Other necessary documents include the Software Requirement Specification and Module Guide. You can find the full documentation and implementation at

https://github.com/GenevaS/EMgine

3 Notation

The module structure in this MIS comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := represents a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 \,|\, c_2 \Rightarrow r_2 \,|\, ... \,|\, c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by EMgine.

Data Type	Notation	Description
Boolean	$\mathbb{B}$	An element in the set of {*True*, *False*}
Character	char	A single symbol or digit
Integer	$\mathbb{Z}$	A number without a fractional component in (-∞, ∞)
Natural	$\mathbb{N}$	A number without a fractional component in [0, ∞)
Real	$\mathbb{R}$	Any number in (-∞, ∞)

The specification of EMgine uses some derived data types:

- *Sets*: Unordered lists filled with elements of the same data type
- *Sequences*: Ordered lists filled with elements of the same data type
- *Strings*: Sequences of characters
- *Tuples*: Contain a list of labelled values, potentially of different data types

EMgine also uses functions, defined by the data types of their inputs and outputs. The MIS describes local functions by giving their type signature, followed by their specification.

4 Module Hierarchy

This table is taken directly from EMgine's Module Guide (Version 1.5).

Level 1	Level 2	Level 3
Behaviour-Hiding Module	Emotion Intensity Module	[M1] Emotion Intensity Type Module [M2] Emotion Intensity Function Module
	Emotion State Module	[M3] Emotion State Type Module [M4] Emotion Generation Module
	Emotion Decay Module	[M5] Emotion Intensity Decay Module [M6] Emotion State Decay Module
	PAD Module	[M7] PAD Type Module [M8] PAD Function Module
	Emotion Module	[M9] Emotion Type Module [M10] Emotion Function Module
	Entity Module	[M11] Goal Type Module [M12] Plan Type Module [M13] Social Attachment Type Module [M14] Attention Type Module
Software Decision Module	World Module	[M15] Time Type Module [M16] World Type Module

5 Emotion Intensity Type Module (M1): EmIntensityT and EmIntensityChgT

The Emotion Intensity ADT defines the emotion intensity $\mathbb{I}$ and emotion intensity change $\mathbb{I}_\Delta$ data types. They are defined in the same module because $\mathbb{I}_\Delta$ directly depends on the existence of $\mathbb{I}$.

Emotion intensity is bounded by $[0, +\infty)$, but emotion intensity change is unbounded. This design is application-agnostic and can exist independently of EMgine.

5.1 Known Information Leaks

- The internal representation of EmIntensityT and EmIntensityChgT ($\mathbb{R}$) is manipulated with arithmetic operations, which are not meaningful outside EMgine's modules

5.2 Uses

- None

5.3 Syntax: EmIntensityT

5.3.1 Exported Constants

- MIN_INTENSITY = 0

5.3.2 Exported Types

- EmIntensityT = { $intensity : \mathbb{R} \mid \text{MIN\_INTENSITY} \leq intensity : intensity$ }

5.3.3 Exported Access Programs

Name	In	Out	Exceptions
new EmIntensityT	$\mathbb{R}$	EmIntensityT	W-EIT_INTENSITY_TOO_SMALL
CompareToIntensity	EmIntensityT	$\mathbb{Z}$	–
EqualsMinIntensity	–	$\mathbb{B}$	–
Normalize	EmIntensityT	EmIntensityT	E-EIT_INTENSITY_CANNOT_BE_ZERO
ToReal	–	$\mathbb{R}$	–
UpdateWithChg	EmIntensityChgT	EmIntensityT	–

5.3.4 Local Access Programs

Name	In	Out	Exceptions
GetIntensity	–	$\mathbb{R}$	–

5.4 Semantics: EmIntensityT

5.4.1 State Variables

- *intensity* : $\mathbb{R}$

5.4.2 State Invariant

- MIN_INTENSITY $\leq$ *intensity*

5.4.3 Access Routine Semantics

new EmIntensityT(i):

- output: *out* := self
- transition: *intensity* := `clamp` (i, MIN_INTENSITY, $+\infty$)
- exception: *exc* := ($i <$ MIN_INTENSITY $\Rightarrow$ W-EIT_INTENSITY_TOO_SMALL)

CompareToIntensity(i):

- output: *out* := ($|C| \leq \epsilon \Rightarrow 0 \mid C > \epsilon \Rightarrow 1 \mid C < -\epsilon \Rightarrow -1$)
  where C = *intensity* − i.GetIntensity()
- exception: None

EqualsMinIntensity():

- output: *out* := $|$ *intensity* − MIN_INTENSITY $| \leq \epsilon$
- exception: None

Normalize(*scale*):

- output: *out* := new EmIntensityT$\left(\frac{\textit{intensity}}{\textit{scale}.\text{GetIntensity}()}\right)$
- exception: *exc* := (*scale*.GetIntensity() = 0 $\Rightarrow$ E-EIT_INTENSITY_CANNOT_BE_ZERO)

ToReal():

- output: *out* := *intensity*
- exception: None

UpdateWithChg(*chg*):

