

D2.4 - Final Bundle of Client-side Components

Citation for published version (APA):

Calvo Morata, A., Fernandez Manjon, B., Rotaru, D. C., Freire Moran, M., Martinez Ortiz, I., Riestra, R., Dascalu, M., Lala, R., Nyamsuren, E., Van der Vegt, W., Bahreini, K., Bontchev, B., Westera, W., & Maurer, M. (2017). *D2.4 - Final Bundle of Client-side Components*.

Document status and date:

Published: 01/08/2017

Document Version:

Peer reviewed version

Document license:

CC BY-NC-SA

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

<https://www.ou.nl/taverne-agreement>

Take down policy

If you believe that this document breaches copyright please contact us at:

pure-support@ou.nl

providing details and we will investigate your claim.

Downloaded from <https://research.ou.nl/> on date: 09 Sep. 2021

Open Universiteit
www.ou.nl





Realising an Applied Gaming Ecosystem

Research and Innovation Action

Grant agreement no.: 644187

D2.4 - Final Bundle of Client-side Components (WP2)

RAGE – WP2 – D2.4

Project Number	H2020-ICT-2014-1
Due Date	31 August 2017
Actual Date	31 July 2017
Document Author/s	Antonio Calvo Morata, Baltasar Fernández Manjón, Dan Cristian Rotaru, Manuel Freire Morán, Iván Martínez Ortiz, Ruben Riestra, Mihai Dascalu, Raja Lala, Mathias Maurer, Enkhbold Nyamsuren, Wim van der Vegt, Kiavash Bahreini, Boyan Bontchev, Wim Westera
Version	1.3
Dissemination level	PU
Status	Final
Document approved by	WW



Document Version Control			
Version	Date	Change Made (and if appropriate reason for change)	Initials of Commentator(s) or Author(s)
0.1	13 July 2017	First version	ACM, BFM, DCR, MFM, IMO
1.0	21 July 2017	Second complete version	ACM, BFM, DCR, MFM, IMO, RR, MD, RL, MM, EN, WvV
1.1	21 July 2017	Complete version updated with partners comments	ACM, BFM, DCR, MFM, IMO, WvV, KB
1.2	31 July 2017	Complete version updated with partners comments	ACM, BFM, DCR, MFM, IMO, BB, WW
1.3	23 August 2017	Updated after QA	ACM, BFM, DCR, MFM, IMO, BB, WW

Document Change Commentator or Author		
Author Initials	Name of Author	Institution
ACM	Antonio Calvo Morata	UCM
BB	Boyan Bontchev	SU
BFM	Baltasar Fernández Manjón	UCM
DCR	Dan Cristian Rotaru	UCM
EN	Enkhbold Nyamsuren	OUNL
IMO	Iván Martínez Ortiz	UCM
KB	Kiavash Bahreini	OUNL
MD	Mihai Dascalu	UPB
MFM	Manuel Freire Morán	UCM
MM	Mathias Maurer	TUG
RL	Raja Lala	UU
RR	Ruben Riestra	Inmark
WvV	Wim van der Vegt	OUNL
WW	Wim Westera	OUNL

Document Quality Control

Version QA	Date	Comments (and if appropriate reason for change)	Initials of QA Person
	9/8/2017	Revisions and Recommendations	PH
	17/8/2017	Revisions and Recommendations	MD

TABLE OF CONTENTS

	Page
EXECUTIVE SUMMARY	6
1 GAS: GAMING ANALYTICS SUITE.....	7
1.1 Included assets.....	8
1.1.1 Client Tracker	8
Links.....	8
2 P-CAP: PLAYER COMPETENCE ADAPTATION PACK	9
2.1 Included assets.....	10
2.1.1 Domain Model Asset	10
General Description	10
Pedagogical Value.....	10
Development Benefits.....	10
Links.....	10
2.1.2 Competence Assessment Asset	11
General Description	11
Pedagogical Value.....	11
Why a game developer benefits from this asset	11
Links.....	11
3 P-MAP: PLAYER MOTIVATION ADAPTATION PACK.....	12
3.1 Included WP2 assets.....	12
3.1.1 Motivation Assessment Asset	12
General Description	13
Pedagogical Value.....	13
Why a game developer benefits from this asset	13
Links.....	13
4 REAL-TIME EMOTION DETECTION FROM FACIAL EXPRESSIONS	14
4.1 Included assets.....	15
4.1.1 Real-time Emotion Detection from Facial Expressions	15
Links.....	15
5 EASY DIALOGUE INTEGRATOR	16
5.1 Included assets.....	17
5.1.1 Communication scenario editor.....	17
Links.....	19
6 SHARED DATA STORAGE COMPONENT	20
6.1 Included assets.....	20
6.1.1 Game Storage asset.....	20
Links.....	21
7 REAL-TIME AROUSAL DETECTION USING GALVANIC SKIN RESPONSE	22
7.1 Included assets.....	23
7.1.1 Real time arousal detection using GSR asset.....	23
Links.....	24
8 MULTIMODAL EMOTION DETECTION.....	25
8.1 Included assets.....	26
8.1.1 Multimodal Emotion Detection Asset.....	26
Links.....	27
9 CONCLUSIONS	28
REFERENCES.....	29

LIST OF FIGURES

Figure 1. Tracker implementations for RAGE Analytics.....	8
Figure 2. Competence structure.....	10
Figure 3. Competence assessment structure.	11
Figure 4. Motivation assessment measures.....	13
Figure 5. Communication scenario editor.	17
Figure 6. Conversations structure in the communication scenario editor.	18
Figure 7. Interface with scenario features.	18
Figure 8. Scenario editor to simplify dialogue creation.	19
Figure 10. Usage of a GSR measuring device.....	23
Figure 11. A screenshot of the visualization application with zooming of raw and filtered signals.	24
Figure 12. Snapshot of the Multimodal Emotion Recognition demo, illustrating the integration of facial, text, and speech recognition	27

LIST OF TABLES

Table 1. Gaming Analytics Suite Factsheet	7
Table 2. Player Competence Adaptation Pack Factsheet	9
Table 3. Player Motivation Adaptation Pack factsheet.....	12
Table 4. Real-time Emotion Detection from Facial Expressions factsheet	14
Table 5. Easy Dialogue Integrator factsheet	16
Table 7. Shared Data Storage Component factsheet	20
Table 8. Real time arousal detection using GSR asset factsheet.....	22
Table 9. Multimodal emotion detection asset factsheet	25

LIST OF ABBREVIATIONS

EDA	Electrodermal Activity
GSR	Galvanic Skin Response
INESC-ID	Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento
JSON	JavaScript Object Notation
OUNL	Open University of the Netherlands
SU	Sofia University
TUGRAZ	Technical University of Graz
UCM	Universidad Complutense de Madrid
UPB	University Politehnica of Bucharest
UU	Utrecht University
xAPI	Experience API (ADL)
XML	eXtensible Markup Language

EXECUTIVE SUMMARY

This document describes the final bundle of client-side components, including descriptions of their functionality, and links to their full designs and downloadable versions. This bundle aggregates only the WP2 assets. Other client-side assets not covered here will be addressed in the final WP3 deliverables.

This document links the asset catalog with the technical descriptions of the client-side assets produced in the project.

