Brains and Blocks: Introducing Novice Programmers to Brain-Computer Interface Application Development

CHRIS S. CRAWFORD, University of Alabama
JUAN E. GILBERT, University of Florida

Brain-Computer Interface (BCI) hardware is becoming more affordable and accessible. However, there is limited work investigating ways to design software that broadens participation with BCI technology. In this article, we present a block-based programming environment designed to assist novice programmers with creating BCI applications. We also discuss learning barriers encountered by novice programmers developing neurofeedback applications. Our findings suggest that visual programming assists novice programmers with building basic BCI applications; however, students may experience understanding and learning barriers initially.

CCS Concepts: • **Social and professional topics** → **Computing education**; • **Human-centered computing** → **Interactive systems and tools**;

Additional Key Words and Phrases: Brain-computer interface, block-based programming, neurofeedback

ACM Reference format:
Chris S. Crawford and Juan E. Gilbert. 2019. Brains and Blocks: Introducing Novice Programmers to Brain-Computer Interface Application Development. *ACM Trans. Comput. Educ.* 19, 4, Article 39 (July 2019), 27 pages.
https://doi.org/10.1145/3335815

1 INTRODUCTION

Advances in Brain-Computer Interfaces (BCIs) are enabling the exploration of novel input techniques (Tan and Nijholt 2010). These advances leverage neurophysiological data (e.g., electroencephalography (EEG) (Berger 1929)) that can be used to sense brain activity associated with various cognitive states (e.g., imagined movements, emotional states).

BCI technology has traditionally been used for basic and clinical/translational research that investigates applications that provide assistive care for persons with clinical conditions. Examples of these BCI applications include BCI controlled wheelchairs (Carlson and del R. Millan 2013; Galán et al. 2008; Iturrate et al. 2009), prosthetic devices (Müller-Putz et al. 2005; Vidal 1977), and virtual keyboards (Birbaumer et al. 1999; Williamson et al. 2009). BCI sensing technology has also been used for motor recovery training during rehabilitation therapy (Buch et al. 2008; Pfurtscheller et al. 2000).

This work was supported in part by a National Science Foundation (NSF) grant (award 1838815).
Authors' addresses: C. S. Crawford, University of Alabama, 3047 HM Comer, 245 7th Avenue, Tuscaloosa, Alabama 35487, email: crawford@cs.ua.edu; J. E. Gilbert, University of Florida, 432 Newell Dr., Gainesville, Florida 32611, email: juan@ufl.edu.

1946-6226/2019/07-ART39 $15.00
https://doi.org/10.1145/3335815

Neurophysiological technology has also been used for educational research. However, most current research involving emerging neurophysiological technology in classroom settings do not engage students directly with topics related to neuroscience. Instead, neurophysiological data is often collected from students to augment learning experiences (Antle et al. 2015; Huang et al. 2014a; Szafir and Mutlu 2012) or measure performance (Yuan et al. 2014). While this approach is useful, there is a critical gap in the knowledge base regarding approaches that teach students new forms of digital literacies involving neurophysiological technology.

The gap in knowledge at the intersection of physiological computing and computer science (CS) education most likely exists, in part, due to the lack of affordable EEG hardware. However, multiple emerging companies have recently taken interest in manufacturing EEG hardware (Emotiv 2018; Interaxon 2018; Neurosky 2018; OpenBCI 2018), which will likely lead to more affordable EEG devices. Although the accuracy of these sensing devices is not yet fit for critical medical-grade applications, they can be used to design non-critical applications such as interactive games (Chumerin et al. 2013). Based on these insights, we argue that the exploration of these sensing devices in the context of computer science education is timely and feasible.

The concept of coupling computer science education and sensing technologies has been previously investigated. For example, preceding CS education work involved sensors for applications such as robotics (Summet et al. 2009) (e.g., light and proximity sensors), air quality (Fjukstad et al. 2018), mobile applications (Dabney et al. 2013), and wearables (Ngai et al. 2013), However, there is limited research that discusses similar approaches featuring novel neurophysiological sensors (e.g., EEG) capable of capturing information regarding brain activity. The research discussed in this article presents a step toward closing this gap through a visual BCI programming environment designed for novice programmers.

2 BACKGROUND

Several research areas have been integrated with CS education to introduce students to novel concepts relevant to computing. These preceding projects include robotics (McGill 2012), media (Guzdial 2003; Guzdial and Ericson 2009), games (DiSalvo et al. 2011), wearables (Ngai et al. 2013), ecology (Cushing et al. 2007), biology (Dodds et al. 2010, 2012), and law (Sloan et al. 2017).

Although the preceding projects address a wide range of areas, there is limited work toward teaching students concepts associated with BCI technology. Similar to previous CS research, BCI has been coupled with areas such as art (Gurkok and Nijholt 2013), gaming (Chanel et al. 2011), media (Pike et al. 2016), and robotics (Millán et al. 2004). These types of applications are often developed using a BCI software platform.

In the early days of general-purpose BCI systems, Wolpaw classified BCI systems into the following key phases: signal acquisition, feature extraction, feature translation, and commands/applications (Wolpaw et al. 2002). BCI software platforms assist developers with creating systems that involve each of these steps. These platforms can be placed into two categories: method and application focused BCI software platforms.

The main distinction between method and application focused BCI software platforms is the degree to which they aid developers with interfacing brain signals with feedback applications. BCI feedback applications are computer programs that provide real-time visual or auditory feedback to BCI users that correspond to instructions derived from raw brain signals. BCI developers are responsible for constructing these instructions and connecting the output to a feedback application. Application focused BCI software platforms are often designed to optimize this task. However, method focused BCI software platforms usually concentrate on signal processing subtasks applied to signals prior to being passed to feedback applications. For example, method focused BCI software platforms such as BCI2000 (Schalk et al. 2004), Biosig (Guger et al. 2001; Schlogl et al. 2007),

and BCILAB (Kothe and Makeig 2013), were primarily designed to assist BCI experts with optimizing and connecting underlying signal processing and feature extraction methods. In contrast, application focused BCI software platforms, such as Venthur's framework (Venthur and Blankertz 2012; Venthur et al. 2010, 2015) and OpenVibe (Renard et al. 2010) are often designed to make the development of BCI applications easier.

OpenViBE allows users to create complete scenarios using a visual flow-based language. Consequently, prior programming knowledge is not required to create basic BCI applications. OpenViBE currently requires programming skills if users wish to create custom interactive neurofeedback applications. However, research from the block-based programming (BBP) community may assist with eliminating this constraint. The work presented in this article draws inspiration from previous BBP work such as Scratch (Resnick et al. 2009), Alice (Conway et al. 2000), and App Inventor (Pokress and Veiga 2013). Inspiration is also drawn from previous investigations of BPP environments for end-user programmers (Blackwell and Hague 2001; Booth and Stumpf 2013; Krebs et al. 2012; Letondal 2006; Millner and Baafi 2011; Ãngeles Serna et al. 2015). Previous text-based and visual approaches such as python and flow-based have shown promise. However, we argue a visual block-based programming approach may provide novice programmers with opportunities to gain hands-on experience with designing interactive BCI applications.

