FP7- 257666-eCute
(month 12, 2011)

Education in Cultural Understanding, Technologically-Enhanced
Collaborative Project (ICT-2009.4.2)
Technology-enhanced learning

Start date of project: 01/09/2010
Duration: 36 months

(D6.1). (Generic Integration Framework specification and Design)

Due date of deliverable: 31/8/2011
Actual submission date: 21/10/2010

AUTHORS: Christopher Ritter, Asad Nazir
CHECKERS: Ruth Aylett

STATUS: [Final]

Project co-funded by the European Commission within the Seventh Framework Programme

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>PU</th>
<th>PP</th>
<th>RE</th>
<th>CO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CO</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Deliverable <D6.1/final>
TABLE OF CONTENTS

1. Purpose of Document: .............................................................................................................. 3
2. Executive Overview: ................................................................................................................ 4
3. Motivations and Requirements for an Integration Framework .................................................. 5
   3.1 Motivation .......................................................................................................................... 5
   3.2 Generic Requirements for an Integration Framework .......................................................... 6
4. Existing Frameworks ................................................................................................................... 8
   4.1 FearNot! Architecture: ....................................................................................................... 8
   4.2 Orient Architecture: .......................................................................................................... 9
   4.3 Lessons Learned .................................................................................................................. 10
      4.3.1 Game Engine Development: ....................................................................................... 10
      4.3.2 ION-Framework: ......................................................................................................... 11
      4.3.3 FAtiMA Agent Architecture: ....................................................................................... 12
      4.3.4 Other Stuff: ................................................................................................................. 12
5. eCute Integration Framework ..................................................................................................... 13
   5.1 Important Components: .................................................................................................... 13
      5.1.1 Theoretical Framework and Associated Learning Goals ............................................. 13
      5.1.2 Evaluation Goals ......................................................................................................... 14
   5.2 The Framework .................................................................................................................... 14
      5.2.1 ION Framework .......................................................................................................... 15
      5.2.2 Output modalities ....................................................................................................... 16
      5.2.3 Unity3D - Game Engine ............................................................................................... 16
      5.2.4 Unity3D - Virtual Agent Embodiment ........................................................................ 18
      5.2.5 Input Modalities .......................................................................................................... 20
6. Conclusion ................................................................................................................................... 22
7. References ................................................................................................................................... 23
1. Purpose of Document:

This deliverable comes as a result of the objective 6.1 which consists of combining the theories as defined in WP2, the designed cultural characters in WP5 and the cultural based interaction as defined in WP4. This document provides the integration framework which will be used to combine the above mentioned components into two showcases for two different user groups.
2. Executive Overview:

Overview of the entire document contents:

This deliverable describes the requirements and design specification of an integration framework that will be used in the project to develop two showcases for two age groups. This document begins by giving an overview of the requirements and motivation for such an integration framework, we then briefly describe two existing frameworks which become the guideline for this new framework.

Based on the requirements and existing approaches we then come up with the proposed framework where we define different components to be integrated, how the communication between the components work and how they are connected using an integration entity.

This document should be read in conjunction with deliverable 2.1, 4.1 and 5.1, as it only describes the integration of components and the details of those components can be found in the above mentioned deliverables.
3. Motivations and Requirements for an Integration Framework

3.1 Motivation

The eCute project focuses on exploring the use of AI based software for the development of educational software supposed to teach about social concepts. The envisioned systems provide a motivating interactive environment where the user learns through observation and interaction with artificially intelligent virtual characters. Main motivation for this approach comes from the successful use of real life role-play games to teach social phenomenon. This shares some strong similarities with the Serious Game approach, where game like environments are used as transportation medium for embedded educational content. The main difference between a typical Serious Game application and the intended outcome of the eCute project is that the element fun is not seen as main motivator for the user to interact with the system. In contrast it is intended to use an interactive virtual drama approach to achieve deep emotional engagement of the user with the acting virtual characters.