For every asset, the following items are available:

1. A Catalogue product fact sheet for the asset or asset suite/bundle. To simplify the asset descriptions and encourage adoption by the game industry, fact sheets may aggregate different individual assets descriptions into a single sheet.
2. The final and updated asset information and links to the source code and documentation.
3. The source code with complete and updated documentation in an open repository (such as a GitHub wiki or similar).

All the updated asset information is permanently available on-line via the public repositories where the assets are stored. It is worth noting that those assets created and licenced as open software will be continuously improved and maintained by their creators until the end of the project (the task has been extended to month 48) and beyond.


This document is focused on client-side components, for a full description of the related server-side components, please refer to D2.2 - Final Bundle of Server-side Components (WP2).

1 GAS: GAMING ANALYTICS SUITE

The Gaming Analytics Suite (GAS) comprises of detailed descriptions of the 5 UCM Assets:

1. Server-Side Interaction Storage and Analytics
2. Authentication & Authorization
3. Game Storage, Server-Side
4. Server-side Dashboard and Analysis
5. Client Tracker

Table 1. Gaming Analytics Suite Factsheet

	<h3 style="text-align: center;">GAS</h3> <h2 style="text-align: center;">Gaming Analytics Suite</h2> <p>UCM Assets: Server-Side Interaction Storage and Analytics Authentication & Authorization Game Storage, Server-Side Server-side Dashboard and Analysis Client Tracker</p>
Short description	Provides data and insights on game usage performance
Type of product	Integrated pack of game components
Main benefit for Game Studio	<ul style="list-style-type: none"> • Gain creator: Evidence based enhanced games: enables games ROI calculation, thus boosting game potential sales • Pain reliever: improved feedback and intelligence, cutting the time to market and ensuring that games are fit for purpose
Main features for Game designers/developers	<ul style="list-style-type: none"> • GAS covers the entire process required to create, review and communicate a game's empirical performance evidence. GAS enables the configuration and execution of data gathering from in-game interactions, their analysis, querying, and dashboard visualization • Enriched data-evidence produced by GAS is useful both as decision making support among users (buy-in critical success factor), as well as a studio's internal monitoring of game's core logic, mechanics and performance, allowing better and quicker improvements.
Main features for Game customer/user	Provides for monitoring and evaluation of an end users' performance and of their progress towards pursued objectives.
Creator	Universidad Complutense de Madrid, UCM
Further info	https://github.com/e-ucm/rage-analytics/wiki

In this document we provide descriptions of the **client-side assets (client tracker)**. All other assets are described in D2.2 – Final Bundle of Server-side Components (WP2).

1.1 Included assets

1.1.1 Client Tracker

This component sends analytics information to a server; or, if a server is not available, the component stores them locally until the server is available (allowing semi-connected use). The tracker implementations were updated as previously described in D2.5 to conform to the xAPI model for serious games (accessible at: <http://xapi.e-ucm.es/vocab/seriousgames>), and are fully described in (Serrano, 2017). This model allows the tracking of learning analytics data for serious games, also referred to as game learning analytics (Freire, 2016).

Three tracker implementations are available and depicted in Figure 1:

- C# version, fully compliant with the RAGE asset architecture explained in D1.4.
- Java / Libgdx version.
- Unity3D version (also written in C#, but using Unity3D libraries).

Following this architecture, as part of the H2020 BEACONING Project, an additional JavaScript version of the tracker has also been implemented.

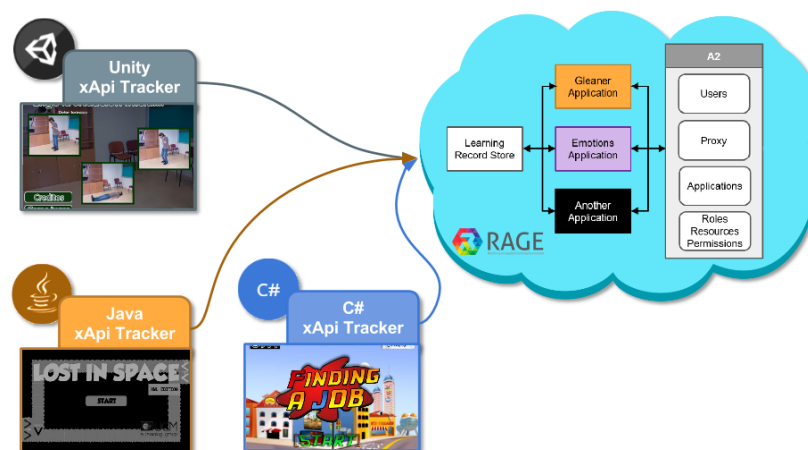


Figure 1. Tracker implementations for RAGE Analytics.

Links

- Sources:
 - <https://github.com/e-ucm/unity-tracker> (Unity version)
 - <https://github.com/e-ucm/ClientSideTrackerAsset> (C# version)
 - <https://github.com/e-ucm/libgdx-tracker> (Java/Libgdx version)
- Design:
 - https://docs.google.com/document/d/1_XtAMvI9mNhR5Qxc9RHF6OWhRBJyL2lq6biAw_tZegAQ/edit?usp=sharing
- User documentation:
 - <https://github.com/e-ucm/rage-analytics/wiki/Tracker>
- Complete RAGE Analytics documentation:
 - <https://github.com/e-ucm/rage-analytics/wiki>
- Example integration with game:
 - <https://github.com/e-ucm/QuizDemo>

2 P-CAP: PLAYER COMPETENCE ADAPTATION PACK

The Player Competence Adaptation Pack (P-CAP) provides detailed information relating to the 3 TUG Assets:

1. Domain Model Asset
2. Competence Assessment Asset
3. Competence-based Adaptation Asset

Table 2. Player Competence Adaptation Pack Factsheet

image	<h3>P-CAP Player Competence Adaptation Pack</h3> <p>TUGRAZ Assets: Domain Model Asset Competence Assessment Asset Competence-based Adaptation Asset</p>
Short description	P-CAP evaluates/profiles player's abilities/ skills/ competencies and supports Adaptation of the game to match these profiles
Type of product	Bundled game components
Main benefit for Game Studio	<p>Gain creator:</p> <ul style="list-style-type: none"> • Demonstrates stakeholders the usefulness of the game based on user interaction data • Demonstrates the learning progress to teachers/students • Identifies students' weaknesses <p>Pain reliever: Optimises the efforts spent in ensuring a well-fitting gaming experience, as decisions can be based on actual data</p>
Main features for Game designers/developers	Manages the level of difficulty the game in line with the player's needs and abilities (max. playing experience with a given effort)
Main features for Game customer/user	Improved tracking of the players learning progress without adapting the game, or (optionally) even adjust the game to optimize the learning tailored to the player. Builds games storylines along the natural structure of the learning domain, i.e. along the natural way of learning the things that are taught.
Creator	Technical University of Graz, TUGRAZ
Further info	TUG asset overview page: http://css-kti.tugraz.at/projects/rage/assets/

2.1 Included assets

2.1.1 Domain Model Asset

General Description

The purpose of the Domain Model Asset is to model the knowledge domain tackled in the game and represent the subject matter to be learned. In the game development process the competences conveyed by a game should be identified. These competences are then modelled in a prerequisite structure capturing dependencies (i.e. prerequisites) between competences (see below for an example). This structure can be used for both, the definition of learning goals (in terms of competences) and the identification of meaningful sequences how the learning goals should be achieved. The competence structure (Figure 2) can be used by other assets or games for competence assessment, recommendations which competences should be learned, goal definitions, and learning progress identification.

