Towards the Development of an Affect-Sensitive Game

Jonathan Jason King Li[1]
jcokingli@gmail.com

Tricia Angela Monsod[1]
trish.monsod@gmail.com

Paul Julian Hao[1]
chocobo251991@yahoo.com

Gabriel Luis Matias[1]
ro_gunbound@yahoo.com

Joshua Rex Cheng[1]
jrexcheng@yahoo.com

Nicole Guloy[1]
ikin_ari@yahoo.com

ABSTRACT
The purpose of this study is to create a brain-computer interface-dependent game that relies on user affect as a method of control. The paper first summarizes the existing research into BCI-based applications. It then describes the steps the researchers have been taking in order to develop a game that makes use of the OCZ Neural-Impulse Actuator to retrieve brain signals from a user in order to manipulate different powers utilizing arousal.

1. INTRODUCTION
1.1 Context of the Study
A game is "a closed, formal system that engages players in structured conflict and resolves in an unequal outcome" [4]. [4] further elaborates this definition by breaking it down into its key terms and phrases.

The term "closed" means that a game has boundaries that define itself from the real world with the implementation of its own rules and consequences. Once the game is over, these consequences have no effect on the real world aside from giving entertainment to its player(s). A game is a "formal system" because it is made up of formal elements such as players, objectives, challenges and plot; all of which interact with each other that define a game and not some other interaction. A "structured conflict" simply means that the player has a certain goal to achieve in a game; however, to fully engage a player to achieve that goal, the player is introduced to certain rules and challenges that obstruct or help the player during the course of the game. Finally, the phrase "resolves in an unequal outcome" means that there is a winner or winners in the game. This definition does not discount simulation games or open-ended games as long as they "provide both a resolution and measurable achievement to their players" [4].

Traditionally, games are controlled using conventional modalities such as the mouse, keyboard or a controller. In recent years, there has been an attempt to make games that respond to human affect, or human cognitive processes such as interest, attention, confusion, frustration, irritation and the like [8].

1.2 Review of Related Literature
A brain-computer interface (BCI) is a connection link between a person's brain and a computer that is formed using another device usually worn on the head. BCI devices can measure voltage differences across the scalp. These electrical brain signals are sent to a computer and are converted into a recognizable pattern which can identify specific cognitive functions in the brain, forming a means of communication between brain and computer [6]. The BCI input modality allows a user to directly interact with a computer without having to use their hands [9].

1.2.1 BCI Possibilities
BCIs have been used mostly in the context of disabled, paralyzed patients [2, 8, 9]. Because of certain limitations in terms of the patient's motor or brain control, the BCI helps the patient overcome the damaged part of their body and operate a program or a piece of equipment much like a regular person could - of course, with a different method of control [9]. BCIs can also be used to measure the affect of test subjects as they perform certain tasks, therefore having great potential in becoming a method by which games can be controlled. Several experimental research games have already been developed for various devices, but due to their experimental nature they remain small-scale and limited [9].

While some BCIs need to be implanted in the human being under the supervision of trained medical personnel, other types come in the form of dry caps which are relatively cheap and easy to set up. This type of device is commonly non-invasive and provides convenient hands-free interaction for users [6]. Quite logically, it is this type of BCI that is ideal for gaming.

Of particular interest in this study is the relation of the brainwaves these BCIs read to affect. This implies that applications can be created that respond directly to or can be controlled by affect modulation. According to the research of [1], a person's affect is a behavioral expression of an emotion. Further studies were made by [3] and from the game development, arousal can be manipulated to express high levels of beta waves over alpha waves. This claim of arousal which implies that the player exhibits more beta waves than alpha waves when aroused is also supported by [1]. Therefore one can imply here that the affect, which is a behavioral expression of an emotion, is directly related to arousal which increases even if the player simply poses behavioral expressions of "active" emotions.

