Physiological-based Dynamic Difficulty Adaptation in a Theragame for Children with Cerebral Palsy
Acknowledgements

I’d like to address my thanks to both Dr. Richard and Dr. Yamaguchi, for their involvements in my research. I’d also like to address my thanks to Pr. Harada, who let me conduct my research in his laboratory and to Pr. Hardouin and Pr. Boimond, for letting me conduct my research in Harada's laboratory.

I am also thankful to all the students of Harada’s laboratory, for having made my stay in TUS as enjoyable as possible.
This research have been done in the context of the Children Rehabilitation Project (CRP), for LARIS (University of Angers), at Harada’s laboratory (Tokyo University of Science)
Contenu
Acknowledgements .......................................................... 2
Introduction ........................................................................ 2
Surroundings ...................................................................... 2
The research question ....................................................... 3
Refining the research question ........................................... 3
Finding the research question ............................................. 4
Details about the approach ................................................ 7
The material ........................................................................ 7
The flow theory ................................................................. 7
Physiological features ....................................................... 8
Preparation of the affective recognition ......................... 10
Step #1 - Implementation of BITalino in Unity3D .... 10
Step #2 – Extraction of physiological features from BITalino signals ......................................................... 12
Step #3 - Training of the learning classifier ......... 13
Experiment – The affective recognition ................. 16
ProvokeAffect application (v2) .................................. 16
Experiment design .......................................................... 16
Result of our experiment .............................................. 16
Conclusion .......................................................................... 17
CRP – Rehabilitation application difficulty .............. 18
Implementation of the affective recognition in the CRP ................................................................. 18
Simple design ................................................................ 20
Application of the simple design in Rehab-Island .... 21
Concrete DDA ................................................................. 22
Example in use ................................................................ 23
Conclusion .......................................................................... 24
Discussion and futur work ............................................. 24
Annex 1 - CVA ................................................................. i
Explanation ...................................................................... i
Rehabilitation .................................................................. i
Annex 2 - CRP ...................................................................... iii
Annex 3 .............................................................................. iv
DDA .................................................................................. iv
Annex 4 – State of the art #1 ......................................... v
Annex 5 – State of the art #2 ......................................... vi
Annex 6 – Feature extraction ECG ............................... vii
Annex 7 – Feature extraction EDA ............................... viii
Annex 8 – MATLAB Implementation ......................... ix
Annex 9 – SVM in Unity3D ............................................ xiii
Annex 10 – ProvokeAffect V2.0 ................................ xvi
Annex 11 – Feature extraction and emotion recognition ................................................................. xvii
Annex 12 – Class Affective DDA Rehab-Island ............ xix
Introduction

The purpose of this research is to provide a physiological-based Dynamic Difficulty Adaptation (DDA) able to be used in a rehabilitation application.

We present our research by going through (1) our state-of-the-art; (2) our approach; (3) our implementation and (4) our research results. We then discuss our findings.

Surroundings

While not directly related to our research, we present below some of the surroundings of our research. To do so, we go through those acronyms:

- CVA
- CRP
- DDA

More details can be found in annex 1, 2, 3.

About CVA

“A stroke, sometimes referred to as a cerebrovascular accident (CVA) is the loss of brain function due to a disturbance in the blood supply to the brain” [1].

CVA is a leading cause of disability, with above 65% of survivors having minor to severe impairments [2]. Luckily, rehabilitation is possible.

In order to make the rehabilitation more effective, rehabilitation applications (such as the CRP) have been lately heavily researched [3].

See annex 1 for more details.

About CRP

The Children Rehabilitation Project (CRP) is an adaptive modular interface destined to CVA’s rehabilitation. It has for purpose to offer a state-of-the-art rehabilitation platform to improve CVA’s rehabilitation through rehabilitation applications (see Figure 1).

This project is supported by Oseo Innovation and researched by LARIS (University of Angers), Harada’s laboratory (Tokyo University of Science) and the Hospital of Angers.

Table 1. From left to right: Tokyo University of Science; University of Angers; Hospital of Angers.

There is actually only one rehabilitation application in the CRP: Rehab-Island (see Figure 1). In Rehab-Island, the user has to catch flying objects by moving its arms. The arms position is captured by Kinect®.

See annex 2 for more details.
About DDA

"Dynamic Difficulty Adaptation (DDA) is the process of automatically change parameters, scenarios, and behaviors in real-time, based on the user’s ability, in order to avoid them becoming bored (if the game is too easy) or frustrated (if it is too hard)" [4].

One of the main DDA’s advantages regarding non-dynamic difficulty is the possibility to consider user’s progression. See annex 3 for more details.

The research question

In this part we go through:

- The finding of the research question;
- The research question.

Refining the research question

Regarding the current state of CVA’s rehabilitation, it was decided that: The research question had to be defined regarding DDA.

The objectives were to find a DDA able to be:

- Implemented in a rehabilitation application.
- Innovative enough to be presented in a research paper for CVA rehabilitation.

To do so, it was necessary to do a state of the art regarding DDA. More details can be found in annex 4, 5.

Definition

We did a pre-research to have a basic understanding of DDA. We found and used this definition:

"Dynamic Difficulty Adaptation (DDA) is the process of automatically change parameters, scenarios, and behaviors in real-time, based on the user’s ability, in order to avoid them becoming bored (if the game is too easy) or frustrated (if it is too hard)" [4].

Study identification (1)

We covered the period 2000 – present. Relevant publications were identified by searching the following electronic database: Association for Computing Machinery [ACM] (last searched 10 March 2014), CI Nii (last searched 10 March 2014), IEEE Xplore (last searched 10 March 2014), The Institution of Engineering and Technology [IET] (last searched 10 March 2014), ScienceDirect (last searched 10 March 2014).

The databases were searched by indexing terms and free-text terms used with synonyms and related terms in the title or abstract. We searched for “DDA” and “Games” (see Table 2). Studies were included if they meet the following criteria: (1) The publication date was after 2000; (2) the full-text publication was written in English or French.

Study identification (2)

We covered the period 2004 – present. Relevant publications were identified by searching the following electronic database: Association for Computing Machinery [ACM] (last searched 1st April 2014), IEEE Xplore (last searched 1st April 2014), ScienceDirect (last searched 1st April 2014), PubMed (last searched 1st April 2014), Google Scholar [50 most relevant occurrences] (last searched 1st April 2014).

The databases were searched by indexing terms and free-text terms used with synonyms and related terms in the full-text. We searched for “Physiology”, “Game” and “DDA” (see Table 3). Studies were included if they meet the following criteria: (1) The publication date was after 2004; (2) the full-text publication was written in English or French.
Difficulty" OR "Dynamic Adaptation" OR "Dynamic Balancing" OR "Dynamic Game" OR "Adaptive Game" OR "Real-time adaptation" OR "Challengeable game" OR "Player satisfaction")

Table 3. Search strategy in IEEE Xplore

Results (2)
We found 173 studies. After removal of the duplicate and application of the criteria there were 149 studies left.

We then selected the studies which focus on DDA based on the affective state of the player, and found a total of 29 relevant studies (see annex 5). This is on those studies that we based our state of the art.

The important difference between the findings and the selected studies (120) is justified by our search strategy. In a previous attempt, we searched “Physiology” AND “DDA”, but it gave near to zero results. So in this attempt, we decided to add several keywords closely related to DDA. This lead to an important number of non-relevant results.

After analysis of the state of the art, we decided the research question.

Finding the research question

As previously mentioned, the objectives were to find a DDA able to be:

• Implemented in a rehabilitation application.
• Innovative enough to be presented in a research paper for CVA rehabilitation.

By choosing to focus on a state of the art DDA using (1) a learning algorithm (such as ANN, SVM …) and (2) physiological signals, we hoped to fulfill those objectives.

We defined our research question as such: CVA’s rehabilitation heavily research the incidence of user’s motivation on rehabilitation. There are not yet (at the best of our knowledge) rehabilitation application including a physiological-based DDA to improve user’s motivation (and to its incidence on rehabilitation). We believe that physiological signals can be used to detect user’s motivation, and – through a physiological-based DDA – can improve rehabilitation. We want to research the incidence of a physiological-based DDA on rehabilitation.

We present bellow our approach possibilities and decision.

In this section, we decided to describe our choices in a succinct way. Some concepts, materials (BITalino Board©) and key-words are used here, but described in the coming chapter.

Available approaches
There were two possibilities to do a DDA using (1) a learning algorithm (such as ANN, SVM …) and (2) physiological signals:

1. Based on the physiological signals of the user, a learning algorithm identifies the adequate difficulty. **We do not identify the affective state.**
2. Based on the physiological signals of the user, a learning algorithm identifies the affective state of the user. The affective state is then passed to an algorithm which determines the adequate difficulty. **We do identify the affective state.**

Choice among available approaches
After discussion, we decided to follow the possibility #2 (see Table 4), which explicit the affective state. Doing so provided us several advantages over the possibility #1:

• **We can use the flow theory.** The flow theory does not depend of the physiological signals but of the affective state. **By using the flow theory, we can guide the user in flow.**
• **Almost all studies use this approach.** From our state of the art, we found that *almost* all use the affective...
state. By following the same approach, we can base our work on the existing work.

We describe bellow our physiological-based DDA’s approach, as well as our implementation approach.

**Approach description – DDA**

We based our physiological-based DDA’s approach as such:

1. Use **BITalino Board©** to get physiological signals;
2. Extract **physiological features** from the physiological signals. *If necessary, use a third-party library for noise reduction*;
3. Use a learning classifier to classify physiological features in affective state. *A training of the learning classifier is necessary*;
4. From the affective state, based on the flow theory, increase or decrease the difficulty to orient the user toward flow state (see Figure 3).

Figure 2 and Figure 3 may look a little obscure. We advise you to go back to them after having assimilated the concepts described in the coming chapter.