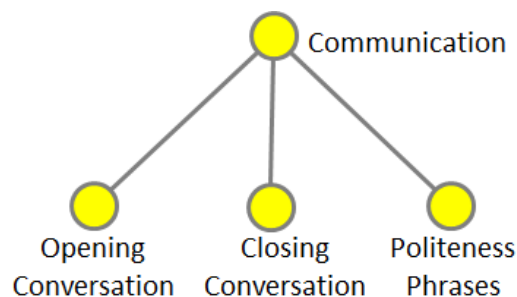


Figure 2. Competence structure.

Pedagogical Value

The main value of this asset is to define a domain knowledge in terms of the competences that should be learned with a game.

Development Benefits

This asset allows an easy handling of competences within a game

- Bringing in a psychological model of competence development into game design
- Competences conveyed by a game can easily be defined, structured, and related to the game and its game situations in advance
- Other assets can easily make use of existing competence structures

Links

- Sources:
<https://github.com/RAGE-TUGraz/CompetenceBasedAssets>
- Design:
<http://css-kti.tugraz.at/projects/rage/assets/designdocuments/DesignDocument-DomainModelAsset.pdf>
- User documentation:
<http://css-kti.tugraz.at/projects/rage/assets/factsheets/FactSheet-DomainModelAsset.pdf>
<http://css-kti.tugraz.at/projects/rage/assets/designdocuments/DesignDocument-DomainModelAsset.pdf>
- Example integration with game:
<http://css-kti.tugraz.at/projects/rage/assets/dummygame/DummyGame-Application.zip>
- TUG asset overview page:

<http://css-kti.tugraz.at/projects/rage/assets/>

2.1.2 Competence Assessment Asset

General Description

The Competence Assessment Asset's function is to uncover the competence state of a player while playing a game. In the game development process the competences addresses by a game are defined and structured. Then, the events or tasks in a game that provide evidence whether an individual competence is available or not should be identified. During game play this information is used by the asset to identify and update the competences that a player has available (Figure 3). The assessment result can be used by other assets to adapt the game play or to select an appropriate game. In addition, the result can be visualized in the dashboard of the UCM infrastructure. The figure below shows an illustration how the competences that are demonstrated by the player during the game play are assessed by the asset.

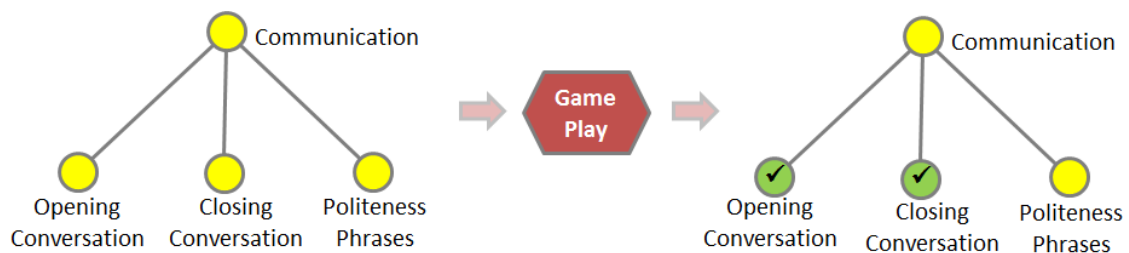


Figure 3. Competence assessment structure.

Pedagogical Value

The value of this asset is in identifying the competences a player possesses. In many cases, acquiring competences is the main goal of performing a learning activity. The assessment, is able to identify competence gaps validate if the achievement of learning goals

Why a game developer benefits from this asset

This asset facilitates competence assessment within a game

- Bringing in a psychological model of competence development into game design
- The asset enables non-invasive competence assessment
- The same domain model can be reused and shared across games

Links

- Sources: <https://github.com/RAGE-TUGraz/CompetenceBasedAssets>
- Design: <http://css-kti.tugraz.at/projects/rage/assets/designdocuments/DesignDocument-CompetenceAssessmentAsset.pdf>
- User documentation: <http://css-kti.tugraz.at/projects/rage/assets/factsheets/FactSheet-CompetenceAssessmentAsset.pdf>
<http://css-kti.tugraz.at/projects/rage/assets/designdocuments/DesignDocument-CompetenceAssessmentAsset.pdf>
- Example integration with game: <http://css-kti.tugraz.at/projects/rage/assets/dummygame/DummyGame-Application.zip>
- TUG asset overview page: <http://css-kti.tugraz.at/projects/rage/assets/>

3 P-MAP: PLAYER MOTIVATION ADAPTATION PACK

The Player Motivation Adaptation Pack (P-MAP) comprises of information relating to the 2 TUG Assets:

1. Motivation Assessment Asset
2. Motivation-based Adaptation Asset

Table 3. Player Motivation Adaptation Pack factsheet

image	<h2>P-MAP Player Motivation Adaptation Pack</h2> <p>TUGRAZ Assets: Motivation Assessment Asset Motivation-based Adaptation Asset</p>
Short description	P-MAP evaluates a players' motivational state and adapts the game to the player's motivational state
Type of product	Bundled game components
Main benefit for Game Studio	<p>Gain creator:</p> <ul style="list-style-type: none"> • Shows stakeholders the motivation unleashed by your game • Identifies unmotivating episodes in the game <p>Pain reliever: Optimises the efforts spent in ensuring maintenance of players' motivation</p>
Main features for Game designers/developers	As a serious game developer , I can use this Assets to adopt my game to the player's motivational state. An intervention at the right point of time can be highly effective.
Main features for Game customer/user	<p>As an educationalist, I can be certain that the player is not bored by educational input and still enjoys playing and learning with the game. The chance for a try and error solving strategy is minimized.</p> <p>As a serious game designer, I do not need to worry about the adequacy of my motivation measuring method; The latest know-how is easily accessible?</p>
Creator	TUGRAZ
Further info	TUG asset overview page: http://css-kti.tugraz.at/projects/rage/assets/

3.1 Included WP2 assets

3.1.1 Motivation Assessment Asset

General Description

The Motivation Assessment Asset aims at assessing the player's motivation to learn while playing a game. A player's motivational state is a crucial aspect within a game and strongly influences how often the game is played or if it is positively perceived. By analysing interaction within the game confidence, attention and satisfaction as three components of motivation are measured and data relating to this is provided to other assets for further processing, e.g. to the motivation-based adaptation asset (T3.4) for maintaining or supporting motivation.

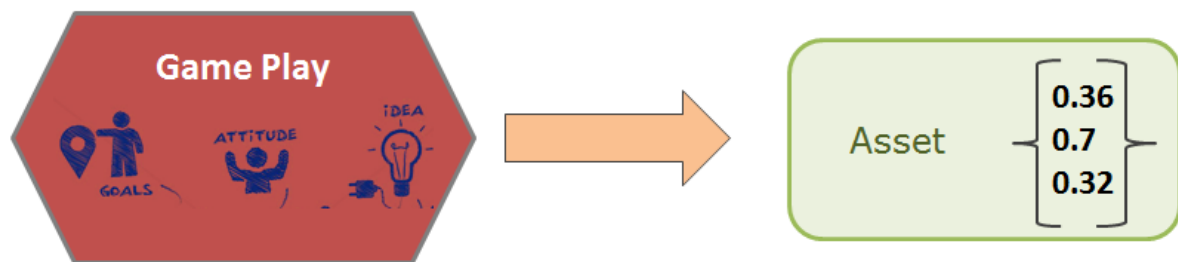


Figure 4. Motivation assessment measures.