1.2.2 Works on BCI Implementation
One of the previous attempts to applying BCI in games was an experiment conducted by [12] through the game of Affective Pacman. Control was done via key presses, and BCI's task in this experiment was to measure the induction of frustration. This was done by building around 2 minute blocks to the controls of the game where 1/3 of them were infused with frustration condition, popping out randomly throughout the course of the game. These blockers induced frustration in terms either the user input or alternatively, the visual output. Whichever the case, the bulk of this experiment laid with the assessment of the user's state, measured using the Self Assessment Manikin (SAM).

Electroencephalogram (EEG) sensors that record electrical brain activity for BCI, is used in conjunction with eye and muscle movements to populate time-frequency plots in relation to the

frequency of key-presses made throughout the game for both the left and right motor cortices. From these plots, differences where present, indicating the presence of Event Related Desynchronization (ERD) to make a comparison between normal and frustration conditions. The result of their experiment led them to the conclusion that inducing frustration in a user deteriorated their BCI performance. The success of this experiment was due to the fact that users who played their game checked off the frustration induction was a mere bug in the game, therefore did not dwell too much on it [12].

1.2 Research Objectives

This study aims to design and build an affect-sensitive game whose main input is BCI-dependent. Here, being 'BCI dependent' implies that the main mode of game control shall be through brainwaves, while being 'affect-sensitive' means that the game will be responsive to changes in the user's affective state.

1.3 Scope and Limitations

The game that is to be developed for this research paper will make use of the OCZ Neural-Impulse Actuator, also known as the NIA. This device has only three sensors on the forehead as compared to the usual 10 or more, which makes it more limited than other BCIs in terms of the type of brainwave data it can detect. This will be used in conjunction with the Brainfingers Access Software that serves as a bridge between reading the brain signals and feeding them to the game program.

The readings that the researchers will focus on are the alpha and beta waves since the research focuses on arousal. According to [1] alpha waves are more emitted than beta waves when one is in a state of relaxation, physically and mentally but are aware of our surroundings. Beta waves are emitted more than alpha waves when one is fully conscious and alert. The limitations of the device also make it difficult to monitor valence, so currently arousal is the only basis of classification.

It remains outside the scope of this research to accurately classify emotion as mentioned by [3], who were the predecessors of this ongoing research. To illustrate an example, being angry implies having high levels of alpha and beta waves, but it is possible for a player to exhibit high levels of alpha and beta waves but the player is not angry [3]. The researchers choose to disregard how the user may actually be feeling, as long as they manage to produce the required change in affect to manipulate the game. Similarly, the extent of feature extraction and classification will be limited and highly simplified, as further discussed. Feature extraction and classification training is not practical given the research time frame and accessibility to sample data.

Another issue is using the signals themselves. By default, most developers, such as [3], must avoid making use of the EMG signal. This is the signal which is caused by muscle movement, and thus any small gesture or act of distraction by the user may throw the game off. Game designers definitely do not want to ask their gamers to sit stock still while playing, thus this signal is excluded [8].

Currently, the game that the authors of this paper hope to develop is a work in progress, and thus majority of the procedures and implementations are experimental in nature. They have been looking into a game designed around the concept of neurofeedback.

2. METHODS

2.1 Instruments

To develop the affect-sensitive game, the researchers make use of the OCZ Neural-Impulse Actuator (NIA) and Headset. It is a noninvasive device that requires minimal set-up time. The headset consists of three diamond shaped conductor plates that will be bound to the forehead by a rubber headband, as shown in Figure 1. This is connected to the NIA device, which is then connected by USB to a computer. The computer must be properly grounded to lessen electromagnetic interference.

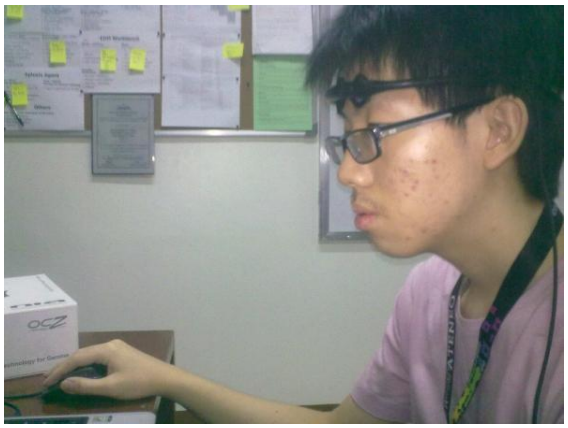

Figure 1: A test subject wearing the OCZ NIA

The software that reads the signals from the OCZ-NIA is the Brainfingers Access Software. Software version 2.0.5.23 nia1.0 is the particular version of Brainfingers software being used by the researchers that allows the writing of Brainfingers data to a shared memory space. This allows programs to access the shared memory space and get data in real time. In this case, the program would be the game.