To achieve these goals it is pertinent for both the characters and the virtual environment to meet certain requirements. These requirements in theory include emotionally engaging behaviour of the character, the users’ strong involvement into the role-play and also the user’s ability to intuitively interact with the characters and the environment.

The eCute project will produce, among other research related artefacts, two interactive learning environments (showcases) that will both be used for evaluation of the associated learning goals. One Showcase will address children at the age between 9 to 11 years and is called MIXER (Moderating Cross-Cultural Empathic Relationships). The other application addresses young adults at the age between 18 to 25 years and is referred to as TRAVELER.

MIXER showcase will target conflicts arising from group level differences. It will be based on the moral circle theory which suggests that people identify themselves with a particular group that exhibit a set of moral traits of equal standards. The mixer scenario tries to focus on increasing competences of the children to learn from such differences through exploration. Affective empathy towards the acting characters (invisible friend) will be the main motivator for the user’s interaction. The goal is to enhance the user’s competence in conflicting situations in terms of cognition, emotion and behaviour (deliverable 2.1 FP7 ecute project)

TRAVELLER showcase for the project is targeting young adults. The scenario is more complex and the focus being on cultural differences rather than just in-group and out-group behaviour. In the scenario for the children there is a focus on the extension of their moral circle. With the young adults the aim is a more conscious recognition of the ways in which cultures can differ. Not only do we have to focus on getting to know groups with different values, but we also have to be able to communicate effectively with them. In this the user becomes a more active participant in the role play. Interacting with the virtual world and
characters by using different interaction devices such as the Kinect sensors, IPADs etc.

3.2 Generic Requirements for an Integration Framework

To realize these two showcases a technical integration framework will be required. In the following paragraph a high level overview of the estimated basic components will be given. After that there will follow an overview over two frameworks that were considered for reuse in the eCute project. Both the FearNot! framework and the Orient framework developed as part of two earlier EU FP5 and FP6 projects (VICTEC and eCircus) will be reviewed as the produced showcases of the accompanied projects show some technical similarities to the intended outcomes of the eCute project.

1. The Basic Components required for an integration Framework:
   a. Autonomous agents:
      A basic component which is used to model the habitants of the virtual world as virtual characters. The user interacts with them by acting out role-plays. The artificial characters are required to simulate emotions for achieving emotional engagement through empathy.

   b. Output - Graphical modalities
      Generates an audio visual representation for the simulated world including the acting characters. An appropriate appearance of the characters and the virtual world is required to improve the user immersion. To support emotional engagement of the user the characters’ embodiment must include expressive behaviour and a supporting representation of environmental variables.

   c. Output - Speech System
      It is required to have some output through speech i.e. the characters should be able to utter sentences during the role-play.

   d. Input/interaction Modalities
      A user should be able to interact with the virtual characters in a virtual world through the most intuitive means. This is important because it ensures minimal distraction from the actual role-play, which is important to achieve the targeted learning goals.

2. Integration Framework
   a. World Simulation
      Once we have all the above components it is of utmost importance how these
components are integrated into one system. It’s not just that these components are present in one place but also how they communicate amongst themselves and also with the environment.

b. Story Facilitator
In educational software it is important that the accompanying narrative supports the goals of the overall application. Emergent narrative based on autonomous characters may conflict with this demand in some cases. For these cases a story facilitator agent can be used to influence the story to emerge into a preferred direction by triggering corrective world events. This assures that the autonomy of the characters is not broken while shaping the story towards a intended outcome gets more feasible.
4. Existing Frameworks

Orient and FearNot! Architectures were both developed as part of an earlier EU projects eCircus and VICTEC. Parts of these architectures have been proven to be very reliable and useful. As such these parts should be reused as much as possible.

Reusing some of the existing components of the FearNot! And Orient architectures will save huge amount of effort and this effort can be redirected towards further developing the AI components and other parts of the framework that have shown to be critical in the precious projects. FATiMA agents are the most valuable component of these two architectures and provide a very powerful and workable solution for designing authored autonomous agents.