Pedagogical Value

The key value is that this asset provides information about a gamer's current motivation to learn and its changing states over gaming episodes. The motivation assessment is done non-intrusively and may be used for adaptation of the game in maintain and enhance motivation.

Why a game developer benefits from this asset

This asset manages player motivation within a game

- Bringing in a psychological model of motivation into game design
- Motivational aspects and their behavioural indicators within the game can be defined
- Motivational assessment rules can be adapted without adaptation of the game

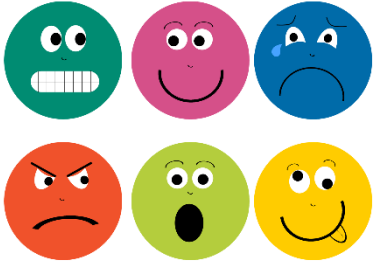
Links

- Sources: <https://github.com/RAGE-TUGraz/MotivationBasedAssets>
- Design: <http://css-kti.tugraz.at/projects/rage/assets/designdocuments/DesignDocument-MotivationAssessmentAsset.pdf>
- User documentation: <http://css-kti.tugraz.at/projects/rage/assets/factsheets/FactSheet-MotivationAssessmentAsset.pdf>
<http://css-kti.tugraz.at/projects/rage/assets/designdocuments/DesignDocument-MotivationAssessmentAsset.pdf>
- Example integration with game: <http://css-kti.tugraz.at/projects/rage/assets/dummygame/DummyGame-Application.zip>
- TUG asset overview page: <http://css-kti.tugraz.at/projects/rage/assets/>

4 REAL-TIME EMOTION DETECTION FROM FACIAL EXPRESSIONS

This asset provides emotion detection from players' facial expressions in real-time. Table 4 represents the factsheet of this asset.

Table 4. Real-time Emotion Detection from Facial Expressions factsheet

	<h3 style="text-align: center;">Real-time Emotion Detection from Facial Expressions</h3> <p>OUNL Assets: Client-Side Real-time Emotion Detection from Facial Expressions</p>
Short description	This asset is a client side software component that can detect emotions from players' facial expressions.
Type of product	Game Component
Main benefit for Game Studio	Gain creator: Facilitates detecting emotions from players' facial expressions. Pain reliever: Allows capturing the players' facial emotional states and incorporates these as variables in games.
Main features for Game designers/developers	This asset detects emotions in real-time and it returns a string representing six basic emotions: happiness, sadness, surprise, fear, disgust, and anger. It can also detect the neutral face. Accuracy is over 80%, which is comparable with human judgment. This asset can be used in games for communication training or conflict management tasks. Or for collecting emotion data during play testing. This asset can be easily integrated in to a variety of different game engines, including, Unity3D.
Main features for Game customer/user	This asset will enrich games, using the player's webcam stream, a single-image file, or a recorded video file as inputs. The software component can manage multiple players, and detect multiple faces and their emotions at the same time.
Creator	Open University of the Netherlands, OUNL
Further info	This asset is available under the Apache-2 open source license, which means that it is free at the point of use, even in commercial applications. URL: https://github.com/rageappliedgame/EmotionDetectionAsset

4.1 Included assets

4.1.1 Real-time Emotion Detection from Facial Expressions

This software component detects the facial expressions of players in real-time and provides detected emotions accordingly.

It works in three different ways to detect facial emotions of players: 1) it can use the player's webcam stream as input in real-time, 2) it can load a single image file, or 3) it can load a recorded video file, and has been implemented in three ways:


- C# version, compliant with the RAGE asset architecture explained in D 1.4.
- Unity3D version (also written in C#, but using Unity3D libraries).
- C++ version, compiled with the RAGE asset architecture explained in D1.4.

Links

- Sources:
 - <https://github.com/rageappliedgame/EmotionDetectionAsset> (C# version)
 - https://github.com/rageappliedgame/EmotionDetectionAsset_UnityDemo (Unity version)
 - https://github.com/rageappliedgame/EmotionDetectionAsset_CPP (C++ version)
- Design Guidelines:
 - <https://github.com/rageappliedgame/EmotionDetectionAsset/blob/master/Design%20Guidelines>
- User's Guide:
 - <https://github.com/rageappliedgame/EmotionDetectionAsset/blob/master/User's%20Guide>
- Example integration with game:
 - https://github.com/rageappliedgame/EmotionDetectionAsset_UnityDemo

5 EASY DIALOGUE INTEGRATOR

Table 5. Easy Dialogue Integrator factsheet

	<p style="text-align: center;">EDI Easy Dialogue Integrator</p> <p>UU Assets: Step based competence assessment Communication Scenario Editor</p>
Short description	End-user-friendly dialogue authoring tool and reasoner for seamlessly incorporating expressive dialogues between virtual character(s) and a player
Type of product	Bundled dialogue creation tools
Main benefit for Game Studio	<p>Gain creator. Easy to use, customer centric designed editor that allows domain-expert users to create expressive ad-hoc (customised) dialogue scenarios</p> <p>Pain Relievers: the expert-user created dialogue is seamlessly integrated in a game, thus reducing costs for expanding usage base</p>
Main features for Game designers/developers	<ul style="list-style-type: none"> • Tested and robust after 3 years of usage in different communication-skills learning environments. • Extensive documentation, game developers from RAGE installed and integrated in their games self-sufficiently. • Easy adaptation to multiple scenarios by empowering programmers to integrate dialogues seamlessly in a game.
Main features for Game customer/user	<p>Easy to use, even for non-programmers. EDI provides the features from most dialogue tools and in addition expressive dialogue constructs. Communication-skills experts develop and modify a scenario independent of a programmer. Runs in a web browser as opposed to existing client-based tools, making EDI more easily accessible to non-professional authors.</p>
Creator	Utrecht University, UU
Further info	<p>https://github.com/UURAGE https://uudsl.github.io/scenario https://github.com/UURAGE/ScenarioEditor/tree/master/doc https://github.com/UURAGE/ScenarioReasoner/tree/master/doc https://youtu.be/rY7mj2my5MU https://youtu.be/-MW8j5EDL6Y</p>

In this document we describe the **client-side asset only**, i.e. **Communication scenario editor**.

5.1 Included assets

5.1.1 Communication scenario editor

The Communication scenario editor is a tool in which a communication expert is able to iteratively develop a communication scenario as a directed acyclic graph of steps. A scenario consists of statements between a player and a virtual character, in which a player chooses between various pre-specified options.