3 SYSTEM DESIGN

Designers of BCI applications are tasked with creating applications that are influenced by information acquired from a neurophysiological device. Comprehending how to assess and use neurophysiological data acquired from a neurophysiological device is a vital step in the process of creating BCI applications. For example, to create an application that adapts to a user's level of relaxation, a developer must understand how to direct data that reflects a user's affective state into a development environment. Once the data is collected in a development environment, the developer must understand how to use logical structures to create applications that provide meaningful feedback to users based on their emotional state. The ability to achieve these tasks often depends on developers' prior experience using Application Program Interfaces (APIs) or configuring BCI software platforms. Each of these tasks can require quite a bit of technical skill. This dependency on technical skills presents major challenges for novice programmers and non-technical users in general who are interested in getting started with BCI development. We addressed these challenges by designing and implementing a system that combines EEG signal acquisition and block-based programming using modern web technology.

Although BCI software platforms have traditionally focused on providing users tools to manipulate the signal processing components of the BCI pipeline, the presented system focuses on engaging users with the feedback component instead. This approach enables users to design feedback applications that leverage affective state data provided by the underlying BCI pipeline. Consequently, novice users may design interactive applications that leverage the dynamic nature of humans' affective states. This involves three core components: EEG apparatus, EEG data communication, and the web application.

3.1 EEG Apparatus

EEG data must first be captured from a user's brain using an EEG apparatus prior to being provided to a computer. This device measures electrical activity from the brain using sensors. As shown in Figure 1, multiple EEG apparatuses exist. These devices range from medical grade devices that are mainly used for clinical translational research (similar to the g.nautilus shown in Figure 1(F)), to consumer grade wearable devices that are often used for less critical applications. Although consumer grade devices can be less accurate than their medical counterparts, they tend to be more

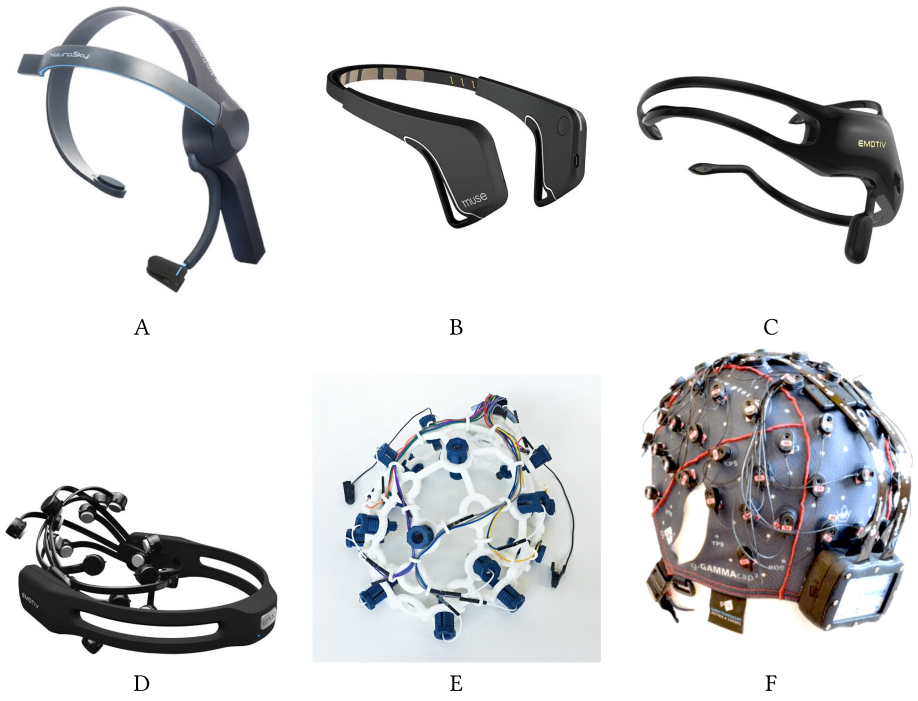
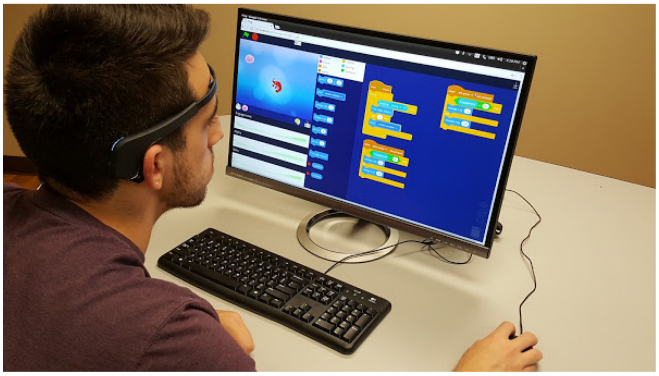


Fig. 1. EEG apparatuses. (A) Neurosky Mindwave, (B) Interaxon Muse, (C) Emotiv Insight, (D) Emotiv Epoch, (E) OpenBCI Ultra Cortex "Mark IV,", and (F) g.Nautilus.

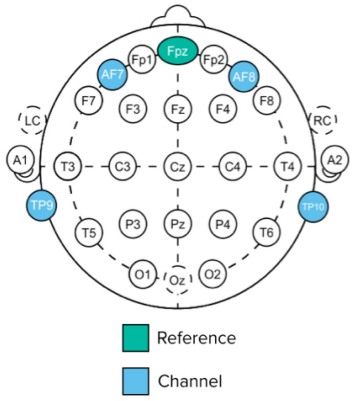
affordable. For example, the devices shown in Figure 1(A)–(E) cost a few hundred dollars. However, more precise devices similar to the one shown in Figure 1(F) can cost thousands of dollars. The recent emergence of affordable BCI devices is a vital step toward making the BCI technology accessible to the general population. Our work aims to leverage this momentum by presenting accessible and easy to use feedback development software for EEG apparatuses.

To communicate the importance of the relationship between BCI software and EEG apparatuses, consider conventional input devices such as a keyboard and mouse. Now imagine users needing to learn an entirely new language to efficiently complete a word processing task. Although this may not be a problem for some users, it would present a clear challenge for others. Fortunately, this is not the state of word processing technology. However, this is a relatively accurate depiction of the current state of BCI software in reference to neurofeedback application development tasks. In an effort to apply this concept to BCI, the Interaxon Muse device shown in Figure 1(B) was used as an EEG-based input modality during the study presented in this article. However, any BCI device capable of communicating with a computer can be integrated.

Brain signals acquired with this device are represented in microvolts (μV), which provide information about the brain's electrical activity. As illustrated in Figure 2(B), this device consists of four channels (TP9, AF7, AF8, and TP10) and one reference (Fpz) based on the international 10-20 electrode positioning system (Pivik et al. 1993). The reference electrode is used as a reference for measurement by the other electrodes. The muse is designed to be mounted on the forehead as shown in Figure 2(A). This allows it to be easily mounted without hair causing significant signal quality issues. In addition, this area is related to measurements of user engagement and attention (Lebedev et al. 2004).