Existing FearNot! And Orient Architectures were designed to have the ability of combining pedagogically motivated emergent narrative produced using both autonomous agents and user input. This is also a goal of the eCute project.

In the following section the most important aspects of both architectures shall be reviewed in a bit more detail (strengths and weaknesses). Afterwards the architectures will be compared and an argument will be given why some parts of the have to be redesigned for eCute and which Components are going to be reused.

4.1 FearNot! Architecture:

The FearNot! showcase was developed throughout the VICTEC project and improved during the eCircus project. FearNot! provided a emotional engaging application that successfully increased young children’s ability to recognize the dimension of physical and relational bullying. Autonomous characters running the FATiMA agent generate affective behaviour for telling episodic stories about bullying[4]. The user observed and advised the active characters acting out an invisible friend. To control the emergent narrative it utilizes a story facilitator. This is very important to keep the story within certain boundaries that are determined by the underlying learning goals. The story facilitator [3] keeps check on the emergence and tries to guides the story. The following diagram shows the basic Architecture as defined in [3]:

Deliverable <D6.1/final>
The architecture of FearNot! consists of three layers: The application layer, the world model and the Graphics layer. Application layer combines the user-interface (output modality), world model (integration component) with the FATiMA agent architecture. FATiMA (FearNot Affective Mind Architecture) is the architecture used by the affective characters in FearNot! [5]. The application layer also initiates a story facilitator agent [3]. The world model is build upon the first iteration of the ION framework [1] which includes symbolic representations of all entities in the application. The ION framework is used to create an abstract simulation of the world. The last layer mainly consists of a graphic-focused game engine and the graphical content. As the evaluation was focusing on school environments the input modalities were kept simple. Mouse and keyboard based control paradigms were used. The application integrated these into the graphical game engine as there was no necessity to provide other devices.

The architecture of FearNot has significantly evolved over the years, though the basic underlying concepts did not change much. Since the introduction of the ION framework it provides better synchronisation capabilities between the individual components of the framework. New mediation mechanisms improved the synchronisation between the overall simulation and its asynchronous components (user interaction, agents). Most of the components were iteratively upgraded based on problems faced during actual implementation. So for example the graphical embodiment of the agents and the simulated world environment was firstly based on Wild Tangent (www.wildtangent.com) Engine. Later this option was replaced by Ogre3D graphics engine (www.ogre3d.org) because of better scalability, performance, ease of use, licensing and support. (Deliverble 3.2.1 FP5 Victec Project)

4.2 Orient Architecture:

Most of the experience made with FearNot! encouraged the use of a similar approach for other applications. ORIENT (Overcoming Refugee Integration with Empathic Novel Technology) was the second showcase built in the EU FP6 eCircus project. Orient was designed to be a more role-play game like-9-experience. Focus was set on group
interaction. This was a key difference compared to the invisible friend idea of FearNot!

Group based role-play game activity naturally leads to more complex requirements for the technical integration framework. Especially the graphical realizer had to support many more tasks than the FearNot! one. A complete game engine had to be built on the basis of Ogre3D. As of the good experience it was decided to use a similar approach based on integrating the different components using the ION framework. Some components were modified and updated others had been completely redesigned. The FAtiMA agents had motivations modelled into them and also a cultural dimensions model had been added. Orient also integrated new and innovative interaction devices as the Wii mote, the dance mat and others. Figure 2 shows a high level schemata of the resulting Orient architecture[FP6 eCircus Project Deliverable 4.2.1-8.7.1].

---

Figure 2: Orient Architecture

### 4.3 Lessons Learned

Though both architectures served different (evaluation) goals and used quite different design principles they ended up sharing some very strong similarities. In eCute we will use this knowledge to reduce the development time by creating a common solution used for both showcases:

#### 4.3.1 Game Engine Development:

In the cases of FearNot! and Orient (game) content management (textures, models, animations, etc.) were not very well-supported by the graphics options available.
Though Ogre3D, among all candidates, provides a solid and professional implementation of a real-time graphics engine it does not care about actual content management. These things are supposed to be implemented by the application itself. For both FearNot! and Orient such content management frameworks (game engine) have been developed based on ION. An initial idea was to simply merge these frameworks for direct reuse in eCute. But because of their development histories these frameworks have grown quite differently. Additionally their current complexity makes it nearly impossible to merge them easily. The alternative option of reusing both frameworks as they are means to develop and to maintain two independent frameworks. This is not feasible in means time and manpower.