2.2 Procedure

The researchers have divided the development process into three major activities, which they refer to as phases of development. As these phases can be undergone in parallel to one another and can be interchanged throughout the entire course of the research, there is no specific order to them, thus making implementation plans prone to constant updates and modifications.

2.2.1 Arousal Detection and Interpretation Phase

Using the Brainfingers shared memory function that the OCZ NIA software generates allows the researchers to access brainwave data. For this research, the researchers are more interested in the alpha and beta signals, primarily important aspects to computing for arousal as mentioned in Chapter 1.3.

Previous research has shown that arousal can be detected by "a higher beta power and coherence in the parietal lobe, plus lower alpha activity" [1], which they also refer to as the "beta-alpha ratio" [1]. To emulate this, the researchers have taken the beta and alpha readings from the Brainfingers software, and estimate

$$\text{Arousal} = \text{Beta} / \text{Alpha}$$

to determine the arousal level of a user. The effectiveness of this equation has been tested and they have shown that another understanding of arousal allows players to have better control of their gameplay. [1] mentions that arousal implies having higher beta levels than alpha levels. Therefore, the researchers have modified the equation to estimate arousal as a condition where

$$\Sigma \text{ Beta} >= \Sigma \text{ Alpha}$$

that causes players to sustain arousal and meet their goals much more freely. This condition also allows players to adjust accordingly to any shifts in their own arousal.

2.2.2 Game Design and Development Phase

The researchers have defined the various game elements needed, including the context of the game, gameplay, and the incorporation of stimuli into the game environment through brainstorming and consultation. With the basics of the game design covered, the researchers have since begun the game development.

In its initial stages of development, placeholder graphics are being used to ensure that the UI reacts appropriately to the input that is picked up from the NIA device. As development continues, the researchers are slowly revamping the graphics found in-game, creating new art assets for both the background as well as interactive objects to be manipulated by the powers to promote a visual consistency.

A training stage will be developed so participants may familiarize themselves with the game controls and practice manipulating their affect accordingly. This is needed, as various online user reviews have reported that use of the OCZ NIA requires much familiarization before anyone is able to sufficiently control a game.

In general, for this phase, the player's affect is measured through the use of the Brainfingers and is then converted into signals. The alpha and beta signals are converted into values, which then determine what in-game function to be executed.

2.2.2 Agent Design and Development Phase

To guide first time participants accordingly, an agent is currently being developed to assist the player in familiarizing with the new modality by acting as the protagonist's mentor that will only be present through dialogue that the player reads on the screen.

An agent is "a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives" [10]. This implies that an agent then is a piece of software that performs an objective while being independent from user control given a virtual environment. However, an agent reacts to the changing virtual environment that is usually caused by the player. In theory, based on the actions of the player, the agent can assist the player.

The agent design will theoretically be affect-sensitive, interpreting the brain signals of the player in conjunction with an internal timer that will measure how long the player spends attempting a particular affect-sensitive task. After a certain amount of time, the agent will activate and offer the player something else to think about should they be emitting the wrong arousal level.

2.3 Game Details

2.3.1 Context and Premise

Gifted with extraordinary powers, the protagonist is sent to attend School of Thought, a school that allows one to control such dangerous abilities that they control with their mind. Graduates of this school have gone on to become defenders of peace, using their powers for good. The player controls this protagonist when they wake up inside the confines of a dungeon cell. The protagonist, sensing that they have been kidnapped, makes use of his innate powers to break out of the cell and find an escape route to avoid getting experimented on.