A) User building neurofeedback application



B) Muse electrode positions

Fig. 2. Object management components.

Fig. 3. System design.

Participants used the Interaxon Muse EEG headset to capture attention levels (or relaxation levels), which played a vital role in driving feedback in the neurofeedback applications. Specific details concerning the communication protocol used to send EEG data from the BCI device to the computer are discussed in the following section.

3.2 EEG Data Communication

As shown in Figure 3, EEG data was communicated wirelessly using a Bluetooth 2.0 connection between the EEG apparatus and the computer. Specifically, a Dell 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. The first step in establishing a connection to the BCI device is pairing it to the computer via Bluetooth. This consists of initiating the computer's Bluetooth discovery mode and making the Muse discoverable by holding the Muse's power button down for five seconds. Afterward, the Muse device appears as a nearby Bluetooth device.

Once the EEG device is paired via Bluetooth to the computer, raw EEG signals are acquired on the computer using MuseIO (Interaxon n.d.c), a research client application provided by Interaxon. Information about the EEG device's channel quality is also communicated. Given this work is more focused on the general experience, MuseIO was considered sufficient for the initial prototype presented. Prior to making this design decision, the authors confirmed that EEG data collected from the Muse could be passed to other programs such as OpenViBE if further processing is required in the future. To accomplish this, data is passed from MuseIO to OpenViBE using the Lab Streaming Layer (LSL) (Kothe 2014) communication protocol. Furthermore, the application can support alternative BCI hardware that provides OSC or LSL communication options.

BCI device manufacturers often offer software developer kits (SDKs) for EEG apparatuses. Although these kits offer many tools, they usually do not include ways to directly send EEG data

directly to a web application. Instead, communication protocols such as OSC are commonly used. Although multiple approaches could be used to address this constraint, this work also aims to design a hardware agnostic system architecture. To address this challenge, a server capable of receiving OSC messages was developed. This was accomplished by using the osc.js package (Dzialocha n.d.). OSC's message-based design was leveraged to route data from the EEG apparatus device to appropriate server-side function.

3.2.1 Band Power Session Scores. Band power session scores, computed via MuseIO, were used to capture theta, alpha, and beta frequency band (Interaxon n.d.b) information. MuseIO was used to compute and transport the band power session scores to a server application developed using the node.js JavaScript runtime environment. Each of the utilized frequency bands have been associated with various affective states (e.g., Theta/Deep Meditation (4–8Hz), Alpha/Relaxation (8–13Hz), Beta/Attention (13–30Hz)). A Fast Fourier Transform (FFT) was used to compute the power spectral density of each frequency. FFT calculations were used to derive band power session scores. This approach featured a hamming window of 256 samples (at 220Hz) that was moved 22 samples for each subsequent FFT calculation (0.1s). This method resulted in a 90% overlap between each window.

Band power calculations take into account the sum of the power spectral density between a specific range (e.g., Beta 12–30Hz). Band power measurements are used to generate band power session scores that are scaled between 0 and 1 using a linear function. This linear function returns 0 if the most recent band power value is less than the 10th percentile of the distribution of band powers. The linear function returns 1 if the current value is equal to or higher than the 90th percentile of the distribution of band powers (Interaxon n.d.a). Band power influence on session scores decays with a half life of 10s and indefinitely moves toward 0. This method results in recently observed band powers being more heavily weighted. The band power scores are communicated to the server application at 10Hz using the Open Sound Control (OSC) protocol (Wright 2005). Once the server receives these scores from MuseIO, they are passed to a client web application via WebSockets. Affective state blocks (further discussed in Section 3.3.1) provide users access to the most recent band power session score received from the server.

Band power session scores assist with mapping gradual shifts of affective states to control signals intended for visual objects. Converting values derived from volatile trends to control signals could lead to unintended erratic visual feedback and additional work for novice developers. The steps used to calculate band power session scores do not address spikes in readings that may be caused by eye blinks or muscle movement artifacts. While this is one current limitation of Neuroblock, users new to BCI are often excited to see the spikes corresponding to their eye blinks and movements. In the future, we plan to provide an option to toggle filtering options to introduce digital signal processing concepts.

3.3 Web Application

The web application component was also informed by previous BBP environments such as Scratch (Maloney et al. 2010) and OpenBlocks (Roque 2007). This section discusses how established design principles are supported in our web application. The web interface uses a single-window, multi-pane design to make locating features and navigating the interface easy for users. This approach ensures that core components of the system are always visible. Figures 4 and 5 show the single-window interface which has four panes. The view shown in Figure 4 reflects the interface when a user selects the affective state data viewer. Figure 5 reflects the interface when the sprite viewer is selected. Interface controls are clearly marked using labels and icons related to the controls function. For example, controls associated with the blocks used to develop applications have labels such as motion, sound, data, events, control, sensing, and operators.

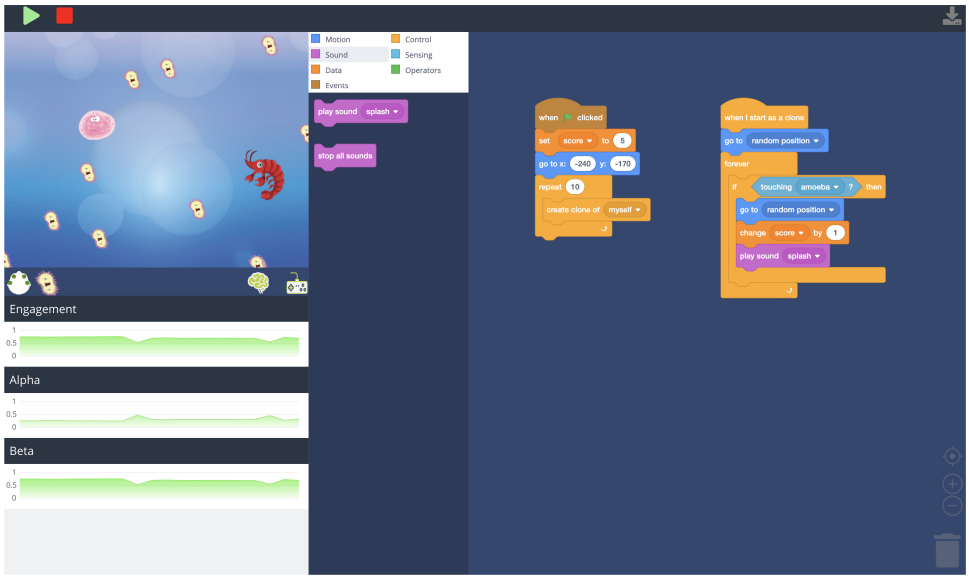


Fig. 4. Web application interface with affective state data viewer selected.