Another aspect of the previous frameworks is that because of Ogre3D the frameworks do not directly support standard content production pipelines. Such a pipeline is defined as any set of external content production tools (e.g. 3DSMax, Photoshop, etc.) that need to seamlessly interact with the implementation framework to provide the e.g. graphical content used by the final application. Ogre3D unfortunately relies on non-standard file-formats. For common content production tools specialised converters and exporter are provided, but of the special requirements of these converters the resulting content is bound to be used in Ogre3D based applications. Also these converters are mostly supplied by community members and thus show no regular support and tend to be buggy. Such a framework is tedious to extend and the integration of new content or update of existing is difficult at least.

Another aspect that has been shown to be very cumbersome and time consuming is the implementation of the scenarios itself. Ogre3D, just as most other open source graphic/game-engines, provide no build-in editor environments for defining scenario content like e.g. level-design and character positioning. In FearNot! most of the content was specified by plain text files. For Orient some simple editors were written to ease the burden of content management and scenario definition. But because it was not central to the associated project these editors ended up being quite unintuitive and error prone. Extending and maintaining them might be possible, but will be time consuming and is not focus of the eCute project as well.

In eCute we will try to address all these problems together by recreating parts of the previous frameworks using the Unity3D game engine in replacement for Ogre3D. Unity3D (www.unity3d.com) is a full featured, very flexible game engine available for free. It delivers a descent optimized graphics engine, but also the integration of game based physics simulations, sound (3D stereo), full network support and other standard features while maintaining full platform-independency. Moreover it seamlessly integrates all standard content creation pipelines by direct support of according standard file formats. It also delivers a very intuitive WYSIWYG editor environment.

4.3.2 ION-Framework:

The ION Framework has proven itself to be a very attractive way to model a fully synchronized world-simulation for the agents. It has proven to be very flexible and thus applicable under very different circumstances without loss of performance. In the predecessor projects it managed to also integrate a lot of different input modalities, to provide a
basis for any entity-component based game architecture and to integrate with a variety of different graphic solutions. This flexibility is of particular interest for the eCute project, as for example the TRAVELER scenario will explore very different input modalities from the MIXER scenario. The ION framework has been partially redesigned to be more intuitive while maintaining flexibility. The ION framework will be used as a central integration component in the eCute project.

4.3.3 FAtiMA Agent Architecture:

FAtiMA agents have shown to be very suitable to model autonomous empathic characters, a key point in any application that strives for deep emotional engagement based on empathy. The agent architecture has been refactored and significantly extended since. Further improvements will be carried out throughout the eCute project (see deliverable 5.1). A still remaining drawback of FAtiMA agents is the lack of intuitive editors and tools available for defining the knowledge base, the emotional reaction rules and planning domains of these agents. The current solution is based on text-files and the usage of an XML based syntax. This won’t change within the eCute project as designing and implementing such authoring environments is neither trivial nor within scope of the eCute project. The hope is that the advantages of using Unity3D speed up general scenario definition tasks and the implementation of the showcases leaving more time to authoring the agents.

Some other issue with the agents was that the visual embodiment of the agents’ character didn’t match the emotional potential of the agents itself. Because of that it is assumed that a lot of empathic potential was lost. The reasons were determined to be insufficient expressiveness and visual ambiguity. This made it hard – but not impossible - for the user to decode the emotional state of the character. Especially children showed problems in reading the emotional state of a characters in Fear!Not. In FearNot! emotional expressions were based on discrete switching facial textures. This achieved therepresentations of simple comic-like facial expressions, with all its advantages and glitches

4.3.4 Other Stuff:
The concept of an input device server to decouple input devices from the implementation of the actual game application has be found to be very useful and the concept will be take over into the eCute framework. Also the concept of a story facilitator agent to shape the emergent narrative produced by the characters will be taken over.
5. eCute Integration Framework

Here we describe the eCute Integration Framework. We start by identifying the integration components:

5.1 Important Components:

The following is a summary of the identified important components that will have to be integrated in the eCute integration framework. Parts of it are under current development others are to be realized or by reusing artefacts from previous projects.