2.3.2 Gameplay

Due to the limitation of the device, navigation and selection of objects will be done using mouse and keyboard input. The powers themselves, however, will be controlled through the BCI. These powers are indicated by icons on the interface, and are cycled through by the BCI device by manner of muscle movement, usually by clenching one's teeth. Once a power has been activated, the BCI device will read the player's affect and through use of the formula, compute for arousal to accomplish a particular task.

Throughout the game, the player will find various objects that can be interacted with using the different powers of the protagonist. The game, divided into three stages, has objectives related to these objects that the player must complete in order to progress.

The powers have all been designed based on arousal level, each power requiring the player to manipulate their arousal levels in different ways. Pyro has two different effects: maintaining a mid-range level of arousal will let the object smolder, while abruptly increasing arousal will set it ablaze. Ghost is an ability that allows the protagonist to pass through obstacles, as long as the game player maintains a low level of arousal. The previous version developed by [3] had a power called Ecological Empathy, which required the player to steadily increase their arousal in order to help plants grow. It has been changed to Telekinesis which instead requires the player to maintain their arousal level within a certain range for an amount of time before the range is re-randomized, requiring the player to change their arousal level until the object is fully "grown".

2.4 Implementation Details

Each GameStage.cs class will utilize several common components which can be encapsulated further into subclasses. Because of this, the first game engine that [3] developed was what they consider to be a mini game-engine. They developed a class library, which contains 'utility classes' such as Areas, Meters, and the generic Power class, which takes care of Brainfingers data access, normalization and classification. Each stage has its own set of areas and interactive objects that can be modified by the programmer in any way he wants. The programmer can add a number of objects and stages to his liking by simply specifying parameters such as map positions and the type of power associated with it. The next step is just a matter of adding them to the specific area of choice. Once this has been set up, the mini game-engine can be used to create any number of stages.

It is with the class Area.cs that the agent will primarily interact with. Aside from handling interactive objects for its screen, each Area will also have a list of things the agent can say pertaining to the certain task at hand. Ideally, the agent will be able to blurt out tips when it senses that the player is having a hard time accomplishing one of the mental tasks.

The code snippet for manipulating burnable objects' data as shown in the class Pyro.cs is shown in Figure 2. The method `manipulateObjectData()` is called per update cycle of the game, and then it checks for the current arousal level of the player (denoted by the if-else statements). It then sets the current state of the object. The object will then appear as burnt, burning, or smoking. The methods `isSmoldering()` and `isAblaze()` return true if and when the player's current arousal level is within the proper range.

```
public override void
manipulateObjectData(ObjectData objectData,
GameTime theGameTime)
{
       Vector3 c = objectData.getColor();
       if (isSmoldering())
       {
              objectData.setSmoking(true);
              if (c.X > 0.3)
                    objectData.setColor(new
                    Vector3(c.X - BURN_RATE,
                    c.Y -BURN_RATE, c.Z -
                    BURN_RATE));
              else
              {
                    objectData.
                    setSmoking(false);
                    objectData.
                    setFinished(true);
                    objectData.
                    setBlazing(true);
              }
       }
       else if (isAblaze())
       {
             objectData.setColor(new
                    Vector3(0, 0, 0));
             objectData.setSmoking(false);
             objectData.setFinished(true);
             objectData.setBlazing(true);
       }
       else
       {
              objectData.setSmoking(false);
              if(c.X<1)
                    objectData.setColor(new
                    Vector3(c.X +
                    BURN_RATE/2, c.Y +
                    BURN_RATE/2, c.Z +
                    BURN_RATE/2));
       }
}
bool isSmoldering()
{
      return Power.getInstance().getArousal()
      >= SMOLDER &&
      Power.getInstance().getArousal() <=
      BLAZE;
}
bool isAblaze()
{
      return Power.getInstance().getArousal()
      >= BLAZE;
}
```