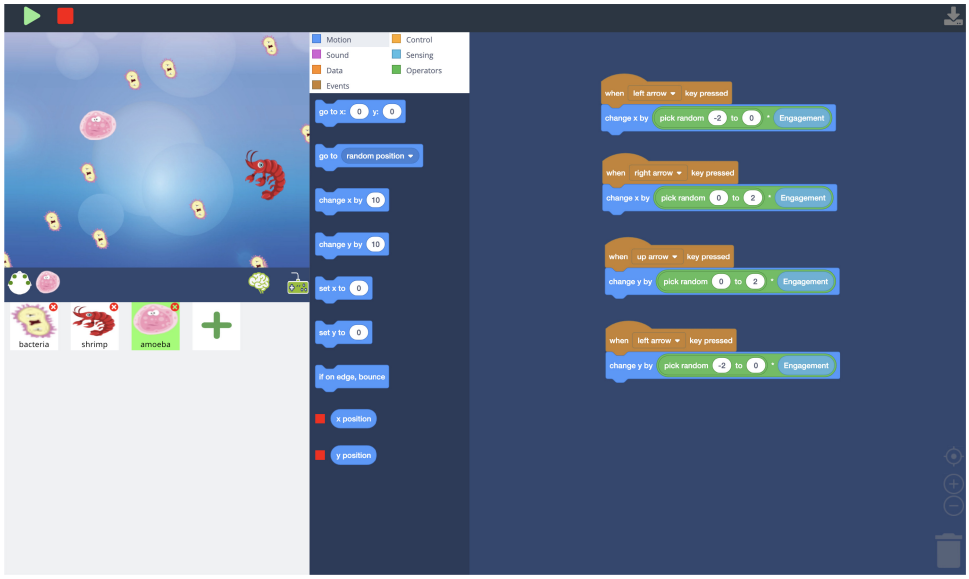


Fig. 5. Web application interface with sprite viewer selected.

Integrating constraints into the neurofeedback application development tool assists with making the development process simple. In many existing BCI software platforms, users can easily cause a system to enter an invalid state mistakenly. For example, a user desiring to develop a neurofeedback application driven by relaxation levels may be allowed to enter the wrong frequency range. This would cause the application to function improperly and may cause a novice developer to become frustrated. Anecdotal observations by the author have shown that people interested in learning BCI often become overwhelmed with the numerous signal processing options available

Table 1. Stage Components

Name	Description
Input Handling	Handles mouse and keyboard input
Scene Manager	Assists with organizing game sprites and settings (ex. background color)
Sprites	2D bitmap images that can be integrated into scenes

Table 2. Block Components

Category	Description
Control	Add control logic such as loops and conditional statements
Data	Create/Modify Variables
Events	Capture events such as a mouse click or keypress. Handles communication between sprites.
Operators	Mathematical operators
Affective State	Blocks reflecting mental state levels (Ex. Engagement), Frequency Band Power (ex. alpha, beta).
Sound	Plays audio feedback
Sensing	Handles functions such as object collision or cursor position

in existing tools. Constraining the available options to specific frequency band ranges related to affective states can prevent users from experiencing this issue. This design principle was utilized as much as possible (without limiting users too much) in an effort to simplify the process of integrating brain activity information into applications. The physical puzzle styled characteristics of the blocks shown in Figures 4 and 5 also provide constraints. For example, if users try to join blocks that cannot be syntactically joined, the two blocks will not physically connect. This prevents users from introducing syntax errors in their programs. This also allows the users to focus more on debugging logic instead of syntax errors. The neurofeedback development tool presented in this work aims to be consistent both internally and across other applications that users may use. For example, all block types (sprite, data, affective state, etc.) have the same operations to achieve goals such as adding, deleting, and modifying blocks. This tool also provides copy and paste shortcut keys for block manipulation that are common with most applications.

The stage component (Table 1), located toward the top left corner of the interface (Figures 4 and 5), contains programmable objects (sprites) that can be designed to respond to a user's affective state. Users can add and remove sprites from this stage area. The stage component was implemented using the open source Scratch-VM library (Lifelong-Kindergarten n.d.). This library assists with maintaining the state of the block-based developed application. Examples of this include managing the location of objects and the current value of variables. This allows users to add logic that can be used to provide instructions to objects featured in the application.

3.3.1 Blocks. The block interface component provides block elements that will be used to create neurofeedback applications. This component draws inspiration from Scratch (Maloney et al. 2010). It features a command palette that is used to switch between block categories. As shown in Table 2, 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 discussed earlier. This library was selected based on its ability to effectively maintain and support various states of a web-based feedback application. To provide neurofeedback functionality, the sensing

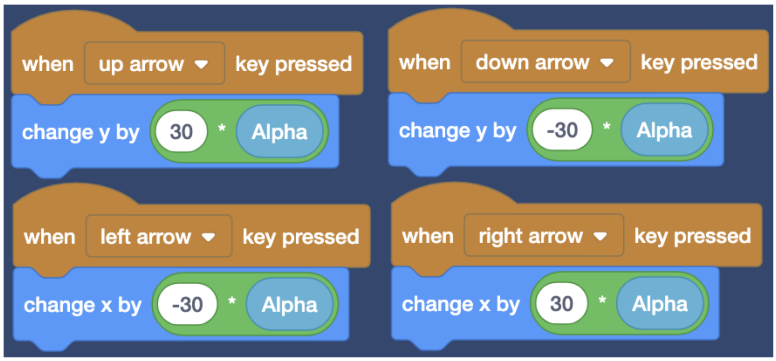
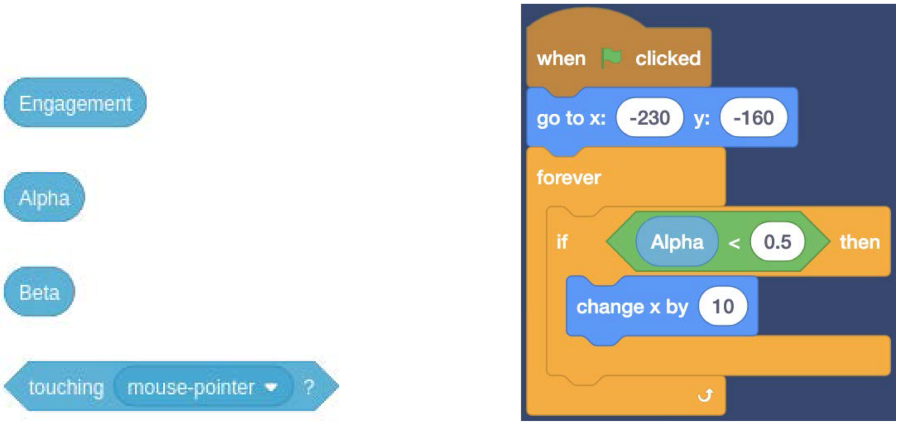


Fig. 6. Basic hybrid-BCI application.



A) Sensing blocks.

B) Simple script that moves an object based on a user's relaxation level.