---

**Anxiety**

<table>
<thead>
<tr>
<th>Anxiety</th>
<th>Anxiety</th>
</tr>
</thead>
<tbody>
<tr>
<td>No boredom</td>
<td>Boredom</td>
</tr>
</tbody>
</table>

*Decrease difficulty*

<table>
<thead>
<tr>
<th>No Anxiety</th>
<th>No anxiety</th>
</tr>
</thead>
<tbody>
<tr>
<td>No boredom</td>
<td>Boredom</td>
</tr>
</tbody>
</table>

*Maybe flow*

<table>
<thead>
<tr>
<th>Maybe flow</th>
<th>Increase difficulty</th>
</tr>
</thead>
</table>

Figure 2. The user can be in flow only if the difficulty is above a threshold value. Note that Anxiety and Boredom together are less likely to be flow...

Several different trails were possible regarding DDA’s algorithm. Since it was the 1st study of the laboratory using BITalino, we decided to use a simple algorithm (see Figure 3), while keeping in mind it may need to be improved.

---

By modifying the challenge in such a way (step 4), we have our DDA. We refer to this approach as **AFFECT-DDA**, since it is based on the affective state of the user.
The test of this system – and the validation of our approach - will take place in the Hospital of Angers with Children with Cerebral Palsy.

Approach description – Implementation in CRP
We based our AFFECT-DDA’s implementation in the CRP as such:

1. Identify the variables which influence the difficulty.
2. Define a step and a range for each variable, as well as their impact on the difficulty. Iterations are necessary here;
3. Find a “difficulty equation” (if I want to lower the difficulty, how should I change the variables?).
4. Put the AFFECT-DDA and the “difficulty equation” together.
Details about the approach

In this part, we go through:

- The **material** (software/hardware);
- The **flow theory**;
- The **physiological features**.

This section may seem a little bit messy in its organization. It is due to two facts: (1) there are a lot of – small - different aspects to cover; (2) we can only present one aspect at a time, even if each aspect is “linked” to others aspects.

The material

We present here our available materials (software/hardware).

About our software

We had:

1. **MATLAB**. Our MATLAB’s version was r2014a, and we disposed of several toolboxes, including **Signals Processing v6.21** and **MATLAB Compiler v5.1**.
2. **Unity3D Pro**. Our Unity3D Pro’s version was 4.52f1, without third party assets.

About our hardware

We had:

1. **2 x set of BITalino Board© (and several disposable electrodes)**.

We describe bellow the BITalino Board© (if you feel comfortable enough about the BITalino Board©, please feel free to skip this part).

From now on, will now shorten BITalino Board© by BITalino.

Be aware however, that there are several others types of BITalino boards.

About BITalino

The BITalino (see Figure 5) is a low cost device (149€ [5]) able to read three common physiological signals: **EMG, ECG, EDA**. It is also able to read **ACC and LUX**.

![BITalino Board](image)

**Figure 5 The BITalino Board**

While fairly new (2013 [6]), it already shows promises. We summarize bellow the pros and cons of using it:

**Pros**

- **Cheap**. ...Since others materials commonly used in research (such as Procomp5 [7]), are above the thousands of €.
- **Described in the literature**.
- Available SDK.
- Made to be “hacked”.

**Cons**

- **No features extraction software**. In affective recognition, at the best of our knowledge, physiological signals are only use through their features [8].
- Not yet used in the literature (as of 2014).

Since ECG and EDA can be used to detect Anxiety / Boredom, and since the number of features to extract is only of 14, we decided to use BITalino.

The flow theory

We present here the flow theory and its significance for our research.
The #1 state of the art done on DDA, leaded us to take in consideration the flow theory:

The flow is defined as: The state in which people are so involved in an activity that nothing else seems to matter; the experience itself is so enjoyable that people will do it even at great cost, for the sheer sake of doing it [9]; and has been developed by Mihaly Csikszentmihalyi.

The idea behind the flow theory is as such:

- The user is more susceptible to be in the flow state when the challenge equals the skill.
- If the challenge is less important than the skill, the user is bored (too easy).
- If the challenge is more important than the skill, the user is frustrated (too hard).

In the flow theory (see Figure 8), Anxiety and Boredom are among the affective state which determine in which “direction” the challenge should “move” to meet user’s skills.

Moreover, both of Anxiety and Boredom have been previously used in studies about DDA [10] [11]. We decided therefore to focus on those states.

Physiological features

During our survey on affective DDA, we found that: ECG’s features and EDA’s features are among the most used features (ECG’s features and EDA’s features have been used to identify Boredom [11] and Anxiety [10], which can help to know if the user enjoys the application).

We describe bellow ECG and EDA, and then argue our ECG’s feature and EDA’s features choices (if you feel comfortable enough about ECG and EDA please feel free to skip this part).
About ECG

“ElectroCardioGraphy (ECG) is the recording of the electrical activity of the heart” [12].

Several studies use an important ECG’s feature: Heart-Rate Variability (HRV); in emotion identification [13].

“Heart-Rate Variability (HRV) [or IBI] is the physiological phenomenon of variation in the time interval between heartbeats” (see Figure 9) [14].

Figure 9 HRV, here called RR interval.

About EDA

“Skin conductance, also known as galvanic skin response (GSR) is a method of measuring the electrical conductance of the skin, which varies depending on the amount of sweat-induced moisture on the skin” [15]. While, in its measurement, it is slightly different from EDA (Electrodermal Activity), GSR and EDA are almost synonym.

Several studies use an important EDA’s feature: Skin Conductance Response; in emotion identification [16].

SCR is also a synonym for GSR. However, as several studies before us [11], we refer to GSR’s peaks as SRC (see Figure 10).

Figure 10 GSR. In red the SCR.

Our features choices (1)

It has been shown [17] that the optimal minimum parameters for emotion identification were:

- GSR Count (SCR);
- Mean Heart Rate (or mean RR, or mean HRV);
- Variance Heart Rate (or RR, or NN, or HRV, or IBI).
Those features, in top of being the optimal minimum parameters for emotion identification, have several interests for our research:

- **Used in Anxiety / Boredom identification;**
- **Derived from ECG’s and EDA’s signals (and therefore can be get from BITalino);**
- **Easy to calculate.**

We decided to use those optimal features.

Two studies which focus on Anxiety [10] / Boredom [11] detection, used, in top of those optimal features, several others features. Since some of them where fairly easy to calculate, we decided to integrate them.

In total, we selected 14 features *(the features underlined are the optimal features)*:

- **For GSR**
  - GSR Mean;
  - GSR SD;
  - GSR 1 derivation average;
  - GSR 1 derivation RMS;
  - **GSR SCR;**
  - GSR 1 difference raw data;
  - GSR 1 difference normalized data;
  - GSR 2 difference normalized data.

- **For ECG**
  - **IBI RMSSD;**
  - IBI Pnn50;
  - IBI 1 difference raw data;
  - IBI 1 difference normalized data;
  - IBI 2 difference normalized data;
  - **RR Mean.**

### Preparation of the affective recognition

In this part, we go through the steps #1, #2 and #3 of our AFFECT-DDA approach. As a quick reminder:

1. **Use BITalino Board© to get physiological signals;**
2. **Extract physiological features from the physiological signals. If necessary, use a third-party library for noise reduction;**
3. **Use a learning classifier to classify physiological features in affective state. A training of the learning classifier is necessary.**

While this report must remain short, we understand that some of the steps may lack details. Please, find in annex a more detailed description of our work.

### Step #1 - Implementation of BITalino in Unity3D

Since the BITalino had to be used in Unity3D, we had to implement it in Unity3D. Unfortunately, there was not C# SDK available.

Several methods (see Table 5) have been tested before developing our own solution.

<table>
<thead>
<tr>
<th>Using BITalino Python SDK</th>
<th>We wanted to use BITalino’s Python API (used by the developer of BITalino) within Unity3D through the library IronPython for .NET 2.0. Because of several problems regarding dynamic linking, this method has been abandoned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using BITalino C# API</td>
<td>We wanted to use part of BITalino’s C# API within Unity3D. Because this API is a Windows 8.1 APP, preventing us from using it as a portable class for Unity3D (.NET 2.0), this method has been abandoned.</td>
</tr>
</tbody>
</table>

Table 5 Test to implement BITalino

After these tentative, we decided to develop a C# SDK (see Figure 12) based on the existing Java SDK [18], able to be implemented in Unity3D (.NET 2.0).
We present below a comparison between OpenSignal (BITalino’s software to visualize physiological signals, using Python SDK) and our C# SDK:

We can see that our C# SDK seems as accurate as the BITalino Python SDK (the small differences come from the fact that we compare two different trials on two different scales).

We developed a Window Form to test the C# SDK (see Figure 16), and then developed the Unity3D scripts to implement it under Unity3D (see Figure 17).

In Unity3D, BITalino can be used with only 3 scripts:

- **BITalino Serial Port**, which is nothing more than a container of variables to pass to Manager BITalino.
- **Manager BITalino**, which handle all the logic of the C# SDK.
- **BITalino Reader**, which read BITalino’s frames (in a specific thread). When a frame is read, an event is raised.
Figure 17 The 3 Unity3D Bitalino script. We can see that we have an in-deep control over Bitalino with a relatively small number of scripts.

Figure 18 Unity3D script Bitalino class diagram

We got good results. We present below one of the applications done under Unity3D with Bitalino. It shows the different signals read by Bitalino.

For all the experimentation, we decided to record the signals at 1000Hz (1000 frames / sec).

After the acquisition of physiological signals, we had to extract their features.

**Step #2 – Extraction of physiological features from Bitalino signals**

Even if the features were relatively easy to calculate, we decided, to “test” our algorithms, to use Matlab (it was less error prone than doing it directly in C#).