5.1.1 Theoretical Framework and Associated Learning Goals

In a game like approach to education as anticipated by the eCute project it is not only important to take care for technical requirements but also for requirements introduced by integrating the anticipated learning goals with the game related aspects. Traditional game development requires the early definition of e.g. motivating story background, design and basic game-play mechanics to provide a solid base for an initial implementation of the underlying architecture (what has to be implemented). Learning goals might conflict with this as for e.g. applicable concepts and models might first have to be developed and evaluated throughout the project, before they can be applied to the final application. This requires lot of flexibility and extensibility to be provided by the integration framework. Expected changes should require a minimal amount of work ensuring that a large parts of the already crafted code base can be reused. Entity-Component based implementation paradigms [1] are a well known way of increasing reusability and flexibility of code. As such this paradigm is recommended in Massive Multiplayer Online Games development, where the requirements to the application will change even throughout the runtime of the system and maintenance is known to be one of most costly aspects\(^1\). In eCute the whole integration framework will be centred on the component based paradigm to improve reusability and flexibility as much as possible.

The serious game approach may also require drafting quick prototypes and preliminary scenarios to evaluate new or uncertain concepts and methods. In such cases it is very important, that the integration system is easy to use and provide error-tolerant concepts for scenario definition and prototypical implementation. It is also useful if the created framework can be used (or easily learned) by non expert users, so that project partners that mainly deal with theoretical concepts or evaluation might be able to rapidly create their own test scenarios for local small/preliminary evaluation of ideas. This is a well known problem in interdisciplinary projects like serious games [].

\(^1\) http://www.gamasutra.com/blogs/MeganFox/20101208/6590/Genre_Engines_101_The_EntityComponent_Model.php
5.1.2 Evaluation Goals

It is important to recognize that the technical implementation framework is also influenced by external factors like evaluation goals and target groups. Some evaluation may require the application to track and lock user interaction and/or application internal variables. Logged data must be written into a database for structured evaluation. This must be considered in the integration framework. Another aspect is the target group used for evaluation. TRAVELOER will be evaluated with young adults. As these will mainly be students, evaluation will be carried out at universities. In such environments the hardware used to run the application can be chosen quite freely. In the MIXER scenario instead the evaluation will focus on 9-11 year old children and will be carried out in schools. The experience made in FearNot! showed, that the hardware in such environments will differ very strongly in both performance and availability. To do successful evaluation in such environments it is important, that the implementation provides as much hardware independency as possible.

5.2 The Framework

Figure 3 shows a schematic high level overview of the eCute integration framework. All major components will be described in the following paragraphs.

![Figure 3: eCute Technical Integration Framework](image-url)
Fatima Agents

FAtiMA agents are based on the naming agent architecture. FAtiMA models agents that simulate emotions based on the OCC model. The FAtiMA agent architecture has grown very large as new components have been added to support cultural dependent behaviour for example. Recently the architecture has been further refactored to be more modular. So the architecture can be personalized to the needs of a particular purpose by removing or adding single components. For more details on the new modular FAtiMA architecture see [Deliverable 5.1 of this project]. Each of these agents is simulated in an independent threat (application) running in parallel with the rest of the framework. By default each agent is thus an isolated, continuously planning entity. That’s why they need to be connected to a world model that decides over the results of action the agent is performing and synchronizes the agents alongside.