Figure 2: Pyro.cs code snippet

Figure 3 features a code snippet taken from the class Power.cs, which deals with everything related to the OCZ NIA device, its readings, and acquiring them from shared memory. The method `initialize()` connects the program to the shared memory where the OCZ NIA, throught the Brainfingers program, deposits its readings. The method `UpdateInput()` then reads from shared memory and stores the necessary signals in the alpha1-3 and beta1-3 variables for computation. The data is then computed using the formula that was based on research done, and then objects in the game are then manipulated using the values stored in Power.cs.

```
//needed for accessing the shared memory space
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Ipc;


namespace SchoolOfThought
{
class Power
{
//retrieves Brainfingers data
niaSDK.DataInterfaces.IPostProcess
niaProcessedData;
protected void initialize()
{
//===== THE FOLLOWING CODE WAS TAKEN FROM
DOCTOR JUNKER'S SDK CLIENT EXAMPLE =========
//allows access to the shared memory space
where the Brainfingers Access writes the data
IpcChannel channel = new
IpcChannel("niaSDKClient");
ChannelServices.RegisterChannel(channel,
false);
niaProcessedData =
(niaSDK.DataInterfaces.IPostProcess)
Activator.GetObject(
    typeof(niaSDK.DataInterfaces.IPostProcess),
       "ipc://niaSDKServer/Data");
//=======================================END=
=====================================


}

public void UpdateInput()
{
       //retrieve Brainfingers input
       alpha1 = (decimal)niaProcessedData.
             Alpha1Js;
       alpha2 = (decimal)niaProcessedData.
             Alpha2Js;
       alpha3 = (decimal)niaProcessedData.
             Alpha3Js;
       beta1 = (decimal)niaProcessedData.
             Beta1Js;
       beta2 = (decimal)niaProcessedData.
             Beta2Js;
       beta3 = (decimal)niaProcessedData.
             Beta3Js;

       //compute for arousal level
       red = (beta1 + beta2 + beta3);
       blue = (alpha1 + alpha2 + alpha3);
       decimal subRed, subBlue, subGreater;
```

```
        if (red >= blue) //active
        {
               if (arousal < 1.0M)
                   arousal += 0.005M;
               else
                    arousal = 1.0M;
              //change the color of the arousal
              orb
              subBlue = red - blue;
              subRed = 0;
              subGreater = red;
        }
        else //focus
        {
               if (arousal > 0M)
                   arousal -= 0.005M;
               else
                    arousal = 0M;

              //change the color of the arousal
              orb
              subRed = blue - red;
              subBlue = 0;
              subGreater = blue;
        }
tint = new Vector3(1.0f - (float)subRed, 1.0f -
(float)subGreater, 1.0f - (float)subBlue);
}
}
}
```

Figure 3: Power.cs code snippet

2.5 Gameplay Interface

For each game stage, the basic game interface generally looks like the screenshot shown in Figure 4. There are four important components of the interface, as described below.

Interactive objects – As the player moves around the game stage, they will come across different objects that can be manipulated by a specific power, determined by the color by which they are glowing (ie. Red is for Pyro, Blue is for Ghost and Green is for Telekinesis). Selecting the object with the corresponding power will bring up the arousal meter.

Arousal Meter – This appears when the player has activated one of their three powers. It indicates the zone in which the player's arousal level is currently located. The design of the meter body differs for each of the powers. Higher alpha indicates leftward movement, while higher beta indicates rightward movement.

Arousal Orb – This indicates arousal level by means of alpha and beta waves. Higher alpha makes the orb bluer, while higher beta makes the orb redder. This serves as feedback for the player in their attempt to manipulate their affect.

Active Power – An icon that shows which power is currently the active power. A player can switch between powers by tensing their muscles by way of clenching their jaws or raising their eyebrows.

The stationary agent textbox has been added to the interface to house any and all dialogues from the agent that will guide and motivate the player towards the desired arousal level for the particular task at hand.