**Features extraction algorithm**

Due to the large amount of data (1000Hz sample of 30 seconds), and the “almost real-time” process, it was important to have efficient algorithms. In order to do so, we proceeded as such:

1. Read the CVS file (which contains ECG and EDA data). Put ECG and EDA data in specific array;
2. Send the ECG array to a “ECG features extraction” script and the EDA array to a “EDA features extraction script”;
3. In the ECG features extraction (see Figure 20 and annex 6):
   a. Sample data at 255Hz;
   b. Normalize data;
   c. Do Teager Energy Operator;
   d. Detect R peaks from a specific threshold (0.2 in our case);
   e. Extract ECG features.
4. In the EDA features extraction (see Figure 21 and annex 7):
   a. Sample data at 255Hz;
   b. Normalize data;
c. Smooth data with local regression algorithm (loess of Matlab);
d. Do a 255-points Bartlett window;
e. Detect SCR from a specific threshold (0.1 in our case);
f. Extract EDA features

gSR_Mean = 0.5473
gSR_SD = 0.4978

sRCMeant1Difference = 0.0069
sRCMeant1Differencenormalized = 0.0069
sRCMeant2Difference Normalized = 0.0070

sRCMean1DifferenceNormalized = 7.1788e-06
sRCMean2DifferenceNormalized = 7.1788e-06
nbSRC = 11

gSRd1_Mean = -1.6172e-05
gSRd1_RMS = 0.6024

We realized after having writing Matlab scripts, that it was possible, trough DLL and the Matlab Compiler Toolbox, to use them (and therefore Matlab's functions) under Unity3D, even in build mode (see annex 8 for implementation). We decided to keep the Matlab scripts, and used them in Unity3D.

After being able to do features extraction, we had to train a learning algorithm able to recognize the affective state from those features.

Step #3 - Training of the learning classifier

To train the learning classifier, we proceed as such:

a. Put the subject in a given affective state (Anxiety or Boredom);
b. Record physiological signals;
c. Extract features from physiological signals;
d. Label features (Anxiety or Boredom);
e. Train the learning classifier by providing it labeled features.

The step #b can be done through BiTalino. The step #c can be done through our Matlab scripts.

In order to do the step #1 (Put the subject in a given affective state (Anxiety or Boredom)), we developed an application which provoke Boredom / Anxiety: ProvokeAffect.

We decided to develop our own application, instead of using verified “emotion induction” pictures, sounds, or movies because those footages only induct primary emotions. In our case, Boredom and Anxiety are secondary emotions.
ProvokeAffect application (v1)
We developed three scenes in ProvokeAffect: (1) Boredom scene, (2) Anxiety scene, (3) Excitement scene. Each of those scenes follows the same rules:

A message displays the name of a color (Red; Blue; Green). The subject have to click (one time) on the sphere having the named color. After a few seconds (or milliseconds), the color’s name change. If the subject clicks on the sphere before the time limit, he gains 1 point. If he does not click on the sphere before the time limit, or if he misses, he loses 1 point.

The differences between each scene depend of the: (1) Overall difficulty, (2) Environment (sounds, background).

To provoke Boredom, the interval of time between each message changes is set to 4 seconds, without difference between (1) color’s spheres, (2) color’s message (see Figure 22). Moreover, the sphere’s number is 2.

To provoke Anxiety, the interval of time between each message changes is set to 0.85 seconds, with difference between (1) color’s spheres, (2) color’s message. Moreover, the sphere’s number is 3.

In top of that, the Anxiety scene has a DDA to increase difficulty is the user is doing too well. The DDA changes the interval of time between each message (minimum: 0.5s; maximum: 0.85s), (2) changes the shape of the sphere, (3) changes the background color, (4) produces heart-beat sound.

To provoke Excitement, the interval of time between each message changes is set to 1.5 seconds, with difference between (1) color’s spheres, (2) color’s message. Moreover, the sphere’s number is 3.

In top of that, the Excitement scene has a DDA to increase difficulty is the user is doing too well. The DDA changes the interval of time between each message (minimum: 0.75s; maximum: 1.5s).

In top of that, it is possible to do series of combo. Successful combo produces (1) score increase, (2) sounds effects.
Having the ProvokeAffect application, we did a preliminary test. The main objective was to see if there were noticeable differences when looking at the physiological signals.

We present below some of our results:

<table>
<thead>
<tr>
<th>EDA - Boredom</th>
<th>EDA - Anxiety</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Graph 1" /></td>
<td><img src="image2.png" alt="Graph 2" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Graph 3" /></td>
<td><img src="image4.png" alt="Graph 4" /></td>
</tr>
</tbody>
</table>

Table 6. Physiological signals in Boredom and Anxiety scene of ProvokeAffect.

We can see that (see Table 6), even without the features, important differences between EDA - Boredom and EDA - Anxiety. While the ECG data did not provide noticeable differences, it can be explained because of the way we did the preliminary test (only one subject, no breaks between scenes).

There was almost no difference between Anxiety and Excitement. Hence, in the experiment, we decided to only consider Boredom and Anxious affective state. Also, while we think ProvokeAffect is an application able to induce Boredom / Anxiety, we decided, in the experiment, to “test” it by providing an Likert scale test to the subject.

We then, from the labeled features, had to train the learning classifier.

Train the learning classifier
From labeled signals (through ProvokeAffect application), we got labeled features (through MATLAB scripts).

In order to do the step #d (Train the learning classifier by providing it labeled features), we implemented a learning classifier able, in real-time, to classify features in Unity3D.

We decided to use a Support Vector Machine to classify physiological features (which avoid over-fitting, even a large number of parameters). To do so, we used LibSVM (used in several research papers [19]). Luckily enough, a C# wrapper was already developed. We modified it a tiny bit to make it run in .NET 2.0 and successfully implemented it under Unity3D (see annex 9).
Experiment – The affective recognition

After having successfully completed all 3 steps of our affective recognition, an experiment was designed and conducted.

There were several objectives to this experiment.

1. Get EDA and ECG signals from subjects in order to have a significant amount of labeled features to train the SVM.
2. Having correct labeled signals, by having the subject reporting, for each scene, if he felled this emotion.

In order to so, we extended ProvokeAffect application with several features.

ProvokeAffect application (v2)

We kept the three scenes, but included them in an automated process (see annex 10). The process was organized as such:

1. Set-up of the experiment (such as choice of the language among French, English and Japanese);
2. Basic information of the subject;
3. Rules’ explanation;
4. Session on the Boredom / Anxiety / Excitement scene of 50 seconds;
5. “Likert scale” self-report on the session (and break of 10 seconds);
6. Repeat from #4 as many time as necessary;
7. Though on the experiment.

We decided to use an automated process to let the subject as “alone” as possible, in order to not parasite his affective state. Also, it is less error prone.

We also added, in each affect scene, the possibility for the subject to indicate in real-time if he was enjoying the experiment, through mouse scroll. By doing so, we hoped to know if the subject was more likely to be in flow-state at a precise time.

However, due to the intensity of the Anxiety scene, subjects usually “forget” to give their enjoyment. Therefore, we did not use this real time enjoyment’s measurement.

Experiment design

We decided to use 12 subjects and designed the experiment as such:

- Use all three scenes of the application, each scene being displayed 4 times, in a specific order: 4x Boredom, 4x Anxiety, 4x Excitement. While our choice may be arguable, we decided to so to avoid previous trial to “parasite” the current trial.

In total there was (4 times 3 scenes) 12 trials by subject, each trials lasting 50 seconds. When giving the data to MATLAB, we cut the 5 first and last seconds of the experiment.

- After each trial, a self-report is conducted. The self-report consist of a few “Likert-scale” questions (A 5 scale, from Very much to not at all). While the majority of the questions are non-relevant, in order to not orient the subject, there are one question on how much he felled Boredom and one question on how much he felled Anxiety, to know how to label its data.

The application stores two files for each trial (1) the physiological signals, (2) the “scroll event”); and two files for each experiment (1) the subject information, (2) the self-reports’ answers.

Result of our experiment

The experiment went well.

For one subject we had to use a variation of our ECG recording. But even with this variation, we had similar ECG signals than with the usual recording method, and decided therefore to use them.

The SVM gave us a classification rate close to (after grid search) 70%, which is close to the classification rate of previous studies (usually, from our state of the art, it is around 75-80%). We believe that the classification may improve with (1) more subjects, (2) more features (like the Legendre moments of the different signals [11]).
Conclusion

We know conclude the part #1 of our research. We managed to record physiological signals, to extract features from physiological signals, to classify them in affective state through a SVM, to induct affective state, and to gather enough data to have an acceptable classification rate.

In the next part, we link what we done so far the CRP.
CRP – Rehabilitation application difficulty

In this part, we link our affective recognition to an affective DDA able to be implemented in the CRP. As a reminder, here as steps we have to go through:

1. From the affective state, based on the flow theory, increase or decrease the difficulty to orient the user toward flow state.
2. Identify the variables which change the difficulty.
3. Define a step and a range for each variable, as well as their impact on the difficulty. Iterations are necessary here;
4. Find a “difficulty equation” (if I want to lower the difficulty, how should I change the variables?).
5. “Put the AFFECT DDA” and the “difficulty equation” together.

Prior to those steps, we present how we implemented the affective recognition in the CRP.

Implementation of the affective recognition in the CRP

In this part, we go through:

- Features extraction (and results);
- Emotion recognition (and results).
- Concrete DDA (and example).

Features extraction

The implementation was relatively easy. We simply “linked” our previous finding together.

When BITalino read a frame, an event BITalinoEventArg is fired. This event is handled by DataTreatment, from which AffectiveTreatment is derived. When AffectiveTreatment, which “extract” ECG / EDA features, extract the features, an event AffectiveTreatmentEventArg is fired. This event is handled by AffectiveRecognition, which “recognize” the affective state. More details can be found in annex 11.

Prior to those steps, we present how we implemented the affective recognition in the CRP.

Implementation of the affective recognition in the CRP

In this part, we go through:

- Features extraction (and results);
- Emotion recognition (and results).
- Concrete DDA (and example).

Features extraction

The implementation was relatively easy. We simply “linked” our previous finding together.

Figure 25; Class diagram of our features extraction. See annex 11 for a bigger view.

While there are some small differences for the HRV between Unity3D and MATLAB, we can see that we have almost the same exact same values.

