NeuroBlock: A Block-Based Programming Approach to Neurofeedback Application Development

Chris S. Crawford
University of Alabama
Department of Computer Science
Tuscaloosa, Alabama, USA
crawford@cs.ua.edu

Juan E. Gilbert
University of Florida
Dept. of Computer & Information Science & Engineering
Gainesville, Florida, USA
juan@ufl.edu

***Abstract*—Brain-Computer Interface (BCI) applications are gaining popularity as Electroencephalography (EEG) hardware becomes more accessible. BCI technology is used for various purposes such as neurophysiological evaluation, device control, user-state monitoring, and cognitive improvement. Although BCI software platforms exist, there are few systems designed to assist novice programmers with creating BCI applications. We present "NeuroBlock", a block-based programming approach to neurofeedback application development. Motivated by insights presented in BCI and visual languages literature, our system enables novice programmers to build applications that adapt to users' affective state. We evaluated the system's appropriateness by tasking novice programmers with developing neurofeedback applications inspired by previous BCI studies. Our exploratory study with 40 participants demonstrates that novice programmers are capable of developing neurofeedback applications using NeuroBlock. Furthermore, participants had a positive perception of NeuroBlock's usability.**

I. Introduction

Brain-Computer Interface (BCI) systems are used to measure central nervous system (CNS) activity. BCI systems convert CNS activity to artificial output that is then used to replace, restore, enhance, supplement, or improve natural CNS output [28]. BCI functions through acquiring brain signals, identifying patterns, and producing actions based on the observed patterns. This process allows users to interact with their environment without having to use their peripheral nerves and muscles [18]. Electrical signals from the brain were first captured from the cortical surface of animals in 1875 [8]. In 1929 Hans Berger reported the first successful attempt to capture electroencephalography (EEG) signals from the human scalp [4]. In 1973, Jacques Vidal coined the term "Brain-Computer Interface" [27] after developing a computer-based system that acquired electrical signals captured from the scalp and converted it to commands. Since the introduction of general-purpose BCI software platforms in the early 2000s several open-source software applications have been released.

Although BCI hardware, such as caps and headsets are capable of acquiring brain signals, BCI software is essential to the process of creating useful output. Without the software component, the applications mentioned above would not be possible. Although software plays a critical role, BCI software platforms can be difficult for users that lack a strong programming background. To address this issue we present NeuroBlock, a block-based programming approach to neurofeedback application development. Blocks languages are gradually becoming more popular [3]. Furthermore, researchers have recently investigated the use of block-based programming environments (BBPEs) to support end-user programming in various domains [5], [16], [20], [6], [15], [29]. However, to our knowledge, no prior work investigates the use of BBPEs to assist novice programmers with creating neurofeedback applications. To address this gap, we make the following contributions:

- System design of NeuroBlock, a block-based programming approach to neurofeedback application development.
- Preliminary study of NeuroBlock, conducted with 40 novice programmers that investigates the system's usability.

II. Related Works

This research is guided by literature that investigates BCI software platforms. BCISP are frameworks that assist developers with creating BCI applications. These systems typically consist of the following key components: signal acquisition, feature extraction, feature translation, and commands/feedback applications [19]. Many BCISP have been developed that follow this general design.

BCI2000 is one of the earliest general-purpose BCISPs to apply the BCI pipeline concept [25]. The main goals of BCI2000 are to provide researchers with a flexible BCI software platform and provide standard tools for BCI research. BCI2000 is popular amongst BCI experts. BCILAB is an alternative BCISP for BCI experts [14].

Venthur et al. [26] were one of the earliest researchers to investigate ways to ease BCI feedback application development. Their work produced the first Python-based BCISP.