The editor (see Figure 5) balances usability for a non-programming (communications) expert and expressiveness of the constructs in which a scenario can be expressed. With the latest release of the asset, an instance of this editor can be created catering to the specific needs of a game-developer/educator. A game developer can specify virtual character(s), properties (e.g. name of a character) and parameters (e.g. score)

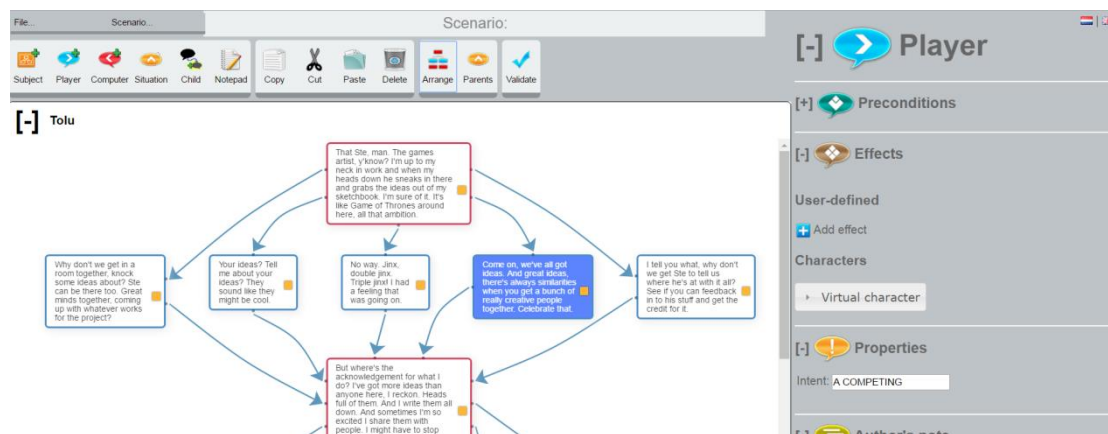


Figure 5. Communication scenario editor.

The editor has the following basic features that simplify game creation:

- Develop a conversation between a virtual character (computer node) and a player as a graph. The graph is from top to bottom in time.
- Choices for a player are as multiple player statements from a computer node.
- Each player statement choice leads to a score, and effects like emotions in the virtual character.
- A statement can have a condition.

With the latest release (v4) of the asset, an instance of this editor can be configured catering to the specific needs of a game developer. A game developer can specify virtual character(s), properties (e.g. name of a character) and parameters (e.g. score) specific to their game needs.

- Scores and emotions can be configured as parameters of scenarios, which can be modified at a statement.
- Properties of a scenario can be configured. For example, you can specify the name of a character.

Conversations can be structured at a higher level of abstractions in subjects. See screen-shot in Figure 6.

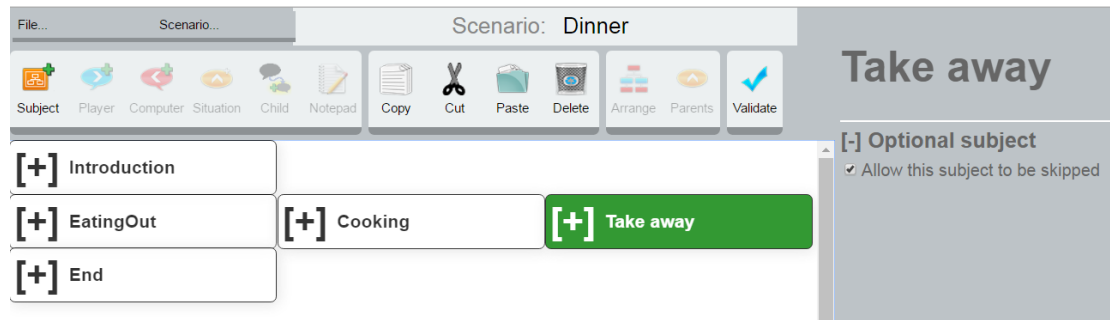


Figure 6. Conversations structure in the communication scenario editor.

A subject on the horizontal level is interleaving with another subject(s) on the same level. In the above figure, 'Eating out', 'Cooking', and 'Take away' are interleaving subjects. During the simulation a player is presented with statement choices from within these interleaving subjects without a predetermined order. This is useful when a player should communicate with a virtual character on multiple subjects, but the order in which statements within these subjects are followed is not important.

A subject may be marked as optional, which means that the subject may be skipped in simulation.

Subject variability is augmented with statement variability. In addition to conditionality in nodes, three features that a scenario author may specify on a computer statement are 'Jump to another subject', 'Early end of scenario' and 'End of scenario'. The interface is depicted in the screen shot in Figure 7.

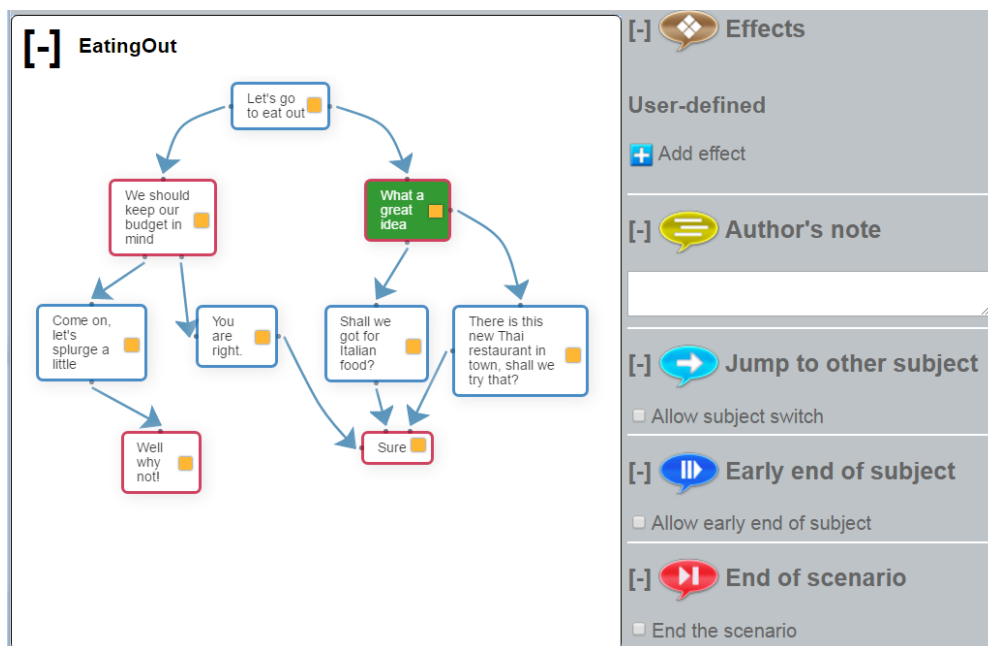


Figure 7. Interface with scenario features.

A 'jump' indicates whether it is allowed to jump from this node to another node in a subject in this interleave level. An 'early end of subject' indicates that it is allowed to go from this node to a node in one of subjects at the same interleave level and if there are no other subjects in the same interleave, that it is possible to jump to a node in a subject in the next interleave. An 'end of scenario' indicates that the scenario is terminated after this statement (thus constituting an end-game state).

A combination of these features allows for variability and expressiveness in a scenario.

A domain expert can then export valid scenarios in XML format. Scripts may be saved locally and imported to modify and continue development. The output of the editor, a dialogue, is stored as an XML file that follows the schema: <https://uudsl.github.io/scenario>

Relation to other asset(s)

The output of the editor can be ‘parsed’ and ‘reasoned’ by another asset: 2.2 Step-based competency asset, aka scenario reasoner asset. We have designed the ‘Editor’ and the ‘Assessment’ assets to be loosely coupled using a REST architecture. The reasoner asset processes any valid scenario produced according to the schema in the scenario language, using any valid configuration (single Virtual Character /multiple VC’s; configurable scores, parameters, properties).