Also, the execution of MATLAB ECG / EDA algorithms is relatively fast (about 1 second for 40 seconds of samples). Moreover, since the calculation happens on a different thread, the FPS is not slow down.
The features extraction is handled (from user’s perspective), by 4 scripts. Given the “in-deep” control the user has, this is relatively low.

Emotion recognition
The implementation was relatively easy. We simply “linked” our previous finding together.

Figure 27; Class diagram of our emotion recognition. See annex 11 for a bigger view.

AffectiveRecognition makes the junction between AffectiveTreatment and the SVMMManager (see Figure 28 and Figure 29).

Sample of results
Here are the results we got under Unity3D for about 8 minutes of test, with 40 seconds of sample between each call to the SVM:

Figure 28 "Affective" SVM under Unity3D

We can see that the SVM is correctly initialized. We made it do a grid search on the parameters, to find the best parameters possible, explaining the “strange” value of the parameters. The SVM trained has here a cross validation rate of about 67%. It is then call every 40 seconds, and successfully classify the data got from the BiTalino through AffectiveTreatment.

Unity3D’s profiler is not affected by the calculation, meaning that the application will be able to run smoothly even with the calculation (see Figure 31).
Figure 31; Unity3D’s profiler. The pics are non-related to our calculation. From the profiler, our calculations are negligible.

In our configuration, since it is necessary to recognize Anxiety AND Boredom, we decided to use 2 SVM. One trained to recognize Anxiety and the other trained to recognize Boredom (see Figure 32).

Figure 32; The Affective Recognition. We can see that there are 2 SVM: 1 for Anxiety, 1 for Boredom.

Figure 33; The ManagerA. The ManagerB, while focusing on Boredom, is similar.

In total, to be able to “successfully” recognize the affective state in the CRP, we need 12 scripts:

- 3 for BITalino
- 1 for BITalino’s physiological signals treatment (feature).
- 1 for the Affective recognition, which link BITalino physiological signals (through its features) to the SVM.
- 3 for the SVM which recognize Anxiety
- 3 for the SVM which recognize Boredom

While perfectible, we think our design allow the user to easily configure the emotion recognition.

In the CRP, we place those scripts into several GameObjects (see Figure 34). Note that those GameObjects can easily be placed in 1 prefab / package, to be later reused.

Figure 34. BITalino handle BITalino’s script. SVMManagerA (and B) handle SVM’s scripts. AffectiveRecognition handle AffectiveTreatment and AffectiveRecognition.

We still need however, to link our affective recognition to our DDA.

Since there are (at the best of our knowledge\(^1\)) no methods to determine the difficulty of an application, we decided to come up with our own method.

There is actually only one rehabilitation application in the CRP: Rehab-Island. But the CRP is supposed to be able to handle several rehabilitation applications.

Simple design

We used a simple design to modify the difficulty.

- The AFFECT-DDA only modifies the overall difficulty, which is defined for each rehabilitation application.
- The difficulty’s range goes from 0% to 100%. 0% corresponds to the lowest difficulty, 100% to the highest difficulty.

As a reminder, here is our AFFECT DDA design:

\(^1\) Since this problematic is not directly related to our research, we did not do an in-deep state of the art.
The difficulty is defined inside the application, with – in the case of Rehab-Island - the following steps:

1. For each parameter, a minimal and maximal value is defined (example: for the speed of object, from 1 to 4).
2. An equation is defined for each parameter. This equation takes in input the difficulty’s value and gives in output the parameter’s value.

We then defined an equation for the parameters. In our case, we decided to go straightforward. The parameter(s) with a strong influence follow a LOG equation, the parameter(s) with a medium influence follow a LINEAR equation, and the parameter(s) with a low influence follow an EXP equation.

Below are the parameter’ curves:

Note that those steps are an example. Depending of the application, the difficulty’s definition may vary. The main advantage of is that the difficulty is “independent” from the parameters.
According to our design, by changing the difficulty the parameters are correctly adapted.

It is important to know if the parameters equation corresponds to the difficulty that may feel the user. To do so, we would like to use a Likert-scale self-report, which should be tested with CVA’s survivors.

A last thing to think about is the way we increase / decrease difficulty. As a reminder, here is our AFFECT-DDA’s approach:

Since the difficulty is continuous, we decide to use a simple LINEAR equation. Also, since the sample of physiological signals is of 30-40 seconds, we decide to use a low-medium adjustment: Every 30-40 seconds, the difficulty is adjusted of 10%.

It is important to know if this step is unnoticed by the user. To do so, we tested on our own. The increase was slow enough to be unnoticed. A change of 20% in 30-40 seconds was notified, as well as a change of 10% in 10-20 seconds.

Concrete DDA

As a reminder, in our design the DDA changes the difficulty from 0% to 100% (very easy to very hard). The rehabilitation application then “applies” the difficulty. It allows the developer to freely implement its own difficulty.

In our case, since Rehab-Island was already developed, it allows us to freely implement our DDA in the current implementation.

We already discussed our DDA in Rehab-Island. We describe below our implementation.

Rehab-Island is an application still in development. It has been developed by several developers (mostly trainee)
without a definite “plan” of development. It (of course) resulted as:

- No – easy - possibility to maintain it;
- No – easy - possibility to make it evolve;
- A lot of dead-code;
- A lack of documentation (of any sort).

Therefore, we decided to not follow a defined “plan” of development and to “add to the mess”. We created a class dedicated to Rehab-Island (see Figure 41), and let it do all the work. See annex 12 for more details.

Figure 41; Class diagram of our affect DDA.

There is only one script for Rehab-Island (see Figure 42). Note that the lack of public parameter is voluntary.

Figure 42. Note that AffectiveDDA is not designed for Rehab-Island and can be reused. It is not part of the "dirty-code". Affective DDARehabIsland, on the other hand, it designed for Rehab-Island only.

Example in use

We show bellow some screen-shot of the application Rehab-Island in use with our Affect DDA. For the demonstration, we forced the Affect DDA to detect Anxiety and no Boredom.

The Affective DDA get the user’s affect every 40 seconds and was configured to change difficulty of 10% over 30 seconds. We let the application run for 2 minutes and got the expected results (diminution of difficulty, change of difficulty’s parameter value).

Figure 43; In top, the application after about 30s, below, the application after about 1m30s. The velocity decreases while the distance and size increases.

---

2 It was originally design to follow a strict MVC pattern (which was to being with a debatable choice to do under Unity3D). The overall difficulty probably prevented others developers to follow it.

23
Figure 44: The difficulty changes every frame.

**Conclusion**

We saw how we choose to implement our AFFECT-DDA into Rehab-Island. We successfully managed to links all our steps in the application, and therefore to do our approach.

Everything, from the user perspective is held in only 12 scripts, and the design has been done in such a way that it is easily maintainable and extendable (expect for the Affect DDA Rehab-Island script).

It was the first time for this laboratory to develop a system based on physiological signals. The solution developed in this research is now used in several others projects of this laboratory, and therefore improved.

Even if the most difficult have been done, there are still a few things which need to be done or improve.

- Concerning BITalino, there are a lot of noises in the signals. Even if we managed, through MATLAB, to “erase” most of it, there is still room for improvement.

- Concerning the features extractions, even if we have a reasonable set of features, we believe that others features should be extracted (such as Legendre-Moment [11]).

Also, while the experiment we did (at every level) only concerned healthy students (22-24 years old), we still need to test the system with children, and more especially with children currently in CVA’s rehabilitation. This is, we think, the most important thing to do in the future research.

Another important thing to do in the future research is to find a better device to capture physiological signals. We believe that BITalino may not be the most adapted solution in a rehabilitation system (a smartphone able to detect heart-beat may be better). *We used BITalino since it helped us to better solve most of the problematics link to the affective recognition.*

**Discussion and future works**

Unfortunately the validation of our approach can only be done in the Hospital of Angers, with Children with Cerebral Palsy. We still believe that our AFFECT-DDA (*because it orients the user in the flow state*) will improve user’s motivation and therefore its rehabilitation be we need to test it, and to validate it.


    Available:
    http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
    [Accessed 08 2014].

[Online]. Available:
    [Accessed 08 2014].

    Available:

   [Accessed 08 2014].
Annex 1 - CVA

Explanation

“A stroke, sometimes referred to as a cerebrovascular accident (CVA) is the loss of brain function due to a disturbance in the blood supply to the brain” [1].

While there are several types of CVA (see Figure 45), all affect area of the brain.

![Types of Stroke](image)

**Figure 45.** The two types of stroke.

We present below some of the Effects and Statistics related to the CVA, in order to better understand the CVA’s rehabilitation problematic.

### About CVA - Statistics

According to the World Health Organization, **15 million people suffer stroke worldwide each year.** Of these, 5 million die and another 5 million are permanently disabled.

**Stroke is a leading cause of disability,** and at the same time a leading preventable cause of disability. **25% of survivors recover with minor impairments,** while **40% of survivors experience moderate to severe impairments requiring special care.**

Stroke costs the United States an estimated $36.500.000.000 each year. This total includes the cost of healthcare services, medications to treat stroke, and missed days of work. **The average cost of care for a patient up to 90 days after stroke is $15.000.**

*For more information, please refers to the following source: [2].*

### About CVA – Effects

*While this list does not pretend to be exhaustive, it presents some the main CVA effects (others than death...). In bold the effects targeted by the CRP.*

<table>
<thead>
<tr>
<th>Effect</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hemiparesis</td>
<td>Weakness on one side of the body;</td>
</tr>
<tr>
<td>Hemiplegia</td>
<td>Paralysis on one side of the body;</td>
</tr>
<tr>
<td>One-sided neglect</td>
<td>Ignoring the weaker side;</td>
</tr>
<tr>
<td>Dysphagia</td>
<td>Trouble swallowing.</td>
</tr>
<tr>
<td>Cognitive changes</td>
<td>Changes in mood; Behavior or personality changes.</td>
</tr>
<tr>
<td>Aphasia</td>
<td>Difficulty with speech and language;</td>
</tr>
<tr>
<td>Vision problems</td>
<td></td>
</tr>
</tbody>
</table>

*For more information, please refers to the following source: [2].*

Like mentioned above, CVA is a leading cause of disability, with above 65% of survivors having minor to severe impairments [2]. Luckily, rehabilitation is possible.