Fig. 7. Object management components.

category blocks were designed to provide information associated with EEG-based affective state and the mouse events. The affective state blocks include alpha, beta, and engagement blocks. The alpha and beta blocks shown in Figure 7(A) hold values ranging from 0 to 1. These values are calculated using the process discussed in Section 3.2.1. The engagement block holds values that also range from 0 to 1. Calculations for engagement values are informed by a commonly used formula (Pope et al. 1995) that calculates engagement using theta (4–8Hz), alpha (8–13Hz), and beta (13–30Hz) frequency bands (Hassib et al. 2017; Huang et al. 2014b; Szafir and Mutlu 2013; Yan et al. 2016). The engagement index E is calculated as: E = beta / (alpha + theta).

To use these blocks, users move blocks from the block toolbox section to the block workspace by performing a drag-and-drop operation. Afterward, blocks are combined by connecting the puzzle-style blocks together. Figure 6 shows these blocks after the user has assembled them together. In the example shown in Figure 6, the user created instructions that move an object once keys are pressed at a speed relative to the current alpha band power session score. Figure 7(B) also illustrates a simple program. The if block is used in this script along with a greater than operator block to create a condition that checks whether the alpha band power session score is greater than 0.5. The change x by motion block is also used to move the object 10 units whenever this condition is

Fig. 8. Affective state line graphs.

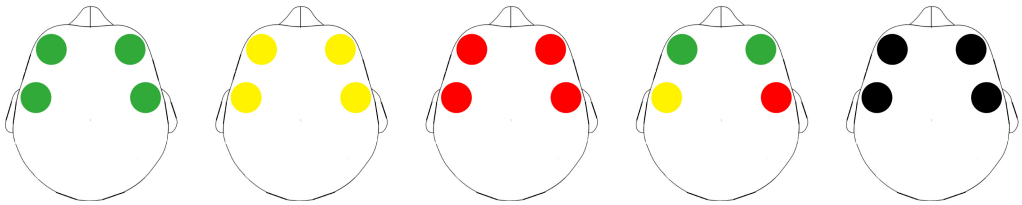
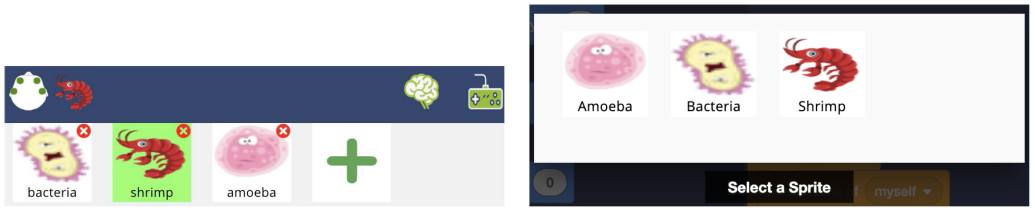


Fig. 9. Channel quality feedback. (A) Good channel quality, (B) OK channel quality, (C) Bad channel quality, (D) Varying channel quality, and (E) No Signal.

true. This example translates to the object moving whenever the BCI device detects mental states related to high relaxation levels.

3.3.2 Neurofeedback. 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. These platforms often include some form of affective state and channel quality feedback. The affective state viewer is a component that provides users feedback about their current affective state. As shown in Figure 8, affective state data are presented as line graphs. These graphs display the recent band power session scores values passed from the server as discussed in the EEG data communication section. This feedback serves as a way to check how affective state levels influence the neurofeedback application. Line graph visualizations are provided for calculations of engagement, alpha, and beta EEG frequency bands.

The channel quality viewer assists users with making sure the BCI device is mounted properly. This primarily consists of ensuring the device has proper contact with a user's forehead. The state of contact is organized into three levels: bad, ok, and good. Each of these levels is presented to users visually as red, yellow, and green indicators, respectively. As shown in Figure 9, these indicators are positioned over a top-down view of a head. This image is used to assist users with mapping sensor indicators shown in the interface to the physical sensors on the EEG apparatus. Channel quality information is passed to the web application using the same pipeline discussed in the EEG data communication section. Figure 9(A) shows an example of when all channels have good signal quality. The channels in Figure 9(B) have moderately good signal quality. The red indicators shown in Figure 9(C) indicate poor signal quality. Channels' signal quality can also



A) Object management pane B) Object selection menu

Fig. 10. Object management components.

vary as shown in Figure 9(D). When no signal is detected, the channel quality feedback is black as shown in Figure 9(E).

Presently, the Scratch-VM library is designed to work as an extension of the Scratch project. Although this is sufficient for developers looking to work on projects that were built using Scratch, it could present challenges for those aiming to implement a platform independent of Scratch. One specific challenge was loading local graphics into the stage environment. This issue was addressed by loading documents from a load directory instead of fetching it from a server hosting scratch projects. Based on observation during the early testing phases, it is natural for users to interact with objects in the stage environment via the mouse. However, once a new object was selected, the workspace originally did not dynamically change.

To address this challenge, input event listener methods offered by the Scratch-VM were leveraged to capture mouse events in the stage area. Once these events were captured, additional Scratch-VM methods were used to change the active script displayed in the workspace. It was also necessary to implement features that assist users with managing objects in the stage area This was addressed by creating a sprite pane area below the stage area as shown in Figure 10(A). Users may add objects by clicking the green plus sign in the sprite pane. Once the green plus sign is clicked, the menu shown in Figure 10(B) presents users with options to add objects to the application. Users may click on an object to add it to the stage area. Objects can also be removed by clicking the red X button in the top right corner of the object icon. Clicking on an object icon in the sprite pane will also change the scripts being displayed in the workspace. When an object is selected, its background is set to green to provide feedback about which object is active.

4 EVALUATION

To explore the BBP approach to neurofeedback application development presented in this work, a user study was conducted. The goal of this study was to obtain feedback from students that assists future designers of novice-friendly neurofeedback development environments. The students recruited for this study (n = 40) were enrolled in an introductory programming course at the University of Florida. The study participants were between the ages of 18 and 30. There was a total of 14 females and 26 males. Each participant had limited experience (approximately 3 weeks) using Java, which was used in the introductory programming course. Participants' majors included computer science, computer engineering, mechanical engineering, electrical engineering, digital arts and sciences, statistics, criminology, and mathematics. Each student was screened to verify that they did not have prior experience developing BCI applications. Two participants were familiar with BBP environments.

4.1 Procedures

The study consisted of three think-aloud sessions over the span of 5 days with 1-day gaps. Each session had a different level of difficulty. Session difficulty was determined based on the number

of sprites and scripts. The complexity of scripts was measured using the McCabe cyclomatic complexity metric which counts the number of decision points (if and if else blocks) (Aivaloglou and Hermans 2016; McCabe 1976). Participants completed each session in order of increasing difficulty. Participants received a total of $75 via a gift card for completing all three sessions. Partial compensation was awarded after each session. Participants received $10 for completing the first session. After completing the second session, participants received $25. Once participants completed the third session, they were awarded $40.

Session 1 began with a pre-session questionnaire that collected general demographic information. Participants watched a 13-minute tutorial video before beginning Session 1. The video explained basic features and basic examples of neurofeedback applications. However, no strategies regarding the best ways to use features were discussed. Participants watched a 2-minute tutorial toward the beginning of Session 2 that covered ways to handle object collisions. A 2-minute tutorial video on methods to create object clones was also shown to participants. Pre-task exercises were provided to subjects during each session.