- output: *out* := new EmIntensityT(i')

$$\text{where } i' = \begin{cases} 0.1 \cdot \log_2\left(2^{10 \cdot intensity} + 2^{10 \cdot C}\right), & C > \epsilon \\ 0.1 \cdot \log_2\left(2^{10 \cdot intensity} - 2^{10 \cdot |C|}\right), & C \leq \epsilon \ \wedge \ intensity > |C| \\ 0.0, & C \leq \epsilon \ \wedge \ intensity \leq |C| \\ intensity, & Otherwise \end{cases}$$

where C = *chg*.GetDelta() # from EmIntensityChgT

- exception: None

5.4.4 Local Functions

GetIntensity():

- output: *out* := *intensity*
- exception: None

5.5 Syntax: EmIntensityChgT

5.5.1 Exported Types

- EmIntensityChgT : $\mathbb{R}$

5.5.2 Exported Access Programs

Name	In	Out	Exceptions
new EmIntensityChgT	$\mathbb{R}$	EmIntensityChgT	–
CompareToIntensityChg	EmIntensityChgT	$\mathbb{Z}$	–
EqualsMinIntensity	–	$\mathbb{B}$	–
ScaleByValue	$\mathbb{R}$	EmIntensityChgT	–
ToReal	–	$\mathbb{R}$	–

5.5.3 Local Access Programs

Name	In	Out	Exceptions
GetDelta	–	$\mathbb{R}$	–

5.6 Semantics: EmIntensityChgT

5.6.1 State Variables

- *iDelta* : $\mathbb{R}$

5.6.2 Access Routine Semantics

new EmIntensityChgT(*delta*):

- output: *out* := self
- transition: *iDelta* := *delta*
- exception: None

CompareToIntensityChg(*d*):

- output: $out := (|C| \leq \epsilon \Rightarrow 0 \mid C > \epsilon \Rightarrow 1 \mid C < -\epsilon \Rightarrow -1)$
  where $C = iDelta - d$.GetDelta()
- exception: None

EqualsMinIntensity():

- output: $out := |iDelta - \text{MIN\_INTENSITY}| \leq \epsilon$
- exception: None

ScaleByValue(v):

- output: *out* := new EmIntensityT($iDelta \cdot v$)
- exception: None

ToReal():

- output: *out* := *iDelta*
- exception: None

5.6.3 Local Functions

GetDelta():

- output: *out* := *iDelta*
- exception: None

6 Emotion State Type Module (M3): EmotionKindsT and EmStateT

The Emotion State ADT defines the emotion kinds/categories/types $\mathbb{K}$ and emotion state $\mathbb{ES}$ data types. They are defined in the same module because $\mathbb{ES}$ directly depends on the size and order of $\mathbb{K}$.

The $\mathbb{ES}$ data type contains two records defining the current intensity and maximum intensity for each emotion kind $k \in \mathbb{K}$. Emotion state current intensities are bounded by $[min, max.k]$, where min is defined by EmIntensityT and $max.k$ is the maximum intensity for emotion type k. For convenience, a default maximum intensity of 100 is available if users do not want to define their own.

This design is application-agnostic and can exist independently of EMgine iff EmIntensityT or equivalent are available.

6.1 Uses

- EmIntensityT (Section 5)

6.2 Syntax

6.2.1 Exported Constants

- MAX_INTENSITY_DEFAULT = new EmIntensityT(100.0)
- NUM_EMOTION_KINDS = | EmotionKindsT |

6.2.2 Exported Types

- EmotionKindsT = {`Fear`, `Anger`, `Sadness`, `Joy`, `Interest`, `Surprise`, `Disgust`, `Acceptance`}
- EmStateT = $\Big\{ \begin{aligned} &\textit{intensities} = \text{Tuple of } (\forall k \in \text{EmotionKindsT} \to k : \text{EmIntensityT}), \\ &\textit{max} = \text{Tuple of } (\forall k \in \text{EmotionKindsT} \to k : \text{EmIntensityT}) \end{aligned} \Big\}$

6.2.3 Exported Access Programs

Name	In	Out	Exceptions
new EmStateT	Seq. of EmIntensityT, Seq. of EmIntensityT	EmStateT	E-EST_TOO_FEW_VALUES_FOR_STATE, E-EST_TOO_MANY_VALUES_FOR_STATE
GetNumEmotionKinds	–	$\mathbb{N}$	–
GetIntensityOf	EmotionKindsT	EmIntensityT	–
UpdateAllIntensitiesWithChgs	Seq. of EmIntensityChgT	EmStateT	E-EST_TOO_FEW_VALUES_FOR_STATE, E-EST_TOO_MANY_VALUES_FOR_STATE
GetMaxIntensityOf	EmotionKindsT	EmIntensityT	–
ChangeMaxIntensityOf	EmotionKindsT, EmIntensityT	–	–
ResetMaxIntensityOf	EmotionKindsT	–	–
UpdateAllMaxIntensities	Seq. of EmIntensityT	–	E-EST_TOO_FEW_VALUES_FOR_STATE, E-EST_TOO_MANY_VALUES_FOR_STATE