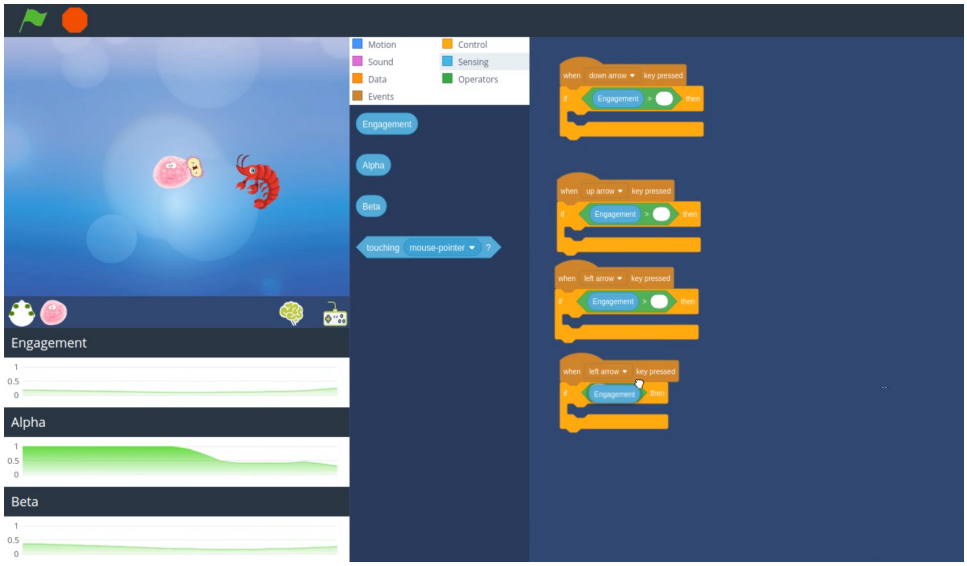

Fig. 1. NeuroBlock interface.

This platform was introduced as interest in Python began to grow within the neuroscience community. Many BCI platforms prior to this system required users to create custom feedback applications using C++, which could be a tedious process for non-programmers.

OpenViBE is another BCISP designed with non-programmers in mind. The goal of OpenViBE is to assist virtual reality developers, clinicians, and BCI researchers with designing testing, and using BCI [24]. OpenViBE was designed to ease the development of BCI applications. Unlike many other BCI platforms, OpenViBE allows users to create complete scenarios using a graphical language. Consequently, prior programming knowledge is not required to create basic BCI applications. However, OpenViBE currently requires programming skills if users wish to create custom feedback applications.

Although block-based approaches have been developed for end-users from various domains, there has been limited research investigating the use of a BBPE for neurofeedback application development. To address this gap we present NeuroBlock, a block-based programming approach to neurofeedback application development. This research will build upon recent advances of the visual programming language community by applying approaches that have been proven to enhance the experience of novice programmers. This work aims to apply the low floors concept to neurofeedback application development. It also involves minimizing the barriers that hinder novices with getting started. To evaluate NeuroBlock, a preliminary investigation of the system's usability is discussed.

III. The NeuroBlock System

A. EEG Data Communication

As shown in figure 2, EEG data was communicated wirelessly using a Bluetooth 2.0 connection between the Muse EEG headset [12] and the computer. Specifically, a Dell

Fig. 2. System design.

Latitude E6530 laptop with a quad core 2.9Ghz Intel i7 CPU was used during the study. The Muse device performs at a sampling rate of 220Hz. A notch filter is applied at 60Hz to remove artifacts such as power line interference. Once the EEG device is paired via Bluetooth to the computer, raw EEG signals are acquired on the computer using MuseIO [13], a research client application provided by Interaxon. EEG frequency band data was transported from MuseIO to a server application developed using the node.js JavaScript runtime environment. EEG frequency bands are associated with affective states and are commonly used in neurofeedback applications [21], [11]. NeuroBlock leverages EEG frequency bands to capture data related to alpha (relaxation) and beta (alertness). The alpha and beta information is scaled between 0 and 1 using a linear function and are called band power scores. The band power scores are communicated to the server application at 10 Hz using the Open Sound Control (OSC) protocol. Once the server receives these scores they are passed to a client web application via WebSockets. Information about the EEG device's channel quality is also communicated to the web application using this protocol.

B. Web Application

The web application component of NeuroBlock was inspired by previous BBPEs such as Scratch [17]. As shown in figure 1, the web interface uses a single-window, multi-pane design to make locating features and navigating the interface easy for users. The interface consists of 3 main components:

stage, affective state feedback, and block interface. As shown in figure 1, the stage component is located near the top left corner of the interface. The stage component contains programmable objects that can be designed to respond to a user's affective state. The stage component was implemented using the open-source Scratch-VM library [1].