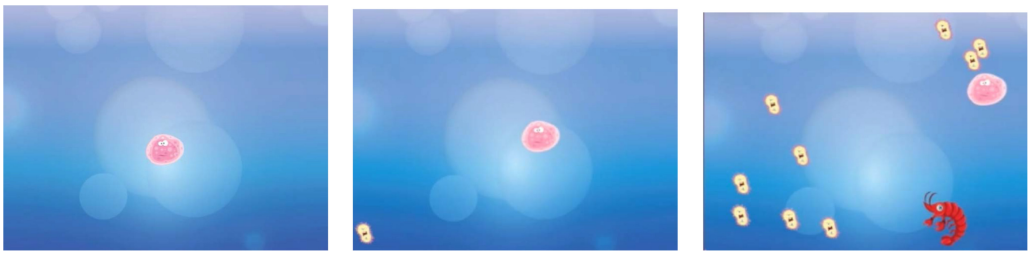
Participants had 20 minutes to complete each pre-task. Prior to the task, participants completed a pre-task exercise. During the pre-task exercise, participants were instructed to build a neurofeedback application. Each pre-task exercise was designed to ensure participants were proficient enough to start 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 each session task. This featured a different application with more instructions and objects.

4.1.1 Interviews. Session tasks were followed by a semi-structured interview to learn more about students experiences. During the interviews, the researcher sat alongside the participant as they both faced a computer running the system. The protocol for the interviews began with questions concerning positive experiences with the system. Afterward, participants addressed questions concerning bothersome experiences. The interview concluded with participants discussing what they would like to change. Audio recordings were generated of each interview for post-experiment analysis. The interviews were used to gain a better understanding of the end-user programming barriers participants encountered during each session. These interviews were analyzed using Grounded Theory (Corbin and Strauss 2008) to identify concepts and major themes related to end-user programming barriers and the general experiences of participants. This approach was also used to generate a list of concepts and categories related to participants' perceptions of a block-based programming approach to neurofeedback application development. The first step of this analysis featured an open coding phase. During this phase, transcripts of participants interviews were analyzed. A basic inductive theory approach was used to organize participant feedback based on common concepts related to pain points participants encountered. The first phase was primarily focused on gathering positive and negative insights into categories that provide a general description of participants perceptions of the implemented system. Afterward, axial coding was used to identify patterns in participants' feedback. A constant comparison approach was used across participants to identify similar and different patterns. Each of the tasks used during the study were informed by the Bacteria Hunt neurofeedback study (Mühl et al. 2010).

4.1.2 Analysis. To gain a preliminary understanding of the system from an end-user programming perspective, a commonly used learning barriers coding scheme was used to code audio recordings of the think-aloud sessions (Ko et al. 2004) (Table 3). The Atlas.ti software was used to code screen recordings captured during each session. Using this approach allowed researchers to analyze participants' verbalizations along with visual information from the interface. For example, think-aloud verbalizations related to barriers often occurred soon after the mouse was idle for

Table 3. Learning Barriers Coding Scheme

Code	Description
Design	User does not know what they want the system to do
Selection	User knows what they want to do but does not know what to use
Coordination	User knows what blocks to use but does not know how to make them work together
Use	User knows what blocks to use but does not know how to use them
Understanding	User thinks they know how to use components together but the system did not do what was expected
Information	User has an idea of why their program did not do what they expected, but does not know how to check



Session 1 Session 2 Session 3

Fig. 11. Screenshot of the stage component during each task.

an extended period. Two researchers coded small sections of the screen recordings and compared their results to reach an agreement.

4.2 Session 1

The goal of the Session 1 pre-task was to create a neurofeedback application featuring an amoeba sprite that moves upward as relaxation levels (alpha band power session scores) increased. Participants completed the task once they built an application that moved the amoeba to the top of the stage based on the users relaxation level. During the Session 1 task, participants were asked to create a hybrid neurofeedback application featuring an amoeba sprite as shown in Figure 11. This consisted of designing an application that leverages both the keyboard and the EEG apparatus to control the amoeba object. Participants were instructed to design an application that boosts the speed of the amoeba sprite when high levels of engagement are detected. They were also asked to reduce the speed when high levels of relaxation (alpha band power session scores) were detected. Additional instructions included adding jittery motions to the amoeba when high levels of relaxation or low levels of engagement were detected. This program featured one sprite, three scripts, and one decision point. The goal of the application was to move the amoeba to the left side of the stage using the keyboard. Furthermore, participants were asked to positively map the amoebas speed to engagement levels provided by the EEG apparatus. They were also instructed to add collision detection functionality for the amoeba and shrimp. The instructions provided also mentioned that the amoeba should move to a random position whenever two objects collided.

4.2.1 Learning Barriers. To gain a better understanding of students' experiences we evaluated learning barriers participants encountered while completing the neurofeedback development task.

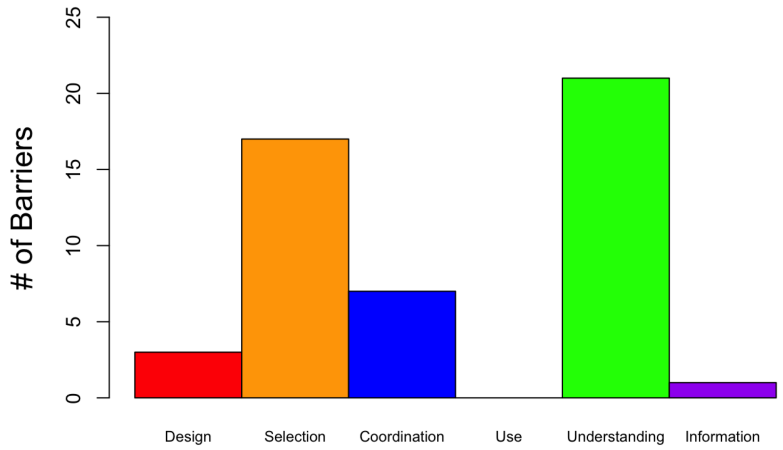


Fig. 12. Screenshot of the stage component during Session 1 task.

As shown in Figure 12, understanding barriers were the most common barriers during Session 1. The number reported reflects the total number of barriers that occurred for all participants. Understanding barriers were responsible for 42% (21) of all Session 1 barriers. Selection barriers also occurred frequently. Selection barriers were accountable for 34% (17) of all Session 1 barriers. The third most frequent barriers were coordination barriers. These barriers were liable for approximately 14% (7) of Session 1 barriers. The least frequent barriers during Session 1 were use and information barriers. These results suggest that when participants are initially introduced to the system, they have issues evaluating the program's behavior. Understanding barriers were often related to user-generated code errors that caused the participants' program to exhibit unexpected behavior. For example, after an order of operation error one participant stated:

> "Nothing is happening and I am pressing the arrows … I am thinking I did the program wrong.'