6.2.4 Local Access Programs

Name	In	Out	Exceptions
ValidateIntensityOf	EmotionKindsT	–	W-EST_INTENSITY_TOO_LARGE

6.3 Semantics

6.3.1 State Variables

- *intensities* : Tuple of ($\forall k \in$ EmotionKindsT $\rightarrow k :$ EmIntensityT)
- *max* : Tuple of ($\forall k \in$ EmotionKindsT $\rightarrow k :$ EmIntensityT)

6.3.2 State Invariant

- $\forall k \in$ EmotionKindsT $\rightarrow$ *intensities*.k.CompareToIntensity(*max*.k) ≤ 0
  $\wedge\ \exists k \in$ EmotionKindsT $\rightarrow \neg$*intensities*.k.EqualsMinIntensity()
  # CompareToIntensity, EqualsMinIntensity from EmIntensityT

6.3.3 Access Routine Semantics

new EmStateT(i, m):

- output: *out* := self
- transition: *intensities*, *max* := i, UpdateAllMaxIntensities(m)
- exception: *exc* := ($|i| <$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_FEW_VALUES_FOR_STATE
  $|i| >$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_MANY_VALUES_FOR_STATE
  $|m| <$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_FEW_VALUES_FOR_STATE
  $||m| >$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_MANY_VALUES_FOR_STATE)
  # Additional exceptions thrown from UpdateAllMaxIntensities

GetNumEmotionKinds():

- output: out := NUM_EMOTION_KINDS
- exception: None

GetIntensityOf(*emotion*):

- output: *out* := *intensities.emotion*
- exception: None

UpdateAllIntensitiesWithChgs(*chgs*):

- output: *out* := new EmStateT(i, *max*)
  where i = *intensities* with $\forall k \in$ EmotionKindsT $\rightarrow$ ValidateIntensityOf($I(k)$)
  where $I(k)$ = *intensities.k*.UpdateWithChg(*chgs.k*) # from EmIntensityT
- exception: *exc* := ($|chgs| <$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_FEW_VALUES_FOR_STATE
  $|chgs| >$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_MANY_VALUES_FOR_STATE)
  # Additional exceptions thrown from ValidateIntensityOf

GetMaxIntensityOf(*emotion*):

- output: *out* := *max.emotion*
- exception: None

ChangeMaxIntensityOf(*emotion*, *max*):

- transition: *max.emotion*, *intensities.emotion* := *max*, ValidateIntensityOf(*emotion*)
- None # Additional exceptions thrown from ValidateIntensityOf

ResetMaxIntensityOf(*emotion*):

- transition: ChangeMaxIntensityOf(*emotion*, M)
  where M = new EmIntensityT(MAX_INTENSITY_DEFAULT)
- exception: None # Additional exceptions thrown from ChangeMaxIntensityOf

UpdateAllMaxIntensities(m):

- transition: *max* := $\forall k \in$ EmotionKindsT $\rightarrow$ ChangeMaxIntensityOf($k, m.k$)
- exception: *exc* := ($|m| <$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_FEW_VALUES_FOR_STATE
  $||m| >$ NUM_EMOTION_KINDS $\Rightarrow$ E-EST_TOO_MANY_VALUES_FOR_STATE)
  # Additional exceptions thrown from ChangeMaxIntensityOf

6.4 Local Functions

ValidateIntensityOf(*emotion*):

- transition: *intensities.emotion* :=
  (*intensities.emotion*.CompareToIntensity(*max.emotion*) $> 0 \Rightarrow$ *max.emotion*
  $| \epsilon \Rightarrow i$)
- exception: *exc* := (i.CompareToIntensity(*max.emotion*) $> 0 \Rightarrow$ W-EST_INTENSITY_TOO_LARGE)

7 References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. 2003. *Fundamentals of Software Engineering* (2nd ed.). Prentice Hall, Upper Saddle River, NJ, USA. ISBN 0-13-305699-6.

Daniel M. Hoffman and Paul A. Strooper. 1995. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA. http://citeseer.ist.psu.edu/428727.html

A Error Messages

ID	Description
E-EIT_INTENSITY_CANNOT_BE_ZERO	Error: Reference intensity has a value of zero. Cannot complete normalization.
E-EST_TOO_FEW_VALUES_FOR_STATE	Error: Data provided for *n* emotion kinds, which is less than the NUM_EMOTION_KINDS types required by the emotion state. No changes made.
E-EST_TOO_MANY_VALUES_FOR_STATE	Error: Data provided for *n* emotion kinds, which is more than the NUM_EMOTION_KINDS types required by the emotion state. No changes made.

B Warning Messages

ID	Description
W-EIT_INTENSITY_TOO_SMALL	Warning: Value provided is smaller than the minimum emotion intensity. Clamping to the range [MIN_INTENSITY, ∞).
W-EST_INTENSITY_TOO_LARGE	Warning: Emotion intensity for *emotion* is larger than its maximum intensity. Clamping to the range [*intensities.emotion*.GetMinIntensity(), *max.emotion*].