Affective state data collected from the EEG apparatus influences multiple feedback components in the interface. This is common in various types of BCI software platforms. It is also common in commercial EEG software such as Emotiv Epoc [9]. These components can be organized in two categories: channel quality viewer and affective state viewer. The channel quality viewer component is positioned directly below the stage component. It consists of 4 circles over a top-down view of a head. This component assists users with making sure the BCI device is mounted properly. The affective state viewer is a component that provides users feedback about their current state. This component is positioned directly below the channel quality viewer. As shown in figure 1, affective state data is presented as line graphs. These graphs display the recent band power session scores values passed from the server.

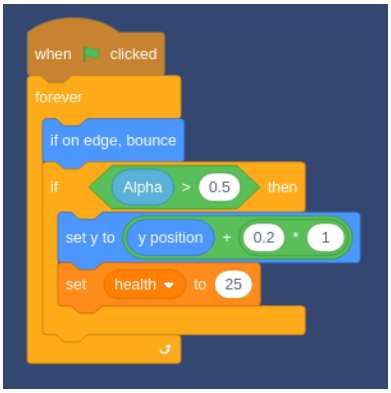

Fig. 3. Simple script that moves a character up when alpha levels are high.

The block interface component provides block elements that are used to create neurofeedback applications. This component draws inspiration from Scratch. It features a command palette that is used to switch between block categories. The block workspace is directly to the right of the command palette. Users use this area to create programs using blocks. Seven block categories are available in the system: motions, sound, data, events, control, sensing, and operators. These block components were implemented using the Scratch-VM library [1]. The sensing block category hold blocks associated with affective state blocks and the mouse. The affective state blocks include alpha, beta, and engagement blocks. The alpha and beta blocks holds values ranging from 0 to 1. These values are derived from a process based on processing EEG frequency band information as discussed in the EEG data communication section. Figure 3 presents a simple script that causes a character to move up when a user's alpha level (relaxation) is higher than 0.5. The engagement (E) block holds values that also range from 0 to 1 and are calculated using a commonly used formula ($E = \beta / (\alpha + \theta)$) [23].

IV. Experiment

To explore a BBPE approach to neurofeedback application development, a user study was conducted. The goal of this study is to gather preliminary insights on the usability of NeuroBlock. Specifically, this work investigates the following questions: How do novice programmers perceive the usability of a block-based neurofeedback development tool?

A. Participants

We recruited 40 student participants from an introductory programming course. The participants' age ranged from 18 to 30. A total of 14 females and 26 males participated in the study. All participants had limited programming experience with Java which was covered in the introductory programming course. Student participants had a wide variety of majors which included computer science, computer engineering, mechanical engineering, electrical engineering, digital arts and sciences, statistics, criminology, and mathematics. Participants were screened to ensure they did not have experience creating neurofeedback applications or using BCI devices. Only two participants had previous experience using a block-based programming environment. In both cases participants reported only faintly remembering using visual programming languages such as scratch while in high school.

B. Procedures

Each of the participants completed a total of 3 think-aloud sessions over the span of 5 days. Due to the length of each neurofeedback development task participants completed 1 session per day. After each session the participants returned for their next session 2 days later. Each session began with a pre-session questionnaire that collected general demographic information.

Once the pre-session questionnaire was complete participants watched a 13-minute tutorial video which explained how to use NeuroBlock's core features. This tutorial only focused on basic examples and did not imply any strategies concerning the best way to use NeuroBlock's features. The 13-minute tutorial video was only shown during session 1. During session 2 participants watched an approximately 2-minute tutorial that introduced the concept of collision. A 2-minute video tutorial was shown during session 3 that discussed how to create object clones. Participants had 20 minutes to complete the pre-task. Prior to the task participants completed a pre-task exercise. During the pre-task, participants were instructed to build a neurofeedback application. Each pre-task was designed to ensure participants were proficient enough to start the session task. During the session task participants were instructed to build an additional neurofeedback application. Screen recording software captured the interface during the task. Participants had 45 minutes to complete the session task. This featured a different application with more instructions and objects. Once participants completed the session task they were given a post-session questionnaire. The post questionnaire included the

System Usability Score (SUS) survey [7]. Participants created a total of three applications that were informed by the bacteria hunt application [22].