5.2.1 ION Framework

The ION-framework (Figure 4) is a generalized, discrete time step and fully synchronized simulation framework. It is totally based on an entity-component design paradigm. In an ION-simulation everything is defined to be an Entity (an identifiable object), a Property or an Action. Entities can be connected to Properties and Actions to describe more complex Entities. Properties are plain variables that only allow for synchronized access and atomic modification. Actions are Properties that change their values or modify Properties over time.

Actions can be Started, Paused, Resumed or Stopped. As such they describe behavioural aspects of an Entity. ION has undergone refinement since eCircus. The current version is much more flexible and more intuitive to use. In the context of eCute it will serve again as the central integration component that also holds an abstract simulation of the world the agents are interacting in. As such there is no much difference compared to previous projects.
5.2.2 Output modalities

The use of the Unity3D game engine is one of the most impacting changes in the integration framework if compared to previous projects. But the advantages of a fully featured, professional developed commercially used game-engine paired with the flexibility Unity3D provides outpaces the effort of redesigning or maintaining existing frameworks. In the following paragraphs we will name the main responsibilities within the framework explaining the motivations for that decision.

5.2.3 Unity3D - Game Engine

Unity3D will be used as basis for the Realizer-component in eCute. The Unity3D game engine will deliver a visual representation of the virtual world simulated in ION. It will also provide other services that cannot be supported by the abstract world representation of the ION-framework e.g. line of sight checks, navigation requests, etc.

Unity3D is completely developed under the entity-component based paradigm. In Unity3D every game object is an abstract entity. Entities aggregate behavioural components. Such components model behavioural facets of the entity they are attached to. Components are written according only to the behaviour they are supposed to realize not with respect to the entity they are attached to. E.g. a component may realize the navigation of any entity through the environment. As such components can be shared between entities and reusability increases.

Unity3D provides a large range of prebuilt scripts for most common problems in general game development. All such predefined behaviour can be extended or replaced by user defined scripts. As such Unity3D provides a game engine that can be personalized. Only very low level functionality like the rendering core cannot be overwritten in the freely available version of Unity3D. This requires an advanced license model. But a careful review of the supported features showed, that the free version of Unity provides enough flexibility and functionality for the realization of the eCute and alike showcases. Further all not provided functionality relies on special hardware requirements and should not be part for the basic framework.

Unity3D provides a fully featured content pipeline. Most current standard file formats for 2D and 3D graphic content and sound content are directly supported. No exporters or importers are required. Unity3D supports commercial tools like 3DStudio Max or Maya but also freely available options like Blender natively. eCute wants to produce open source content. As such Unity3D does not require interested followers to work with the same tools as the developers did. Additionally a Unity3D project is just a plain folder structure. Files can be inserted, modified and replaced directly from the operating system’s view or by the according tools. All made changes will be automatically registered by Unity3D runtime environment
and immediately applied to the project. This supports rapid prototyping and iterative refinement of components and graphical content. Also all predefined import behaviour (e.g. loading of 3D models) can be replaced or extended by user defined scripts if necessary. But for now there is no necessity to implement any such low level functionality.

The game engine is quite descent optimized although it cannot compete with all other options in terms of performance. It provides steady and professional support and a very active community. Also regular updates are provided for all license models. Moreover the Unity3D engine is vastly platform independent. Currently Windows, Mac, Linux, iPhone, common game-consoles and web-browser based applications are supported. Especially the support of mobile devices like the iPhone guaranties that the game engine is sufficiently optimized to target low level hardware. Also there is no need to develop different code for different platforms. With the easy of a few clicks Unity compiles the same code for different platforms as long as the used features are not relying on special hardware.

5.2.3.1 Unity3D - WYSIWYG – Editor

Unity3D combines full automatic content registration and management with WYSIWYG and Drag&Drop editing paradigms. This will speed up the definition of graphical content compared to Orient and FearNot! assembling scenarios, positioning characters and other assets will be done in a fraction of time required with the Orient Framework. Additionally all editor functionality can be extended by so called editor scripts. As ION-components can be linked to Unity3D components it becomes possible to create the abstract world description required by ION alongside with the virtual 3D representation. This ensures consistency among these models. The update of existing content is fully handled by Unity3D which also includes already placed assets.