In other cases, understanding barriers were related to participants misunderstanding how to use affective state feedback to debug motion commands that are linked to affective state information acquired from the BCI device. For instance, one participant did not realize low levels of engagement were stopping the amoeba from moving and stated:

> "I think I followed the instructions, but I am not sure why it is not doing anything."

The results also suggest that some users had issues selecting blocks related to more abstract concepts relative to basic block operations. This often occurred when participants were attempting to add a random function to their program. The web application has a random block that returns a random number between two user-defined integers. For example, while trying to accomplish this, one participant stated:

> "Do I type random? … It's probably here somewhere … Where are you random?"

4.2.2 Interviews. Participants responses during post-session interviews were used to better understand the end-user programming barriers observed during the screen recording analysis. The Session 1 interviews gave insight into participants first impressions of the system. It also provided details on how the participants initially perceived BCI. Five categories (Table 4) emerged from the Session 1 interviews: self-regulation, interacting with blocks, ease of use, affective-based

Table 4. Session 1 Selected Responses Related to the Five Main Categories

Category	Participant quote
Self-Regulation	"Being able to see alpha and beta signals and engagement signals … I guess it was kind of difficult to try to manipulate them myself."
Interacting with blocks	"Honestly, it was, sometimes I would try to move a block, and I'd move an individual part within the block."
Ease of Use	"The system is very easy to use. The program, it's very, very straightforward."
Affective-Based Control and Motion	"I thought it was really cool how you could use your brain to make it move."
Visual Appeal of Affective feedback	"It was cool seeing the level of brainwaves and stuff interact with the computer."

control and motion, and visual appeal of affective feedback. Neurofeedback applications often require users to voluntarily manipulate their mental state to achieve a goal. This process can also be defined as the voluntary self-regulation of signals from the central nervous system (Kothe and Makeig 2013). Many participants identified self-regulation as challenging during the Session 1 interviews. For example, one participant stated:

> "I guess it was kind of difficult to try to manipulate them myself. I would think I am relaxed but all of a sudden the signals would be all over the place.. So umm it was amazing to see but also kind of frustrating."

Although participants were familiar with basic concepts related to programming logic, none of the participants had experience using BCI devices. Consequently, participants often commented on the novelty of self-regulation during Session 1. One participant stated:

> "Yeah, I thought it was pretty cool … In your everyday life, you're not really forced to channel your relaxation like that in a way to accomplish a goal … It was a challenge because you had to suppress anxiety while also staying relaxed so you that you could increase the score and eventually win the game. So I thought that the challenge was pretty enjoyable."

Most of the interview questions were focused on identifying drawbacks of the system. However, participants often focused on self-regulation. When asked about aspects of the system that were bothersome, one participant responded:

> "I'd say not really the system, it's more like trying to control your brain, I guess. If it's doing something, you can't really control your level of engagement, I mean consciously. But other than that, there wasn't anything else bothersome or anything."

No specific instructions were given to influence how participants approached self-regulation. However, many participants attempted to develop various strategies to assist with their self-regulation goals. One participant mentioned:

> "When I was trying to move it up, I would look at the words. I'm like, Okay, how do you spell this? I try to concentrate or, Why is the amoeba going down whenever I'm trying to move it up? So I try to think of questions and I try to answer them."

Although self-regulation may not directly impact the kinds of end-user programming barriers that novice programmers face, it is important to investigate how it may influence novice users interacting with a neurofeedback application development platform. Leveraging self-regulation with an interactive and user-friendly environment may engage new audiences with BCI. This approach could also support recent research on ways to improve the self-regulation of children who have suffered from multiple traumas (Antle et al. 2015).

Many of the responses provided by participants were related to their interactions with the block components contained in the system. The responses can be separated into two categories: Search and Utilization. Since Session 1 was the first time many participants interacted with a block-based language, many participants faced challenges while trying to find blocks. Participants mentioned these challenges even after watching the tutorial video and completing the pre-test. One participant stated:

> "I couldn't remember where everything was like under the … I don't remember what its call … the control panel like with all the motion and events but I mean I can just click through it and eventually I would remember where everything was."

A second participant reiterated this point stating:

> "The instructions would ask you to use blocks that you didn't know for sure that were there, but it was nice, you knew that they were gonna be there, kind of thing."

Although the participants stated they had issues recalling where blocks were, they seemed comfortable addressing this issue by browsing through the block categories. This observation is also supported by previous studies investigating visual programming environments (Weintrop and Wilensky 2015). Participants that seemed more comfortable with text-based languages also expressed frustrations with searching for blocks. Many of the responses were related to text-based entry. For example, one participant stated:

> "Having your operators on a different page, instead of being able to just enter them on in the keyboard, so you have to navigate then to a different area then to get that."

This response is also supported by previous research investigating hybrid approaches to visual programming (Koitz and Slany 2014). In general, participants expressed that they needed more practice before being able to efficiently locate all of the blocks. One participant particularly stated:

> "Just that I would have to just gain familiarity with the various [blocks] … where all the things are located, the categories like the variables, the motion, the operators and everything like that. I just needed to get the feel of where everything was. But besides that, it was very intuitive."

Responses related to block usage in the workspace area were also frequently shared. In this context, block utilization focuses on the process of manipulating blocks in the workspace. Participants often mentioned how visual features of the blocks such as color and shape assisted them with creating their applications. For example, one participant shared:

> "While using it umm it actually helps even if you don't know what you doing and the way the blocks are shaped I guess in a way you know which goes where. Its a block in a diamond shape you know its a Boolean so you know if you use that .. Color coding helps to … it helps you distinguish and even if you don't know what you're doing you know what you're doing."

Another participant seemed to be fond of their ability to modify the program once they realized they had made a mistake.

> "Because I ran into something when I was doing the task, and I had forgotten to put a forever block, and I was going through all the stuff, and I was like, Oh, that was the one thing that I had forgotten. And I just had to move one thing, and it was that simple to fix."

Not everyone reported the same experience related to using the blocks. A few participants reported having issues connecting blocks together. This often occurred when participants were building mathematical formulas. One participant stated:

> "Sometimes when I was trying to put the multiplication thing when I was writing the program, there was a couple times where it went to where I didn't want it to."

Although the block-based design supported novice programmers, participants mentioned drawbacks that align with findings found in previous literature. Numerous participants commented that the system was easy to use. While discussing what contributed to the ease of use participants mentioned a variety of features. Participants often compared the Session 1 task to their experiences using Java. One participant stated:

> "I don't know, it was visually appealing, and I'm used to programming just blank, kind of boring pages of code. But this made it look like it's easier to use, and it's a little bit more simple and fun, I guess I'd say."

A second participant stated:

> "It was pretty easy, it is cool. I don't know, it was a lot easier than programming'cause all the statements were kind of the same as programming but a lot simpler. So that was cool, that was nice."