V. Results

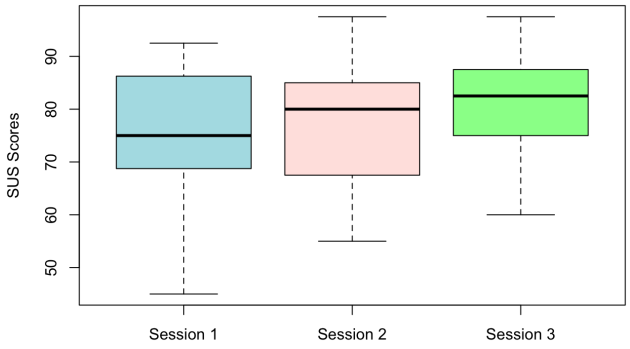

Fig. 4. SUS scores for each session.

Participants reported an average SUS score of 75.50 (SD = 12.07, min = 45.0, max = 92.5) during session one. This score is above average [2]. The session one task was completed in 781.1 seconds on average (SD = 239.88, min = 466, max = 1519). 27.5% of participants completed the task without assistance and 30% completed the task with assistance from the facilitator. During session one 42.5% of participants were unable to complete the task.

During session two participants reported a mean SUS score of 76.38 (SD = 11.38, min = 55.0, max = 97.50) which is also above average. Session two task was completed on average in 797.6 seconds (SD = 219.66, min = 481, max = 1466). 50% of participants completed the task with no assistance from the researcher. 23.6% of participants required assistance to complete the task. 26% of participants failed the task.

Participants reported an above average score of 81.43 (SD = 9.59, min = 55.0, max = 97.50) after completing session 3. Session three was completed in 1018 seconds on average (SD = 261.73, min = 691, max = 1845). 46.7% of participants successfully completed the task with no assistance and 20% of participants completed the task with assistance. The remaining 34% of participants did not complete the task successfully. As shown in figure 4 these results suggest that NeuroBlock was perceived as usable during all three sessions.

VI. Discussion

This work presents the design and evaluation of NeuroBlock, a block-based programming approach to neurofeedback application development. Specifically, a preliminary study is presented that investigates the usability of NeuroBlock. Participants reported above average usability scores during all three sessions. These results suggest that overall participants thought that NeuroBlock was usable and were able to apply techniques related to developing neurofeedback applications using a block-based programming environment.

The findings presented in this work are derived from a multi-session study with 40 participants, but we have not yet conducted a control experiment to compare NeuroBlock to alternative BCI software platforms This is mainly due to other platforms' focus on the signal processing aspect of the BCI pipeline. Given that there are few BCI software platforms similar to NeuroBlock this work takes an exploratory approach. Currently the system only supports three states related to a user's affective state. Going forward it will be vital to implement ways to support several readings from the BCI device. Although this work attempts to investigates a novel approach to neurofeedback application development, further investigation is needed to confirm the appropriateness of this approach.

Despite these positive results, the screen capture analysis showed that formula manipulation was a pain point for participants. Going forward we will integrate a hybrid formula manipulation feature. To address the issue of participants disregarding affective state feedback we will also investigate methods to add additional feedback in the stage area. This may ensure that novice programmers gain a better understanding of how affective state information is influencing their application. Prior to generalizing our results, we plan to conduct a comparison study that evaluates NeuroBlock along side an existing BCI software. The main goal of this work is to expand the reach of BCI technology. To assist with this, we plan to deploy an online version of this tool and follow a model similar to tools such as Scratch and Online Python Tutor [10].

VII. Conclusion

We have presented NeuroBlock, a block-based programming approach to neurofeedback application development. We have discussed NeuroBlock's system design and present a study that investigates the system's usability. Participants rated the usability of NeuroBlock above average. In general, this initial evaluation demonstrates how block-based programming may be leveraged to lower the barrier of entry to BCI technology.

Acknowledgment

This material is based in part upon work supported by Intel Corporation.

References

[1] Llk/scratch-vm: Virtual machine used to represent, run, and maintain the state of programs for scratch 3.0. [Online]. Available: https://github.com/LLK/scratch-vm

[2] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl.Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.

[3] D. Bau, J. Gray, C. Kelleher, J. Sheldon, and F. Turbak, "Learnable programming: blocks and beyond," *Communications of the ACM*, vol. 60, no. 6, pp. 72–80, 2017.

[4] H. Berger, "Über das elektrenkephalogramm des menschen," *European archives of psychiatry and clinical neuroscience*, vol. 87, no. 1, pp. 527–570, 1929.