We present bellow the CVA rehabilitation, in order to better understand our research purpose.

### Rehabilitation

“In the United States more than 700,000 people suffer a stroke each year, and approximately 2/3 of these individuals survive and require rehabilitation. The goals of rehabilitation are to help survivors become as independent as possible and to attain the best possible quality of life. Even though rehabilitation does not "cure" the effects of stroke in that it does not reverse brain damage, rehabilitation can substantially help people achieve the best possible long-term outcome” [20].

The purpose of the rehabilitation, as mentioned above, is not to cure (this is hardly possible) but to give the possibility to avoid complication and acquire independency again.

If the rehabilitation is possible, this is mainly thanks to the cerebral plasticity.

### About CVA - Cerebral plasticity

“The cerebral plasticity refers to changes in neural pathways and synapses which are due to changes in behavior, environment and neural processes, as well as changes resulting from bodily injury” [21].
As shown above (see Figure 46), the cerebral plasticity allow another area of the brain to execute the function which was taking place in the damaged area.

In order to make the rehabilitation more effective, several paths have been explored. The theragames (or rehabilitation applications) is one of them.

We present bellow the theragames, and their advantages on traditional rehabilitation methods.

### About CVA – Theragames

The theragames are part of the Serious Game, which are, since a few years already, widely use in health-care, psychology, ...

![Figure 47 Number of serious games used in health-care.](image)

Among their advantages upon traditional rehabilitation method, there are mainly:

- **Access to quantified data for the doctor and for the patient.**
- **Augmentation of the motivation.**
- Possibility to repeat and to adapt the exercises.
- **Possibility to scale the exercises (via Internet).**

Rehabilitation applications have been lately heavily researched [3], mainly since the motivation is one the main factor in rehabilitation. Several of those theragames include Virtual Reality, in order to propose “every-day life interaction”, immersion (increase of motivation), security ...³

Our research is based on one of those rehabilitation applications: The Children Rehabilitation Project.

---

³ The interfaces and the immersion of the Virtual Reality have definitely their place in rehabilitation application. As such, numerous VR laboratories decided to focus on rehabilitation.
Appendix 2 - CRP

Explanation

The **Children Rehabilitation Project (CRP)** is an adaptive modular interface destined to CVA’s rehabilitation. It has for purpose to offer a state-of-the-art rehabilitation platform to improve CVA’s rehabilitation through rehabilitation applications.

More than adults, children are most likely to lost motivation in front of a repetitive exercise, and therefore to stop doing it. But since the repetition is necessary in the rehabilitation process, it is important to keep them interested.

In order to keep them interested, with the rapidity of development in rehabilitation application, it is now possible to design a highly number of different rehabilitation applications. Nonetheless, the integration time in the existing rehabilitation solution is still high.

In order to diminish the integration time, some projects provide a framework which handles the different rehabilitation application.

The CRP has been developed as such a framework. In top of being able to implement several state of the art rehabilitation applications, it can:

- Automatically adapt the applications (DDA);
- Provide data visualization;
- Provide telerehabilitation (*the possibility to do rehabilitation at home, through an internet connexion*).

It is important to notice that the CRP, which currently focus on children, may be easily adapted to fit adult need.
Annex 3

DDA

“Dynamic Difficulty Adaptation (DDA) is the process of automatically change parameters, scenarios, and behaviors in real-time, based on the user’s ability, in order to avoid them becoming bored (if the game is too easy) or frustrated (if it is too hard)” [4].

To summarize, the DDA is supposed to automatically change the difficulty as the application is too easy / too hard.

There are several reasons of why a “simple selection of level of difficulty (easy-medium-hard)”, at the beginning of the application, may not be enough:

- There is a limited difficulty variation;
- There is a difficulty gaps between levels;
- It is not responsive to player learning;
- It is time-consuming to implements those variation;
- The user has to "guess" his level regarding those levels of difficulty.

![Image](https://example.com/image1)

Figure 48. A partial solutions may to increase the difficulty number, like show in the figure bellow, but it may bring others problems, such as the difficulty to guess his level.

The DDA’s main advantage regarding non-dynamic difficulty adaptation is the possibility to automatically adapt itself from user learning.

Most of the DDA are based on a change of “basic” variables triggered after some events. In such implementation, these changes may produce rubber-band effects which:

- Are “unfair” for the others.
- Produces a sudden change in the difficulty which breaks the flow.

![Image](https://example.com/image2)

Figure 49. One of the most famous example of a "bad DDA" is in Mario kart, where the last user always get the best items.
### Annex 4 – State of the art #1

<table>
<thead>
<tr>
<th>Challenge Title</th>
<th>Authors</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenge-sensitive action selection: an application to game balancing</td>
<td>Andrade, G.</td>
<td>2005</td>
</tr>
<tr>
<td>Dynamic Difficulty Controlling Game System</td>
<td>Sang-Won Um</td>
<td>2007</td>
</tr>
<tr>
<td>Adaptive Computer Game System Using Artificial Neural Networks</td>
<td>Kok Wai Wong</td>
<td>2007</td>
</tr>
<tr>
<td>Dynamic difficulty adjustment of game AI for video game Dead-End</td>
<td>Xinrui Yu</td>
<td>2010</td>
</tr>
<tr>
<td>Creating appropriate challenge level game opponent by the use of dynamic difficulty adjustment</td>
<td>Lingdao Sha</td>
<td>2010</td>
</tr>
<tr>
<td>Dynamic Difficulty Adjustment of Game AI by MCTS for the game Pac-Man</td>
<td>Ya’nan Hao</td>
<td>2010</td>
</tr>
<tr>
<td>Dynamic difficulty adjustment realization based on adaptive neuro-controlled game opponent</td>
<td>Wan Huang</td>
<td>2010</td>
</tr>
<tr>
<td>Adaptive agent generation using machine learning for dynamic difficulty adjustment</td>
<td>Arulraj, J.J.P.</td>
<td>2010</td>
</tr>
<tr>
<td>To create DDA by the approach of ANN from UCT-created data</td>
<td>Xinyu Li</td>
<td>2010</td>
</tr>
<tr>
<td>Optimizing player’s satisfaction through DDA of game AI by UCT for the Game Dead-End</td>
<td>Yidan Zhang</td>
<td>2010</td>
</tr>
<tr>
<td>Guiding User Adaptation in Serious Games</td>
<td>Joost Westra</td>
<td>2010</td>
</tr>
<tr>
<td>Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI</td>
<td>Chin Hiong Tan</td>
<td>2011</td>
</tr>
<tr>
<td>Motivation based difficulty adaptation for therapeutic games</td>
<td>Hocine, N.</td>
<td>2011</td>
</tr>
<tr>
<td>Dynamic difficulty adjustment based on an improved algorithm of UCT for the Pac-Man Game</td>
<td>Bin Wu</td>
<td>2011</td>
</tr>
<tr>
<td>Neuroscience and simulation interface for adaptive assessment in serious games</td>
<td>Thomas D. Parsons</td>
<td>2011</td>
</tr>
<tr>
<td>AntBot: Ant Colonies for Video Games</td>
<td>Recio, G.</td>
<td>2012</td>
</tr>
<tr>
<td>Software design patterns for enabling auto dynamic difficulty in video games</td>
<td>Chowdhury, M.I.</td>
<td>2012</td>
</tr>
<tr>
<td>Digital-pheromone based difficulty adaptation in post-stroke therapeutic games</td>
<td>Abdelkader Gouaiach</td>
<td>2012</td>
</tr>
<tr>
<td>A framework of Dynamic Difficulty Adjustment in competitive multiplayer video games</td>
<td>Baldwin, A.</td>
<td>2013</td>
</tr>
<tr>
<td>Bringing auto dynamic difficulty to commercial games: A reusable design pattern based approach</td>
<td>Chowdhury, M.I.</td>
<td>2013</td>
</tr>
<tr>
<td>A Control-Theoretic Approach to Adaptive Physiological Games</td>
<td>Parnandi, A.</td>
<td>2013</td>
</tr>
</tbody>
</table>