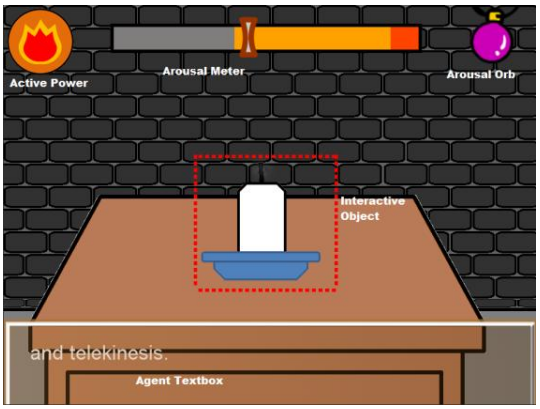

Figure 4: The gameplay interface

3. NEXT STEPS

Full implementation of all the plot-related stages is to be done next. This involves simply hard-coding stage designs and adding stage elements into the game, as well as completing the development of the agent to effectively help first-time users with the modality.

Once the game is completed, formal testing of the software can begin. This step will determine whether the game is actually playable by the participants and whether the software actually achieves its purpose or not.

Nijholt has been quoted to have stated the following: "In recent years we have seen a rising interest in brain-computer interfacing for human-computer interaction and potential game applications. Until now, however, we have almost only seen attempts where BCI is used to measure the affective state of the user or in neurofeedback games. There have hardly been any attempts to design BCI games where BCI is considered to be one of the possible input modalities that can be used to control the game" [9]. This is a great encouragement to game players and developers alike. While this game may still be far from becoming commercial and marketable, joining in the quest for an interesting affect sensitive game and taking these steps down the road towards affective innovation is truly proving to be rewarding.

4. ACKNOWLEDGMENTS

The authors would like to thank Dr. Ma. Mercedes Rodrigo for being an ever patient mentor and adviser. They would also like to thank Dr. Andrew Junker for giving them access to the shared memory features of his Brainfingers Access Software. Finally, they would also like to thank the Department of Science and Technology's Philippine Council for Advanced Science and Technology Research and Development for the research grant entitled "Development of Affect-Sensitive Interfaces" which has enabled them to continue on with the research they are performing to this day.

5. REFERENCES

[1] D. O. Bos. EEG-Based Emotion Recognition: The Influence of Visual and Auditory Stimuli. University of Twente, The Netherlands, 2008.

[2] D. O. Bos and B. Reuderink. Brainbasher: a bci game. Extended Abstracts of the International Conference on Fun and Games, 2008.

[3] J. R. Cheng and N. Guloy. Towards the Development of an Affect Sensitive Game. Ateneo de Manila University, 2010.

[4] T. Fullerton, C. Swain and S. Hoffman. Game Design Workshop. CMP Books, 2004.

[5] G. G. Molina, T. Tsoneva, and A. Nijholt. Emotional brain- computer interfaces. Institute of Electrical and Electronics Engineers, 2009.

[6] C. Muhl, H. Gurkok, D. P.-O. Bos, M. E. Thurlings, L. Scherffig, M. Duvinage, A. A. Elbakyan, S. Kang, M. Poel, and D. Heylen. Bacteria hunt: A multimodal, multiparadigm bci game. Workshop Report for the eNTERFACE Workshop in Genova, Italy, 2009.

[7] C. Muhl and D. Heylen. Cross-modal elicitation of affective experience. Institute of Electrical and Electronics Engineers, 2009.

[8] A. Nijholt. Bci for games: A 'state of the art' survey. International Federation of Information Processing, 2008.

[9] A. Nijholt, B. Reuderink, and D. O. Bos. Turning shortcomings into challenges: Brain-computer interfaces for games. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2009.

[10] L. Padgham and M. Winikoff. Developing Intelligent Agent Systems. John Wiley & Sons, Ltd, 2004.

[11] B. Reuderink. Games and brain-computer interfaces: The state of the art. Internal Report, 2008.

[12] B. Reuderink, A. Nijholt, and M. Poel. Affective pacman: A frustrating game for brain-computer interface experiments. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2009.

6. AUTHORS' INSTITUTIONAL AFFILIATIONS

[1] Ateneo Laboratory for the Learning Sciences, Department of Information Systems and Computer Science, Ateneo de Manila University, Loyola Heights, Quezon City, Philippines, +63 (2) 426-6001 loc 5666, http://penoy.admu.edu.ph/~alls