5.2.3.2 Unity3D – Integration with ION

For the integration of ION with Unity different approaches where explored. The most direct approach would have been to re-implement ION as a set of Mono-Scripts directly within Unity3D. This step showed not to be necessary as unity allows for .Net dlls to be directly used from Unity3D Mono-Scripts. As such the exiting ION-framework implementation can be directly used without any modification. Furthermore updates to ION will only consider the dll and will not lead to changes in the Unity3D connection.

Unity3D’s realisation of the entity-component based paradigm matches very well with the ION-Framework implementation. Each ION class can be mapped (see Table 1) to a corresponding Unity3D feature. This way it becomes possible to link the realizing application in Unity3D with the synchronized world simulation in ION by using simple references (pointers to memory objects). This keeps the the abstract world description in ION separated from the actual audio-visual realization in Unity3D and allows for replacement of the Realizer if better options than Unity3D get available- 17 -in future. Also the direct use of references
to ION Properties instead of local copies for synchronisation critical variables helps to reduce synchronisation problems as the Unity3D script execution order is undefined.

<table>
<thead>
<tr>
<th>ION - Element</th>
<th>UNITY3D - Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>GameObject</td>
</tr>
<tr>
<td>Action</td>
<td>MonoBehaviour</td>
</tr>
<tr>
<td>Property</td>
<td>Script-Local Reference (to ION-Property)</td>
</tr>
</tbody>
</table>

Table 1: Correspondences of ION-Framework elements to Unity3D elements

5.2.4 Unity3D - Virtual Agent Embodiment

Currently Unity3D is used to implement a graphical framework for defining an motivated character’s virtual embodiment. This is done to support improved emotional expressiveness. Unity3D delivers state of the art animation techniques for animated game characters. They are based on skeletal animation (Figure 5), but deliver no native support for higher level techniques like facial expression blend-shapes [6]. The latter may deliver most convincing results for facial expressions but is computationally very expensive and also requires a lot of content production. Especially the first drawback does not match well with the requirement of in school evaluation And also blend shapes still require an animated skeleton for the rest of the body. On the other side traditional skeleton animation approaches are not flexible enough to compete with the expressiveness of facial blend-shapes if they include facial expressions.

![Figure 5: Skeletal animation of polygonal models; Skeleton in neutral Bindpose and the associated character model (left); modifying the skeleton pose deforms the attached character model (right)](image)

For the eCute framework a procedural approach based on a hierarchical decomposed face rig (Figure 6) with partial defined animations will be explored. These partial animations (eyes, mouth, eyebrows, etc.) are then recombined algorithmically during runtime animating the character as a whole (Figure 7). This approach is derived from a technique suggested by Perlin [7] and has
the advantage of being supported by most available hardware – this is not the case for available (community provided) blend-shape implementations for Unity3d.

Figure 6: A characters Face Rig (left); Its decomposition into influential regions: Eyebrows B, Eyes E, Mouth M and Neckbone N (middle); Hierarchy of the partial facial rigs (right); red nodes are dummy bones that allow to re-compose the full facial animation from the partial ones by using Unity3D’s MixingTransform feature

The generated framework still provides only simplified expressions (comic like characters) but it is expected to improve the recognition of emotional states compared to FearNot!. It will support full body animations (walking, sitting, ducking, etc.) combined with facial actions as speaking or crying. Additionally the framework allows blending performed actions with emotional gestures. These can be facial gestures like wrinkled eyebrows as well as full body gestures. One can imagine hanging shoulders and head combined with slower movements as expression of sadness. Another feature that is anticipated to support the recognition of emotions is the change of skin colour in parts of the character (e.g. blushing of the cheeks).