For more information about those studies, please refer to the associate Prezi and associate Excel file. You can find these resources here: lisabiblio.univ-angers.fr
<table>
<thead>
<tr>
<th>Documents title</th>
<th>Authors</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psychophysiological inference and physiological computer games</td>
<td>Stephen H Fairclough</td>
<td>2007</td>
</tr>
<tr>
<td>Dynamic game balancing by recognizing affect</td>
<td>Tim J. W. Tijs</td>
<td>2008</td>
</tr>
<tr>
<td>Creating an emotionally adaptive game</td>
<td>Tim Tijs</td>
<td>2009</td>
</tr>
<tr>
<td>Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback</td>
<td>Changchun Liu</td>
<td>2009</td>
</tr>
<tr>
<td>Emotionally Adapted Games—An Example of a First Person Shooter</td>
<td>Timo Saari</td>
<td>2009</td>
</tr>
<tr>
<td>Improving the gaming experience using real-time adaptive game difficulty adjustments by utilizing physiological feedback</td>
<td>Chang Hoon Yun</td>
<td>2009</td>
</tr>
<tr>
<td>Bio-Affective Computer Interface for Game Interaction</td>
<td>Jorge Arroyo-Palacios</td>
<td>2010</td>
</tr>
<tr>
<td>Characterizing player’s experience from physiological signals using fuzzy decision trees</td>
<td>Levillain, F.</td>
<td>2010</td>
</tr>
<tr>
<td>Boredom, engagement and anxiety as indicators for adaptation to difficulty in games</td>
<td>Kai Kulkkanнием</td>
<td>2010</td>
</tr>
<tr>
<td>Entertainment modeling through physiology in physical play</td>
<td>Tognetti, S.</td>
<td>2010</td>
</tr>
<tr>
<td>Modeling enjoyment preference from physiological responses in a car racing game</td>
<td>Tognetti, S.</td>
<td>2010</td>
</tr>
<tr>
<td>The influence of implicit and explicit biofeedback in first-person shooter games</td>
<td>Kai Kulkkanнием</td>
<td>2010</td>
</tr>
<tr>
<td>An adaptive game-based exercising framework</td>
<td>Silva, J.M.</td>
<td>2011</td>
</tr>
<tr>
<td>Automatic Recognition of Boredom in Video Games Using Novel Biosignal Moment-Based Features</td>
<td>Giakoumis, D.</td>
<td>2011</td>
</tr>
<tr>
<td>Emotion assessment from physiological signals for adaptation of game difficulty</td>
<td>Chanel, G.</td>
<td>2011</td>
</tr>
<tr>
<td>Neuroscience and simulation interface for adaptive assessment in serious games</td>
<td>Thomas D. Parsons</td>
<td>2011</td>
</tr>
<tr>
<td>On the applicability of heart rate for affective gaming</td>
<td>Mohamed Luay</td>
<td>2011</td>
</tr>
<tr>
<td>Exploring the use of physiology in adaptive game design</td>
<td>Shaomei Wu</td>
<td>2011</td>
</tr>
<tr>
<td>GameEMO: how physiological signals show your emotions and enhance your game experience</td>
<td>Chanel Guillaume</td>
<td>2012</td>
</tr>
<tr>
<td>Zone of impulse: physiological data enhanced gaming</td>
<td>Katharina Reitz</td>
<td>2012</td>
</tr>
<tr>
<td>Co-Adaptive and Affective Human-Machine Interface for Improving Training Performances of Virtual Myoelectric Forearm Prosthesis</td>
<td>Rezazadeh, I.M.</td>
<td>2012</td>
</tr>
<tr>
<td>A novel virtual reality driving environment for autism intervention</td>
<td>Dayi Bian</td>
<td>2013</td>
</tr>
<tr>
<td>Dynamic Difficulty Adjustment by Facial Expression</td>
<td>Nan Xiang</td>
<td>2013</td>
</tr>
<tr>
<td>EEG-based Emotion Recognition for Game Difficulty Control</td>
<td>Sang-Yong Park</td>
<td>2013</td>
</tr>
<tr>
<td>A Control-Theoretic Approach to Adaptive Physiological Games</td>
<td>Avinash Parnandi</td>
<td>2013</td>
</tr>
<tr>
<td>Chill-Out: Relaxation Training through Respiratory Biofeedback in a Mobile Casual Game</td>
<td>Avinash Parnandi</td>
<td>2014</td>
</tr>
</tbody>
</table>

For more information about those studies, please refer to the associate Prezi and associate Excel file. You can find these recourses here: lisabiblio.univ-angers.fr
Annex 6 – Feature extraction ECG

Some developed functions used in the script below are not detailed here.

function [ hRV_Mean, hRV_Variance, hRV_RMSSD, hRV_PNN50, sRCMean1Difference, sRCMean2DifferenceNormalized, sRCMean1DifferenceNormalized ] = DoGetDataECG( eCGRaw, timeRaw, sampleSeconde, threshold )

% Refers to:

% eCGRaw and timeRaw should be of the same size.
% timeRaw should be "small". 10-50 seconds for example.
% sampleSeconde = 255; threshold = 0.2 are default values.

if nargin < 4, threshold = 0.2;end
if nargin < 3, sampleSeconde = 255;end

%% Basic calcs
[ eCGSampled, timeSampled ] = DoSample( eCGRaw, timeRaw, sampleSeconde );
eCGSampledNormalized = DoNormalize( eCGSampled );
eCGSampledNormalized = smooth(eCGSampledNormalized,5,'rloess');

%% Detect R-peaks
tEO = TEO( eCGSampledNormalized );
timeSampledTEO = timeSampled( 2 : end - 1 );

%% Calculations

nbRR = 0;
triggerRR = 1;
maxVal = 0;
maxIdxVal = 1;
idxTime = 0;
rRSignal = zeros( length( tEO ), 1 );

for i = 1:length(tEO)
    if tEO(i) > threshold && triggerRR == 1
        idxTime = i;
        nbRR = nbRR + 1;
        triggerRR = 0;
        maxVal = tEO(i);
        maxIdxVal = i;
    end

    if tEO(i) < threshold && triggerRR == 0 && (timeSampledTEO(i) - timeSampledTEO(idxTime)) > 0.11 && after pick
        rRSignal(maxIdxVal) = 10;
        triggerRR = 1;
        maxVal = 0;
    end
end

%% CALC HRV

hRV = []; hRV50 = 0; lastTime = 0;
for i = 1:length(rRSignal)
    if rRSignal(i) > 0
        hRV(end+1) = timeSampledTEO(i) - lastTime;
        lastTime = timeSampledTEO(i);
    end
end

hrVDataNormalized = DoNormalize(hRV);

%% Results

hRV_Mean = mean( hRV ); % OK, see [3]
hRV_Variance = var( hRV ); % OK, see [3]
hRV_RMSSD = DoRms( hRV ); % OK, see [2]
hRV_PNN50 = hRV50 / nbRR; % OK, see [2]

% Mean difference
sRCMean1Difference = DoMeanDifference( hRV, 1 ); % OK, see [2]
sRCMean1DifferenceNormalized = DoMeanDifference( hRVDataNormalized, 1 ); % OK, see [2]
sRCMean2DifferenceNormalized = DoMeanDifference( hRVDataNormalized, 2 ); % OK, see [2]

end
Annex 7 – Feature extraction EDA

Some developed functions used in the script below are not detailed here.

```matlab
function [ gSR_Mean, gSR_SD, gSRd1_Mean, gSRd1_RMS, nbSRC, sRCMean1Difference, sRCMean1DifferenceNormalized, sRCMean2DifferenceNormalized ] = DoGetDataEDA( eDARaw, timeRaw, sampleSeconde, thereshold )

%% Refers to:

% eDARaw and timeRaw should be of the same size.
% timeRaw should be "small". 10-50 seconds for example.
% sampleSeconde = 255; limitSRC = 0.1 are default values.
if nargin < 4, thereshold = 0.1; end
if nargin < 3, sampleSeconde = 255;end

%% Calcs
[ eDASampled, timeSampled ] = DoSample( eDARaw, timeRaw, sampleSeconde );
eDASampledNormalized = DoNormalize( eDASampled );
eDASampledNormalized = smooth(eDASampledNormalized,9,'rloess');
[ convolution, eDASampledNormalized, timeSampled ] = DoConvolutionBartlett( eDASampledNormalized, timeSampled, sampleSeconde );

%% Results
gSR_Mean = mean( eDASampledNormalized ); % OK, see [2]
gSR_SD = sqrt( mean ( eDASampledNormalized - gSR_Mean^2 ) ); % OK, see [2]
dEDA = diff ( eDASampledNormalized );
gSRd1_Mean = mean ( dEDA ); % OK, see [2]
gSRd1_RMS = rms( dEDA ); % OK, see [2]

[ nbSRC ] = DoCountSRC ( convolution, timeSampled, thereshold );% OK, see [2]
sRCMean1Difference = DoMeanDifference( eDASampled, 1 );% OK, see [2]
sRCMean1DifferenceNormalized = DoMeanDifference( eDASampledNormalized, 1 );% OK, see [2]
sRCMean2DifferenceNormalized = DoMeanDifference( eDASampledNormalized, 2 );% OK, see [2]
end
```
Annex 8 – MATLAB Implementation

This annex presents a how-to integrate MATLAB in Unity3D. We use here a C-shared library.

There are several possibilities to implement MATLAB in Unity3D.

- COM;
- DCE;
- C-shared library;
- MATLAB Builder NE.

In all case, it does not require a MATLAB license/software from the user.

According to several sources, C-shared library/MATLAB Builder NE give good performance, while COM / DCE give bad performance.

- C-shared library is not a solution recommended by MATLAB, and MATLAB do not officially test / support it. It requires MATLAB Application Compiler Toolbox.
- MATLAB Builder NE is an impressive solution which “imports” MATLAB into .NET. It is the solution recommended by MATLAB. It requires MATLAB Application Compiler Toolbox AND MATLAB Builder NE (http://www.mathworks.com/products/netbuilder/).

We decided to use a C-shared library, since we do not have MATLAB Builder NE.

Requirements

- MATLAB 32-bit. Unity3D Editor only accepts 32-bit DLL. Note that it should be possible to use 64-bit DLL in the build application. Note also that Unity3D Editor V5 should accept 64-bit DLL.
- MATLAB Application Compiler Toolbox (32-bit).
- Unity3D Pro is not a requirement. Since the C-shared library is an unmanaged library, it cannot be a plugin. Therefore, Unity3D Pro is not required.

MATLAB set-up

Prior to use a MATLAB DLL in Unity3D, we need to generate the DLL from MATLAB. We detail here how to do.

Functions

It is necessary to create the MATLAB function(s) you want to use. For our example, we will use three different MATLAB functions (see annex 1).

```matlab
function [ y ] = testBasicCalc(x)
function [ x1, y1, x2, y2 ] = testAdvancedCalc( fs, lengthSignal )
function testDisplayFigure( x, y )
```

We advise you to be sure that the functions run correctly into MATLAB before going further. In our case, all the function runs correctly.

If we run the following MATLAB script:

```matlab
tic;
[ r1 ] = testBasicCalc( [ 1, 45, 90, 0 ] )
TimeSpent = toc;
TimeSpent

tic;
[ fs1, y1, fs2, y2 ] = testAdvancedCalc( 1000, 1000 );
TimeSpent = toc;
TimeSpent

tic;
testDisplayFigure( fs2, y2 )
TimeSpent = toc;
TimeSpent
```

We get the following results:
Important note

*Not all functions can work from C-shared library.* While a lot are available, some of them are not. It is especially true if you use MATLAB GUI, or some specific Toolbox. Here is an exhaustive list of the functions you can and cannot use using MATLAB Application Compiler:

http://www.mathworks.com/products/compiler/support/c

C-shared library

Once we ensured that MATLAB function(s) run properly, we generate the DLL (here, we will generate a C-shared library), using the Application Compiler of MATLAB.