The relation between the assets is shown figuratively in Figure 8. A game can use the reasoner asset to perform dialogue management.

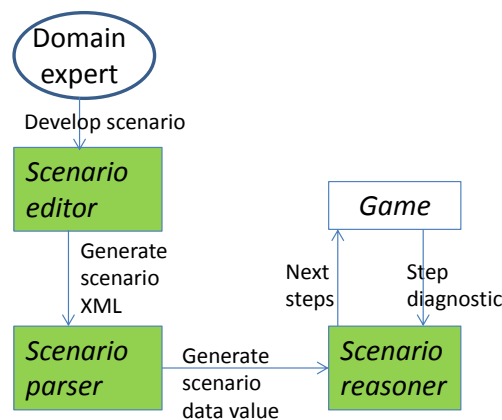


Figure 8. Scenario editor to simplify dialogue creation.

Technical and quality aspects

The Communication Scenario Editor asset is implemented in Javascript and runs on most browsers, eliminating the need for a native client.

Links

- Sources:
<https://github.com/UURAGE>
- Release:
<https://github.com/UURAGE/ScenarioEditor>
- Design:
<https://github.com/UURAGE/ScenarioEditor/blob/master/doc/Software%20design%20document.pdf>
- User documentation:
<https://github.com/UURAGE/ScenarioEditor/tree/master/doc>
- Launcher:
<https://github.com/UURAGE/ScenarioEditor/blob/master/README.md>
- REST API
<https://uudsl.github.io/scenario>

6 SHARED DATA STORAGE COMPONENT

Table 6. Shared Data Storage Component factsheet

image	gsm GAME STORAGE MANAGER OUNL Assets: Game Storage, Client-Side
Short description	Improves a games' data storage management
Type of product	Game component
Main benefit for Asset Developers / Game Studio	Pain Relievers (improvements in game production process/studio profitability) : Provides a versatile storage and persistence of multiple sets of (tree-like) data. Shares integration effort with other rage assets.
Main features for Game designers/developers	Stores multiple treelike data structures. Persists and restore data losses without predefined model classes. JSON and XML support. Structure and data are persisted separately. Data can be persisted and restored to/from multiple storage locations. Supports the Server-Side Game Storage as well as local storage. This asset uses the Apache-2 open source license.
Main features for Game customer/user	n/a.
Creator	Open University of the Netherlands, OUNL
Further info	https://github.com/rageappliedgame/ClientSideGameStorageAsset

6.1 Included assets

6.1.1 Game Storage asset

The Shared Data Storage component is designed for centralized storage of data of other assets as well as the game itself. It offers storage of treelike data structures. Each treelike data structure is accessible by name, each data element has a name. Names of data elements are used to generate the paths of an element.

The Shared Data Storage component offers flexible persistence and is very powerful as the structure is stored in depend of the data itself. The data itself can marked for persistence (serialization) in one of the supported storage locations. Data can be restored (de-serialized) from one or more storage locations in a random order. The only requirement is that the structure of the data is restored before data is being restored.

The Shared Data Storage component also removes the need for model/template classes as it serializes the necessary type info alongside the data. In doing so it does not need these

template classes normally required JSON and XML serialization class for loss-less restoring of data. By offering lossless persistence of data storage, the asset simplifies coding, reduces testing efforts and maintenance.

As storage location for persistence, the Shared Data Storage component supports 4 different locations:

- 1) The Game Storage - Server-Side asset.
- 2) The local file system by using the Bridge as interface.
- 3) No storage (for transient data that does not need to be restored).
- 4) Game based storage (here the component allows for read-only access to values supplied by the game)

As data storage for the local file system, both JSON (Newtonsoft, 2016) and XML formats are supported. The Game Storage – Server-Side asset requires Json. Game based storage requires the implementation of a simple interface on the asset's Bridge. Examples of game data made accessible are player id, game time, current level etc.

The Shared Data Storage component supports Unity3D, Xamarin and other C# based game development platforms. The RAGE client-side architecture simplifies integration into games as assets using this architecture share integration efforts.

Links

- Sources:
<https://github.com/rageappliedgame/ClientSideGameStorageAsset>


7 REAL-TIME AROUSAL DETECTION USING GALVANIC SKIN RESPONSE

The package includes an asset for real time detection of player arousal (excitement) based on measuring galvanic skin response (GSR), referred as well as electrodermal activity (EDA). The asset provides the following core functionalities:

1. Analysis of player's electro-dermal activity based on GSR responses:
 - player responses are received generally from any GSR measuring device (an affordable custom device has been designed to work with the asset)
 - scalable animated visualization of the GSR signals within a time window and using a sampling rate both chosen by the game developer
 - various signal features are extracted in real-time and returned in a JSON response (Crockford, 2006)
2. Real-time detection of player's arousal inferred based on the GSR signal.

The asset recognizes in real-time phasic (event-driven), tonic (base level), and total arousal of an individual player by analysing various features of the GSR signal. Its settings include number of arousal quantification levels, time window, sampling rate, and log file.

Table 7. Real time arousal detection using GSR asset factsheet

	<h3 style="text-align: center;">Real time arousal detection using GSR asset</h3> <p>SU Asset RT arousal detection using GSR</p>
Short description	Recognizes in real-time phasic (event-driven), tonic (base level) and general arousal of an individual player by analysing the GSR signal
Type of product	Game component
Main benefit for Game Studio	<ul style="list-style-type: none"> • Gain creator: Allows real time detections of player arousal from player's electro-dermal activity. • Pain reliever: Facilitates capturing of changes in players' emotional states and incorporates these as variables in games.
Main features for Game designers/developers	<ul style="list-style-type: none"> • Game designers can know and view how the player arousal changes based on objective data extracted while playing your game, in order to adapt the gameplay • Game developers can apply the asset to track the arousal of an individual player in real time and use it for an easy and very fast tailoring the mechanics, dynamics and/or aesthetics of your game.
Main features for Game customer/user	<ul style="list-style-type: none"> • For players: Smoother learning experience via a balanced modulation of difficulty tailored to the player's changes of arousal. • For instructors: The asset can be applied in educational games for a game-based learning or training, which is personalized and adapted according to the user arousal.
Creator	Sofia University, SU
Further info	https://github.com/ddessy/RealTimeArousalDetectionUsingGS

7.1 Included assets

7.1.1 Real time arousal detection using GSR asset

The real time arousal detection using the GSR asset estimates dynamically a player's excitement using its proven linear dependency on the level of electro-dermal activity and, thus, provides a base for the adaptation of content, task difficulty, audio-visual effects and other features of a video game.

The asset reports the total arousal level, baseline (tonic) arousal level reflecting general long-lasting changes in autonomic arousal, and event-driven (phasic) arousal level – all from 0 to N-1, where N is designer-defined value in asset settings. Together with these levels of arousal, the assets return various GSR signal features as a JSON response. As well, the asset allows an optional scalable visualization of raw and filtered signals.