[5] A. F. Blackwell and R. Hague, "Autohan: An architecture for programming the home," in *Human-Centric Computing Languages and Environments, 2001. Proceedings IEEE Symposia on*. IEEE, 2001, pp. 150–157.

[6] T. Booth and S. Stumpf, "End-user experiences of visual and textual programming environments for arduino," in *International Symposium on End User Development*. Springer, 2013, pp. 25–39.

[7] J. Brooke, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[8] R. Caton, "Electrical currents of the brain." *The Journal of nervous and mental disease*, vol. 2, no. 4, p. 610, 1875.

[9] Emotiv. Emotiv epoc - 14 channel wireless eeg headset. [Online]. Available: https://www.emotiv.com/epoc/

[10] P. J. Guo, "Online python tutor: embeddable web-based program visualization for cs education," in *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 2013, pp. 579–584.

[11] S. I. Hjelm and C. Browall, "Brainball-using brain activity for cool competition," in *Proceedings of NordiCHI*, vol. 7, 2000.

[12] Interaxon. Muse ™ — meditation made easy. [Online]. Available: http://www.choosemuse.com/

[13] ——. Museio - muse developers. [Online]. Available: http://developer.choosemuse.com/research-tools/museio

[14] C. A. Kothe and S. Makeig, "Bcilab: a platform for brain–computer interface development," *Journal of neural engineering*, vol. 10, no. 5, p. 056014, 2013.

[15] D. Krebs, A. Conrad, and J. Wang, "Combining visual block programming and graph manipulation for clinical alert rule building," in *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2012, pp. 2453–2458.

[16] C. Letondal, *Participatory programming: Developing programmable bioinformatics tools for end-users*, ser. End user development. Springer, 2006, pp. 207–242.

[17] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, p. 16, 2010.

[18] S. G. Mason and G. E. Birch, "A general framework for brain-computer interface design," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 11, no. 1, pp. 70–85, 2003.

[19] D. J. McFarland and J. R. Wolpaw, "Brain-computer interfaces for communication and control," *Communications of the ACM*, vol. 54, no. 5, pp. 60–66, 2011.

[20] A. Millner and E. Baafi, "Modkit: blending and extending approachable platforms for creating computer programs and interactive objects," in *Proceedings of the 10th International Conference on Interaction Design and Children*. ACM, 2011, pp. 250–253.

[21] C. Mühl, B. Allison, A. Nijholt, and G. Chanel, "A survey of affective brain computer interfaces: principles, state-of-the-art, and challenges," *Brain-Computer Interfaces*, vol. 1, no. 2, pp. 66–84, 2014.

[22] C. Mühl, H. Gürkök, D. P.-O. Bos, M. E. Thurlings, L. Scherffig, M. Duvinage, A. A. Elbakyan, S. Kang, M. Poel, and D. Heylen, "Bacteria hunt," *Journal on Multimodal User Interfaces*, vol. 4, no. 1, pp. 11–25, 2010.

[23] A. T. Pope, E. H. Bogart, and D. S. Bartolome, "Biocybernetic system evaluates indices of operator engagement in automated task," *Biological psychology*, vol. 40, no. 1, pp. 187–195, 1995.

[24] Y. Renard, F. Lotte, G. Gibert, M. Congedo, E. Maby, V. Delannoy, O. Bertrand, and A. Lécuyer, "Openvibe: An open-source software platform to design, test, and use brain–computer interfaces in real and virtual environments," *Presence: teleoperators and virtual environments*, vol. 19, no. 1, pp. 35–53, 2010.

[25] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "Bci2000: a general-purpose brain-computer interface (bci) system," *IEEE Transactions on biomedical engineering*, vol. 51, no. 6, pp. 1034–1043, 2004.

[26] B. Venthur, "Design and implementation of a brain computer interface system," 2015.

[27] J. J. Vidal, "Toward direct brain-computer communication," *Annual Review of Biophysics and Bioengineering*, vol. 2, no. 1, pp. 157–180, 1973.

[28] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain–computer interfaces for communication and control," *Clinical neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002.

[29] M. Ángeles Serna, C. J. Sreenan, and S. Fedor, "A visual programming framework for wireless sensor networks in smart home applications," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*. IEEE, 2015, pp. 1–6.