- Be sure to have selected Library Project (New > Library Project) [see figure 1].
- Select C Shared Library
- Add the MATLAB functions you want to export in the “Add Exported Functions” box.

MATLAB Compiler generates several folders in top of the DLL.

For redistribution files only” contains the DLL, the C Header, the lib and a read-me. *If we look at the C Header, we can see the MATLAB functions we are interested in, as well as other generated functions.*

Note that each MATLAB function has two different wrappers, one having for suffix mlf, and the other one having for suffix mlx. Both functions give the exact same output. *In our example, we use the ones having for suffix mlf.*
Note also the function `testMatlabIntegrationInitialize(void)` and `testMatlabIntegrationTerminate(void)`.

**Integration with Unity3D**

We advise you to put the DLL at the root of your Unity project.

*Note that the DLL have to be compiled in 32b in order to be used in Unity3D Editor. The Unity3D Application can run 64b DLL, but not the Unity3D Editor (Unity3D 5 may allow 64b DLL in the Unity3D Editor).*

Once the DLL has been copied to the root of the Unity project:

- Create two wrapper classes. One wrapper class will be used to call some important methods of the mclmcrrt** DLL (in my case, the mclmcrrt8_3.dll). *Note that you have to have a 32b version of mclmcrrt** DLL in one your path environment. Note also that the name of the method may differ from the documentation (for example, `mxGetPr_proxy` instead of `mxGetPr`). Use Dependency Walker (see figure 53) to know what to call.

---

![Methods of mclmcrrt8_3.dll retrieved using Dependency Walker.](image)

---

- Create the Unity class that you want to use. As example, we created a TestMatlabIntegration class. When you want to use functions of a given DLL, you have to call NameOfDLLInitialize() (for example, `testMatlabIntegrationInitialize()`). When finished to use functions of a given DLL, you have to call NameOfDLLTerminate() (for example, `testMatlabIntegrationTerminate()`).

- Be especially careful about the data management. In C#, do not use pointer, since this is not managed type. Instead, use IntPtr, which is a managed pointer-type. **Also, use only double array when passing / receiving values.**

Here is a detailed example of one the function we use to call the `mlfTestBasicCalc` function:

```csharp
We initialize an input array of double.

double[] input = new double[ ] { 1, 45, 90, 0 };

We instantiate an output array of double.

double[] output = new double[ 4 ];

We get a pointer to a mxArray (an array then used by MATLAB). The last parameter of `mxCreateDoubleMatrix_730_proxy` indicates if yes or no there is a complex value. In our case, there is none.

```csharp
IntPtr x_ptr = WrapperMatlab.mxCreateDoubleMatrix_730_proxy( 4, 1, 0 );
```

We get access to the pointer value (the mxArray).

```csharp
IntPtr xr_ptr = WrapperMatlab.mxGetPr_proxy( x_ptr );
```

We copy our input array to the mxArray. Note that the mxArray is a matrix of 4 rows and 1 column, so we have just enough space.

```csharp
Marshal.Copy ( ( double[ ] ) input, 0, xr_ptr, input.Length );
```

We instantiate an IntPtr, which will be used to get the return value of `mlfTestBasicCalc`.

```csharp
IntPtr y_ptr = IntPtr.Zero;
```

We pass the last instantiated IntPtr as a ref, to simulate an ** mxArray. Note that the first parameter is the number of output argument (1 in our case).

```csharp
try
{
    WrapperTestMatlabIntegration.mlfTestBasicCalc( 1, ref y_ptr, x_ptr );
}
catch ( Exception e )
{
    Console.WriteLine ( e.Message );
}
```

We get access to the pointer value (the mxArray) of the return value.

```csharp
IntPtr yr_ptr = WrapperMatlab.mxGetPr_proxy( y_ptr );
```
We copy our return value to our output array.

```csharp
Marshal.Copy ( yr_ptr, (double[]) output, 0, 4 );
```

We destroyed the mxArray (since there is no garbage collector in C, this is necessary).

```csharp
WrapperMatlab.mxDestroyArray_proxy ( x_ptr );
WrapperMatlab.mxDestroyArray_proxy ( y_ptr );
```

As example, here is our results when we call the three functions previously created from Unity3D (using Debug.Log() to get the return values):

<table>
<thead>
<tr>
<th>Function</th>
<th>Time in Unity3D</th>
<th>Time in MATLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0273</td>
<td>5.5645e-04</td>
</tr>
<tr>
<td>2</td>
<td>0.0024</td>
<td>0.0018</td>
</tr>
<tr>
<td>3</td>
<td>0.0531</td>
<td>0.0996</td>
</tr>
</tbody>
</table>

We can see that we get the same values than with MATLAB. Concerning the time, if we compare it to the time got in MATLAB.

There are some minor differences, but we have the same kind of value. It is as fast to call MATLAB functions from a DLL as from MATLAB.

**Bug and workaround**

MATLAB version R2014a as a known bug which may cause some MATLAB functions to not work properly, even if they should. *This bug may be resolved in the next version of MATLAB.*

You are likely to have installed your 32-bit MATLAB version in C:/Program Files (86x) / ..., but this path contains parenthesis which may cause MATLAB Application Compiler to not link every functions when it generates the DLL.

Here is the detailed workaround:

[http://www.mathworks.com/support/bugreports/1059029](http://www.mathworks.com/support/bugreports/1059029)
Annex 9 – SVM in Unity3D

In order to do implement a SVM, we decided to use libsvm, which has been use in several research papers.

Note that a practical guide on how to use SVM is available through libsvm:

Implementation of libsvm under C# .NET

Several implementations of libsvm under C# .NET have been done. We decided to use the one done by Nicolas Panel, since it the more updated one:
https://github.com/nicolaspanel/libsvm.net.

To do so, we

- Install NuGet Package.

Under Visual Studio: Tools > Extension and Updates > Online > NuGet Package Manager

- Install libsvm using NuGet Package.

Under Visual Studio: Tools > NuGet Package Manager > Package Manager Console

In the manager console: Install-Package libsvm.net

After installation, you should see several new references in the Solution explorer, including libsvm and libsvm.NET.

Implementation of libsvm under Unity3D

Preparation

LIBSVM.NET.dll has been built under the 4.0 .NET framework. Since Unity3D, so far, only implement .NET up to 3.5, we need to rebuild LIBSVM.NET.dll. To do so, download LIBSVM.NET (https://github.com/nicolaspanel/libsvm.net) and open it in Visual Studio.

In Visual Studio, in solution explorer, right click on the LIBSVM.NET project > Properties > Application > Target Framework 3.5.

In Visual Studio, in solution explorer, right click on the solution > Build.

If there is an error during the build, try to find a work-around.

We will use the LIBSVM.NET.dll that has been built.

Implementation

Under Unity3D, in the Assets folder, create a Plugins folder.

In Assets > Plugins, copy your DLL libsvm, LIBSVM.NET.dll, as well as the four IKVM dll.

From now on, you can use libsvm. Just use the namespace System.Linq and libsvm.

using System.Linq;
using libsvm;

Test of libsvm under Unity3D

To test it, we created a script which records the position of a cube. The cube activates triggers which classify its position in three classes: Red, Green, Blue.

Below is an example of a training/test file which can be used by SVM:

Below is the scene we used to train/test the SVM. The cube can move above the different colored squared.
The format is: +[class][index]:[value][index]:[value]...

In our case, we use three classes, the index 1 is the x position of the cube, the index 2 is the y position of the cube, the index 3 is the z position of cube.

We then created a script to train (using grid-search) and test a SVM (RBF Kernel). We copy it here without comments, since it is pretty straightforward.

```csharp
using UnityEngine;
using System.Collections;
using System;
using System.Collections.Generic;
using ikvm;
using IKVM;
using libsvm;

public class TestSVM : MonoBehaviour
{
    public bool doGridSearch = false;
    public Transform toCheck;
    C_SVC svm;

    // Use this for initialization
    void Start ()
    {
        svm_problem prob = ProblemHelper.ReadAndScaleProblem ( MyUtils.FileTraining );
        svm_problem test = ProblemHelper.ReadAndScaleProblem ( MyUtils.FileTest );
        svm_parameter parameters;
        if ( doGridSearch == true )
        {
            parameters = GridSearch ( prob, test );
        }
        else
        {
            parameters = new svm_parameter();
            parameters.C = 1.0;
            parameters.gamma = 0.0625;
        }
        svm = new C_SVC ( prob, KernelHelper.RadialBasisFunctionKernel ( parameters.gamma ), parameters.C );
        var accuracy = svm.GetCrossValidationAccuracy ( 5 ); // with nr_fold > 1
        Debug.Log ( "Accuracy: " + accuracy );
        StartCoroutine ( Prediction ( ) );
    }

    // Update is called once per frame
    void Update ()
    {
    }

    svm_parameter GridSearch ( svm_problem prob, svm_problem test )
    {
        double c = 0.0;
        double gamma = 0.0;
        svm_parameter bestParameters = new svm_parameter ( );
        double bestAccuracy = 0.0;
        for ( double i = -5; i < 10.0; i++ )
        {
            c = Math.Pow ( 2, i );
            for ( double j = -5; j < 10.0; j++ )
            {
                gamma = Math.Pow ( 2, j );
                svm = new C_SVC ( prob, KernelHelper.RadialBasisFunctionKernel ( gamma ), c );
                double accuracy = svm.GetCrossValidationAccuracy ( 5 ); // with nr_fold > 1
                Debug.Log ( "Best accuracy so far: " + bestAccuracy + " Accuracy:" + accuracy + " C:" + c + " Gamma:" + gamma );
                if ( accuracy > bestAccuracy )
                {
```
```
The cross validation give us an accuracy of 0.96%.