The skin conductance (GSR) is measured by an inexpensive custom device (though the asset can use any GSR measuring device) with programmable sampling rate from 8 up to 800 Hz and down-sampled eight times in order to reduce possible noise. The device applies two sensors attached to bottom of fingertips using Velcro straps wrap around fingers as shown in Figure 9. The asset provides real-time analysis of features of GSR signal measured from a player for a customizable time window (with default duration 10s) at chosen sampling rate (default rate is 8Hz). Bothe the raw and filtered signals can be optionally visualized (Figure 10) with possible scalability on the horizontal and vertical axe.

A calibration period is not required but is strongly desirable before starting calculation of player arousal level, in order to produce data that are more accurate.



Figure 9. Usage of a GSR measuring device

The level of arousal is determined in real time, on a scale from zero up to maximum arousal level set by the asset user. This may be useful for fast and easy detection in changes of emotional state of the player and for adaptation purposes, therefore the asset can be combined with the T2.3 Real-time Emotion Detection Asset and the T3.4 Player-centric rule- and pattern-based adaptation asset.

The asset is implemented as a C# client component. It can also be used through a socket, but the GSR device has to be plugged on the client side. Its run-time requirements include Windows Vista or higher and Microsoft .NET Framework.

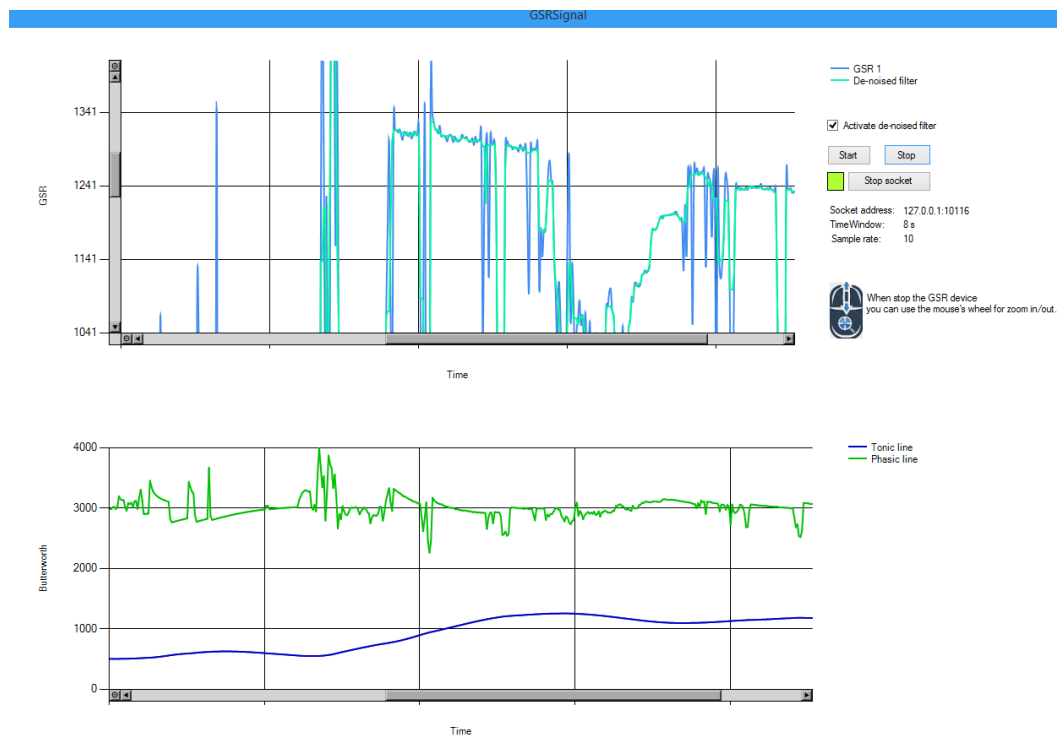


Figure 10. A screenshot of the visualization application with zooming of raw and filtered signals.

Links


- Project overview page:
<https://github.com/ddessy/RealTimeArousalDetectionUsingGSR#overview>
- Deployment instructions:
<https://github.com/ddessy/RealTimeArousalDetectionUsingGSR#deployment-and-usage-instructions>
- Tutorial:
<https://github.com/ddessy/RealTimeArousalDetectionUsingGSR#instalation-and-usage>
- Executable demonstration:
<https://github.com/ddessy/RealTimeArousalDetectionUsingGSR/tree/master/RealTimeArousalDetectionUsingGSR/DisplayGSRSignal/bin>
- Demonstration's source-code:
https://github.com/ddessy/RealTimeArousalDetectionUsingGSR/tree/master/Demo/Access_RealTimeArousalGSRDetection_BySocket
- Demonstration's documentation:
<https://github.com/ddessy/RealTimeArousalDetectionUsingGSR/blob/master/Docs/field-trial2-v04.pdf>
- Deployment documentation:
<https://github.com/ddessy/RealTimeArousalDetectionUsingGSR#deployment-and-usage-instructions>
- Platform requirements:
<https://github.com/ddessy/RealTimeArousalDetectionUsingGSR#technical-details>

8 MULTIMODAL EMOTION DETECTION

This asset can detect the user's emotional and affective state by integrating multiple emotion detection modalities and combining them into a single classification. It incorporates 5 different modalities: facial expression recognition, text recognition, speech recognition, touch recognition and Electrodermal Activity/Galvanic Skin Response (GSR/EDA) recognition for arousal. It provides the following core characteristics:

- Runs with any number of emotion recognition modalities (so game developers are not required to use all modalities), on a when needed basis.
- Simple and straightforward way of integrating and configuring the several emotion detection assets. It provides standardization and normalization of emotional labels and values from the multiple modalities.
- Three fusion policies can be used to merge emotion detection information.

Table 8. Multimodal emotion detection asset factsheet

	<h3 style="text-align: center;">Multimodal emotion detection asset</h3> <p>INESC_ID Asset Multimodal Emotion Detection</p>
<p>Short description</p>	<p>Combines and integrates facial, textual, speech and GSR emotion detection to provide a more complete description of the user's emotional and affective state.</p>
<p>Type of product</p>	<p>Game component</p>
<p>Main benefit for Game Studio</p>	<ul style="list-style-type: none"> • Gain creator: Allows more robust and complete detections (including real-time components such as arousal and facial emotion detection) of user's emotions. • Pain reliever: Facilitates the integration of several emotion detection assets.
<p>Main features for Game designers/developers</p>	<ul style="list-style-type: none"> • Game designers can easily conduct experiments with beta-testers, and use text-based sentiment analysis and other modalities to automatically determine the user's enjoyment of the game. • Game developers can apply the asset to track the emotional state of individual players in real time, and adapting the game experience accordingly
<p>Main features for Game customer/user</p>	<ul style="list-style-type: none"> • For players: A better game experience by adjusting the user's experience to its emotional state. • For instructors: The asset can be applied in educational games for a game-based learning or training, helping select better pedagogical strategies when the user is feeling frustrated.
<p>Creator</p>	<p>Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento, INESC-ID</p>
<p>Further info</p>	<p>https://www.dropbox.com/sh/2qdsnxq1rpi3xb3/AAAXk2bQ8sxaJ-t2Uf65M789a?dl=0</p>

8.1 Included assets

8.1.1 Multimodal Emotion Detection Asset

This asset is designed as a library that runs directly in the client application and communicates using web services with server-side Emotion Recognition Assets (e.g Speech and Text Emotion Recognition Assets).