Figure 7: Composing Facial Expressions; The neutral face (left); three individual animations for each region of the face Eyebrows, Eyes and Mouth for emotion anger (middle); Composed facial expression Anger (right)

Natural Speech Feedback with Text2Speech

In FearNot! it has been recognized that especially younger children perceive the characters more naturally if speech acts are supported by voice. As it is not feasible to provide all speech acts by pre-recorded sound samples (required voices, time, infrastructure, necessary post-processing) another solution had to be found. Throughout these projects different speech synthesis libraries have been tested. In eCute-19 we will explore the use of CereProc.
CereProc is a highly developed speech synthesis system based on word dictionaries. As such it provides more natural language than systems that purely rely on synthesis of basic phonemes. Additionally CereProc supports emotional tags. Such tags generate different pronunciation of words based on the simulated emotion. One problem may be the acquisition of new voices or the extension of libraries. Currently only two voices (male and female) are available for the educational version of CereProc. Especially the simulation of children’s voices might be a challenge.

5.2.5 Input Modalities

Just as in eCircus, eCute requires the integration framework to allow for the integration of new and very different input modalities. An input device server will again be used for taking over this task. The server will provide an abstraction layer between user input events and related in-game events. The following paragraphs state some of the already identified input modalities that will be used in eCute and will be supported by the input server component.

5.2.5.1 Free Text Input and Keyword Recognition

For the MIXER scenario the evaluation will be primarily carried out in schools. This requires the use of common and not too unusual input devices. The both most common ones are mouse and keyboard. Relying on other devices might endanger successful evaluation. Traditional games show a great variety of possible interaction concepts as source for inspiration. But for interaction with characters in dialogues most games go for forced choice dialogues. It is expected that the emotional engagement of the children suffers from such methods as the children are not able to freely express themselves – they are limited to the given choices. Free speech recognition is not very reliable especially on low level hardware. Free text input systems with underlying keyword matching have worked out comparable well for FearNot! and will therefore be reused in eCute. Unity3D provides an easy and flexible way to provide a pleasant graphical user interface (GUI) for this.

5.2.5.2 Kinect – Recognizing Body Gestures

Natural full body interaction is assumed to be the most intuitive way to communicate. Before the rise of Microsoft’s Kinnect it was hard to get freely available and reliable algorithms that use image processing to detect user movement or body gestures. More complex setups were necessary to deliver more robust results and mostly involved unusual hardware configurations. The Kinnect interface comes as an all in one solution for a fairly cheap price. Integration with PC software applications is possible through a wide range of open-source libraries. Recently Microsoft released an also freely available library of their own. [For a more detailed description of the Kinnect interface and its abilities please refer to deliverable D4.1 of this project]
In eCute the Kinect will be used in the TRAVEL showcase which will be evaluated with students from different universities. As such the target platform of the application can be chosen quite freely. A Windows based system is currently the most reasonable, as such system are widely available at the universities. Fortunately a lot of implementations of the Kinect interface are compatible to Windows. Also a direct Unity3D integration – supporting the free version of Unity3D - is available with no usage limitations. The Kinect interface currently only provides primitive data through the available interfaces (User IDs, an approximated skeleton, position of the hand, etc.). As of this a gesture detection framework will have to be developed on top of it and will be integrated into the overall framework.

One alternative to the Kinect is the integration of the already tested and improved Wii-Mote control interface used in Orient-Showcase. But as there is no difference in terms of integration- compared to the Kinect - this is an option with no additional challenge.

5.2.6 Story Facilitator

The story facilitator used in FearNot! and Orient has been further developed since. eCute will again integrate the story facilitator as an important element for shaping the structure of the emerging narrative produced by the autonomous characters. This is important to keep the narrative interesting and motivating for the user. If the story facilitator has to be modified is to be determined throughout the eCute project. Currently no modifications are planned. The story facilitator will be integrated as an autonomous character linked to the ION-Simulation modifying the world and interacting with the contained agents.
6. Conclusion

In this document, we describe the eCute Integration Framework. We described requirements for integration frameworks in general and then we compared two existing frameworks namely FearNot Framework and Orient Framework. These were developed in earlier EU projects and tried to establish a link between different components using ION Framework. After looking at these existing frameworks we describe the components identified in the eCute requirement analysis and how we intend on integrating them.
7. References