Note that this script also check in real time the position of the cube, and predict if it is in a Red, Blue or Green zone by using the SVM classifier.

That’s all.
Annex 10 – ProvokeAffect V2.0

We provide here some screenshot of the ProvokeAffect V2.0. Note that everything is automated.

Figure 54; First scene.

Figure 56; Scene 3, the rule of the application.

Figure 55; Second scene, to get basic information from the subject.
Annex 11 – Feature extraction and emotion recognition

We present here in more details the class diagrams used in the features extraction and emotion recognition implementation.

We present below the classes we use to extract the features.

- **WrapperMatlab** is the wrapper of the MATLAB Compiler DLL: mclmcrt8_3.dll.
- **WrapperFeaturesExtraction** is the wrapper of the MATLAB’s ECG / EDA algorithm’s DLL.
- **HelperMatlab** contains helper functions to marshall C#/MATLAB’s data.
- **ECGFeatures** contains the ECG features we need to detect the affective state.
- **EDAFeatures** contains the EDA features we need to detect the affective state.

- **BITalinoEventArg**, in this implementation, is an event which is fired when a frame is read by **BITalinoReader** (*not represented on this class diagram*). **BITalinoEventArg** is handled by **DataTreatment**, which save the frame read in bufferData.

- **AffectiveTreatmentEventArg**, in this implementation, is an event which is fired when **AffectiveTreatment** do the affective treatment. **AffectiveTreatmentEventArg** is handled by **AffectiveRecognition** (*not represented on this class diagram*), which recognize the affective state from ECG / EDA features.

**DataTreatment** save the frames read by **BITalinoReader**. When a trigger (depending of the bufferSize or of the Time) is triggered, **DataTreatment** call ExecuteTreatment.

- **AffectiveTreatment**, in top of what is doing **DataTreatment**, implements ExecuteTreatment, which execute MATLAB’s ECG / EDA algorithms (through **WrapperMatlab** / **WrapperFeaturesExtraction**).
We present in this part the classes we use to recognize the affective state.

- **SVMManager** handle **SVMMModel** and **SVMIO**, as well as several parameters. In order to access the SVM classifier, other scripts should communicate with **SVMManager** instead of **SVMMModel** or **SVMIO**.
- **SVMMModel** contains the SVM implementation.
- **SVMMModelRBF** implement the Radial Basis Function SVM.
- **SVMIO** handle the input/output/parsing of raw data, so that they can be used by the SVM.
- **SVMIOAffectiveRecognition** parse the ECG / EDA features.

- **AffectiveRecognition** is the final stone of our work. It makes the junction between **AffectiveTreatment**, which give ECG / EDA, and **SVMManager**, which recognize the affective state.
- **SVMAffectiveRecognition** contains an example of how-to do the junction between **AffectiveTreatment** and **SVMManager**, and display the affective state which is recognized.
Except that it cannot be extend, there is not a lot to say about this class.

```csharp
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public struct ParameterAffectiveDDA
{
    public float Min { get; set; }
    public float Max { get; set; }
    public bool Active { get; set; }
    public int Influence { get; set; }

    public ParameterAffectiveDDA(bool active, int influence, float min, float max)
    {
        Active = active;
        Influence = influence;
        Min = min;
        Max = max;
    }
}

public class AffectiveDDARehabIsland : MonoBehaviour
{
    public AffectiveDDA affectiveDDA;
    public ParameterAffectiveDDA speed;
    public ParameterAffectiveDDA size;
    public ParameterAffectiveDDA distance;
    public ParameterAffectiveDDA interval;

    protected float currentDifficulty = 0.5f;

    TaskParameterWindowModel model;
    TaskParameterWindowView view;

    void Start()
    {
        model = SceneManager.GetComponent<TaskParameterWindowModel>();
        view = SceneManager.GetComponent<TaskParameterWindowView>();
        float Log(float min, float max) // EQ = LN ( DIFFICULTY / ( 1 / EXP ( MAX ) - EXP ( MIN ) ) ) + EXP ( MIN )
        {
            return Mathf.Log( currentDifficulty / (1.0f / Mathf.Exp( max ) - Mathf.Exp( min )) + Mathf.Exp( min ));
        }

        float Exp(float min, float max) // EQ = EXP ( DIFFICULTY / ( 1 / LN ( MAX - ( MIN - 1 ) ) ) ) + ( MIN - 1 )
        {
            return Mathf.Exp( currentDifficulty / (1.0f / Mathf.Log( max - (min - 1.0f))) ) + (min - 1);
        }

        float Linear(float min, float max) // A * DIFFICULTY + B ; b = MIN ; a = ( MAX - MIN )
        {
            float b = min;
            float a = (max - min);
            return a * currentDifficulty + b;
        }

        float InverseSymetric(float min, float max, float val) // EQ = MIN + ( MAX - X )
        {
            return (min + (max - val));
        }
    }
```
void ChangeDifficultyHandle ( object sender, DDAEventArg e )
{
    currentDifficulty = e.Difficulty;
    if ( speed.Active == true )
    {
        UpdateSpeed ( Linear ( speed.Min, speed.Max ) );
    }
    if ( size.Active == true )
    {
        UpdateSize ( InverseSymetric ( size.Min, size.Max, Linear ( size.Min, size.Max ) ) );
    }
    if ( distance.Active == true )
    {
        UpdateDistance ( ( InverseSymetric ( distance.Min, distance.Max, Exp ( distance.Min, distance.Max ) ) ) );
    }
    if ( interval.Active == true )
    {
        UpdateInterval ( ( InverseSymetric ( interval.Min, interval.Max, Log ( interval.Min, interval.Max ) ) ) );
    }
    model.DispatchEvent ( DispatchedEvent.TASK_PARAM_IS_UPDATED );
}

void UpdateSpeed ( float val )
{
    // CHANGE VALUE
    view.scrollbar [ Scrollbar.Velocity ].fValue = val;

    // UPDATE VALUE
    model.objParam.velocity = MathController.Instance.ToRoundDown ( view.scrollbar [ Scrollbar.Velocity ].fValue, 1 );
    view.label [ Label.VelocityValue ].label = model.objParam.velocity.ToString ( );
}

void UpdateSize ( float val )
{
    // CHANGE VALUE
    view.scrollbar [ Scrollbar.Size ].fValue = val;

    // UPDATE VALUE
    model.objParam.size = MathController.Instance.ToRoundDown ( view.scrollbar [ Scrollbar.Size ].fValue, 1 );
    view.label [ Label.SizeValue ].label = model.objParam.size.ToString ( );
}

void UpdateDistance ( float val )
{
    // CHANGE VALUE
    view.scrollbar [ Scrollbar.Distance ].fValue = val;

    // UPDATE VALUE
    model.objParam.distance = MathController.Instance.ToRoundDown ( view.scrollbar [ Scrollbar.Distance ].fValue, 1 );
    view.label [ Label.DistanceValue ].label = model.objParam.distance.ToString ( );
}

void UpdateInterval ( float val )
{
    // CHANGE VALUE
    view.scrollbar [ Scrollbar.UpdateInterval ].fValue = val;

    // UPDATE VALUE
    model.objParam.interval = MathController.Instance.ToRoundDown ( view.scrollbar [ Scrollbar.UpdateInterval ].fValue, 1 );
    view.label [ Label.UpdateIntervalValue ].label = model.objParam.interval.ToString ( );
}
Titre : Adaptation dynamique de la difficulté en fonction des signaux physiologiques, pour un theragame pour enfants atteint d’infirmité motrice cérébrale.

Mots clés : Rééducation, DDA, Children Rehabilitation Project, Infirmité motrice cérébrale.

Résumé : L’objectif de cette recherche était de développer une adaptation dynamique de la difficulté (DDA) en fonction de l’affect, afin de l’intégrer dans une application de rééducation après AVC (et plus spécialement dans le Children Rehabilitation Project (CRP)). Le Children Rehabilitation Project est un logiciel destiné à la rééducation après AVC, et a pour objectif de devenir une plateforme capable de fournir une meilleure prise en charge de la rééducation après AVC pour les enfants, en se concentrant sur les jeux sérieux. Dans ce rapport, nous présentons l’ensemble des étapes de notre recherche, en commençant par (1) l’état de l’art sur les DDA et sur les DDA dépendante de l’affect ; et en passant par (2) la description de notre approche basée sur l’affect, (3) l’acquisition de signaux physiologiques, (4) l’extraction des paramètres de ces signaux physiologiques et (5) l’entraînement d’un learning algorithm capable de changer dynamiquement la difficulté en fonction de l’affect d’un utilisateur. Même si nous sommes parvenu, à travers cette recherche, à implémenter dans le CRP une DDA basée sur l’anxiété et l’ennui ; il est nécessaire, dans une future recherche, de l’expérimenter afin de déterminer son impact sur la rééducation des enfants ayant subis un AVC.

Title : Physiological-based Dynamic Difficulty Adaptation in a Theragame for Children with Cerebral Palsy

Keywords: Rehabilitation, DDA, Children Rehabilitation Project, Cerebral Palsy.

Abstract : The purpose of this research is to provide a physiological-based Dynamic Difficulty Adaptation (DDA) able to be used in a CVA rehabilitation application, and more especially in the Children Rehabilitation Project (CRP). The CRP is an adaptive modular interface destined to CVA’s rehabilitation, and has for purpose to offer a state-of-the-art rehabilitation platform to improve CVA’s rehabilitation through rehabilitation applications. In the report, we go through all the steps of this research, by going through (1) the state of the art done on DDA and on physiological-based DDA, (2) the approach based on the emotional state, (3) the acquisition of physiological signals, (4) the extraction of the physiological signals’ features and (5) the training of a learning algorithm able to dynamically change the difficulty. While we successfully managed to implement a physiological-based DDA based on Anxiety and Boredom emotional state (in order to be able to use the flow theory) in the CRP, we still need to experiment it on children in CVA’s rehabilitation.