This asset is prepared to run with any number of emotion recognition modalities, and their configuration and incorporation into the classification fusion process is straightforward. It is important to point out that this asset corresponds to the asset previously identified as 2.3a from deliverable D2.3. A decision was made to change its name to focus on the property of multimodality instead of the real-time. This is in part because this asset can only assure the real-time requirements for the modalities that support it (facial emotion detection and GSR). For instance, although the speech emotion detection is able to run online, the delay between the speech and its classification is too large (around 4 seconds) to be considered real-time.

In the current release, four emotion recognition modalities were integrated: the Sentiment analysis asset from UPB, the speech emotion recognition asset from INESC-ID, and an internally developed asset for Electrodermal Activity (EDA) recognition which will be later replaced by the Real-Time Arousal Detection asset from TUSofia, and finally the Facial Real Time Emotion Detection asset from OUNL. Three classifier fusion policies were implemented: max, weighted average, and Kalman fusion. The max policy just uses the maximum value of its classifiers for predicting a specific emotion, while the weighted average policy takes into account a weighted average from all classifiers contributing to a particular emotion. The Kalman fusion policy is the most complex, and it is based on the system proposed in (Glodeck et al. 2013) using a Kalman filter to combine multiple sensors. The main advantage of this policy is that it can more easily deal with situations where one of the sensors is not currently active, by using information from past observations and explicitly assuming a higher uncertainty about these observations.

Figure 11 shows a snapshot of the demo created to showcase the multimodal emotion detection. It shows in top the graphical interface from the Real-Time Emotion Detection from Facial Expressions asset, together with information (bottom, left to right) from the Text Recognition, Speech Recognition, and EDA Recognition. The bottom right side shows the combined classification of all modalities. As seen in the example, both the facial detection and text recognition detect an emotion of happiness (albeit lesser in the text). The Kalman filter fusion provides an intermediate value for the happy emotion. The speech recognition asset is not active, and thus is not influencing the final classification.

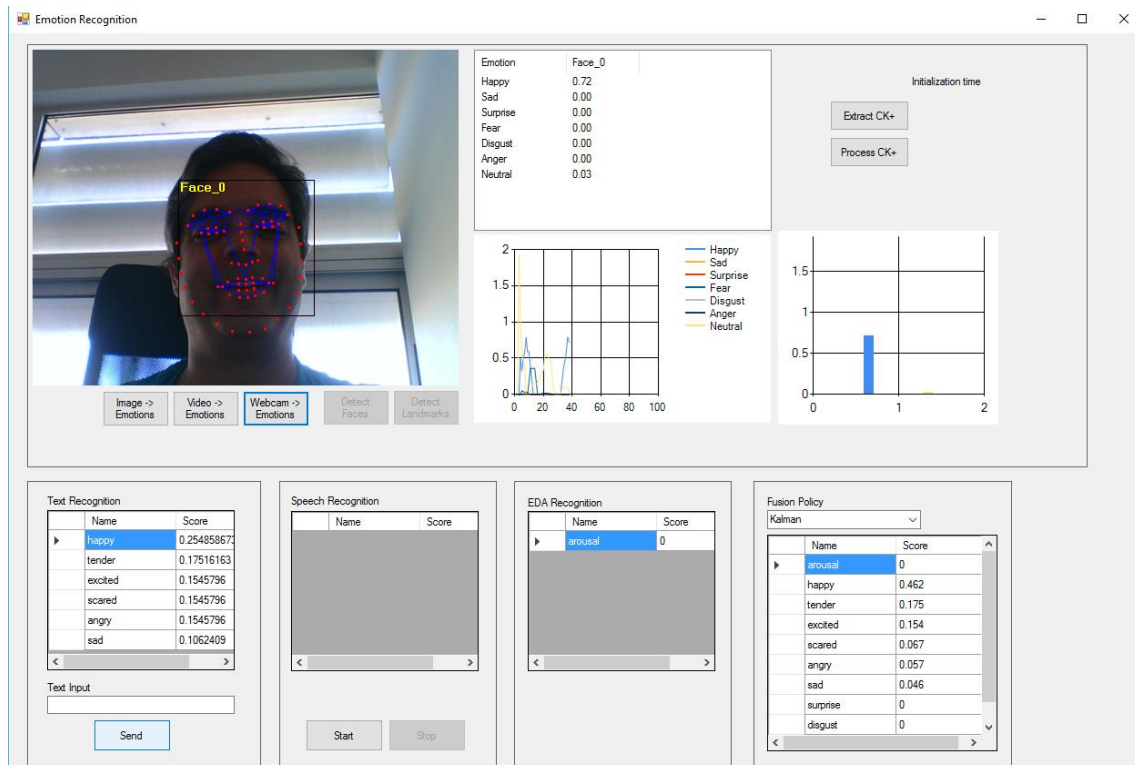


Figure 11. Snapshot of the Multimodal Emotion Recognition demo, illustrating the integration of facial, text, and speech recognition

Links

- Sources:
https://github.com/GAIPS-INESC-ID/FAtiMA-Toolkit/releases/tag/MultimodalEmotionDetectionAsset_Release1_03
- User documentation:
<https://www.dropbox.com/sh/2gdsnxg1rpi3xb3/AAAXk2bQ8sxaJ-t2Uf65M789a?dl=0>
- Example integration
<https://www.dropbox.com/sh/2gdsnxg1rpi3xb3/AAAXk2bQ8sxaJ-t2Uf65M789a?dl=0>

9 CONCLUSIONS

This deliverable describes the final catalogue of WP2 client-side assets produced by RAGE. All of the assets are fully functional and include the complete documentation required for adoption and use by the game industry.

The assets are all open software and are in continuous evolution for corrective maintenance and to respond to clients' new requirements or needs. These changes will be both in documentation and code at the open repositories of the assets. Therefore, those assets will be continuously improved by their creators till the end of the project (the project has decided to extend that WP task till month 48) and beyond.

REFERENCES

(Crockford, 2006) Crockford, J. (2006) *The application/Json Media Type for JavaScript Object Notation (JSON)*, Fremont CA, Internet Engineering Task Force. Retrieved from <http://www.ietf.org/rfc/rfc4627.txt>.

(Freire, 2016) Freire, M., Serrano-Laguna, Á., Iglesias, B.M., Martínez-Ortiz, I., Moreno-Ger, P., Fernández-Manjón, B.: *Game Learning Analytics: Learning Analytics for Serious Games*. In: Learning, Design, and Technology. pp. 1–29. Springer International Publishing, Cham (2016).

(Glodeck et al. 2013) Glodeck, M., Reuter, S., Schels, M., Dietmayer, K. & Schwenker, F. (2013). *Kalman Filter Based Classifier Fusion for Affective State Recognition*. In *Multiple Classifier Systems, Lecture Notes in Computer Science 7872*. Springer.

(Newtonsoft, 2016) Newtonsoft Json.Net (2016) *Json software framework* available at <http://www.newtonsoft.com/json>.

(Serrano, 2017) Serrano-Laguna, Á., Martínez-Ortiz, I., Haag, J., Regan, D., Johnson, A., Fernández-Manjón, B.: *Applying standards to systematize learning analytics in serious games*. *Comput. Stand. Interfaces*. 50, 116–123 (2017).