One observation was that most participants automatically compared the system to text-based languages without being asked to. When this occurred, participants mostly mentioned the system was easy to use even with the issues discussed in the interacting with blocks section. Participants also shared their feelings about working with the Muse headband. One participant mentioned:

> "I think it was pretty easy. I'd say not like when you think of a BCI, you think of a big helmet with a bunch of crazy wires, but it was kind of just like a headband or something, which was pretty easy, and easy to turn it on by a button on the side, and you don't really need a big set up."

Statements related to participants sentiments toward affective-based control and motion were frequently shared. Specifically, affective-based control responses focused on participants' perceptions of being able to influence objects using voluntary self-regulation. Responses related to affective-based motion, on the other hand, focused on how participants felt about the motion caused by self-regulation. These two types of responses were coded to better understand how participants perceived the input (BCI control) and output (visual feedback, motion) components of the BCI system. When asked about what they enjoyed the most about the Session 1 task one participant stated:

> "Well I never used anything like it before so I thought it was pretty interesting to use like a device like that and control you know something on the screen with just your brain."

A second participant stated:

> "It was pretty cool having to focus, trying to move it up and stuff, and it was also interesting seeing what I had to think of in order to get it to move up."

Most of these responses were caused by the novelty of using a BCI device. Participants often used words such as cool, interesting, and fun to describe their first interactions with the system. Another participant shared:

> "I just thought it was super cool that I could make it move based on how concentrated I was, that was a really neat experience. It made myself feel more involved in it, the amoeba would go up. That was pretty neat. ."

A different participant simply stated:

> " I thought it was really cool how you could use your brain to make it move."

Participants also reported enjoying the ability to use affective state blocks to drive the program. For example, one participant mentioned:

> "Just having the alpha and beta waves inputted into the code, and be able to use those values in moving the amoeba . . . "

Along with affective-based control, participants were also interested in the affective state feedback provided in the system. Many comments were related to the experience of seeing visualization reflecting their EEG data for the first time. When asked about the task one participant mentioned:

> "It was really cool seeing the levels of my brain as it was recording it, it was pretty cool seeing the levels of it. Trying to think, make it different by thinking, it's pretty cool."

One participant even mentioned additional ways he would like to use affective feedback stating:

> "It was pretty cool having to monitor how concentrated I was'cause I think that stuff is really cool, getting to see what I'm thinking of if I'm focused or not. That would probably be useful probably when I'm studying for a test or something."

Another participant shared:

> "I thought it was really cool just seeing the levels like that, seeing them change. How you were talking, I could see the differences in real time. That was pretty cool."

One interesting observation is that participants seemed to be solely intrigued by the novelty of seeing a visualization mapped to their mental state. None of the participants commented on visual features specific to the design of the waveforms such as color or shape. As discussed in the following section, this interest would shift during the following sessions.

4.3 Session 2

During the Session 2 task, participants created a hybrid neurofeedback application that featured an amoeba and bacteria object as shown in Figure 11. This also consisted of supporting interactions via the keyboard and EEG apparatus to control the amoeba sprite. Participants were instructed to develop an application that detected a collision between the amoeba and bacteria sprite. Instructions to increase a score variable each time a collision occurred were also provided. They were also instructed to move the objects back to their starting positions when the collision occurred.

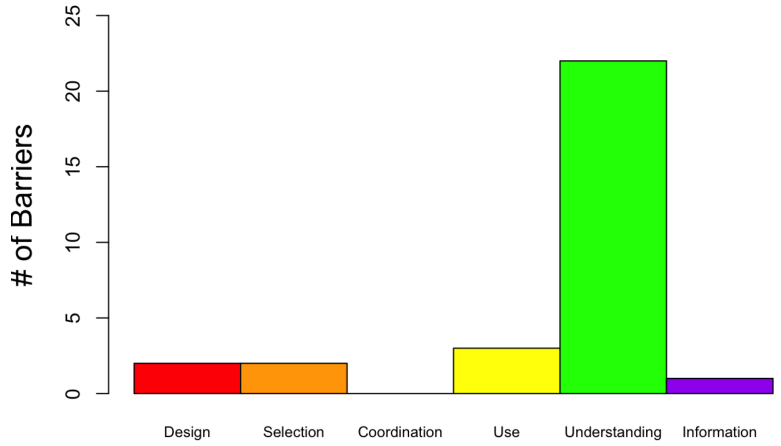


Fig. 13. Screenshot of the stage component during Session 1 task.

Table 5. Session 2 Selected Responses Related to the Five Main Categories

Category	Participant quote
Mounting the BCI Device	"The BCI headset somewhat took a little bit of adjustment to work out enough to the point where I'd rather not take it off than to readjust it."
Familiarity	"This time also it was easier compared to the one before it because initially I wasn't familiar with the system."
User-Defined Factors and Thresholds	"Since alpha is only between zero and one, I'm not gonna do a huge number 'cause then it does nothing."

Participants were asked to design the application so that speed was positively influenced when the BCI device detected high relaxation levels. Additional instructions included moving the bacteria toward the center of the stage when high levels of relaxation were detected and positively mapping beta band power session scores to jitter motion. Beta EEG frequency bands have been associated with attention and alertness (Cho et al. 2002). This program featured two sprites, six scripts, and three decision points.

4.3.1 Learning Barriers. Session 2 barriers. Similar to Session 1, understanding barriers were the most frequent barrier during the second session (Figure 13). These barriers made up 73% (22) of all Session 2 barriers. Coordination and information barriers were the least frequently encountered barriers during Session 2. The re-occurrence of understanding barriers in Session 2 suggest that issues with unexpected program behavior persisted. During Session 2, these barriers were often related to participants forgetting to click the green flag to start testing the program. Prior to realizing this, one participant stated:

> "If the alpha is greater than 0.5 which right now it is it should move [user changes threshold values] … It is not moving at all."

4.3.2 Interviews. Three new categories (Table 5) emerged from the Session 3 interviews: mounting the BCI device, familiarity, and user-defined factors and thresholds. The Session 2 task was a bit more involved as previously discussed. In addition, the novelty of the BCI aspect of the system began to have less of an impact. Instead, participants seemed to focus more on the

functional aspects of the system. One example of this was participants beginning to notice the sensor quality feedback provided in the interface. Based on notes collected while observing participants and the analysis of the screen recordings, participants often disregarded sensor quality feedback and instead focused on how cool the affective state visualizations and object motion were during Session 1. During Session 2, participants seemed to begin viewing the system more as a tool than a cool toy. The goals of the task were slightly more difficult which resulted in the need for more self-regulation while testing the neurofeedback applications. Participants often checked the sensor quality feedback while attempting to debug their applications. Consequently, participants expressed concerns with mounting the BCI device. For example, one participant stated:

> "I thought that the BCI device is sometimes fickle, and it's kind of hard to mount. And I couldn't get the sensors to be green quite all the time."

Participants also commented on how this issue influenced their effectiveness. One participant shared the following in reference to time consumption:

> "Well, I never really knew because the signals would change a lot of times and show up green, or go up black, and then show up yellow. I never really knew if it was completely positioned correctly. That took a while each time for me to try to position it."

A common trend was participants blaming themselves for the issue and not the BCI device. One participant stated:

> "So that was a little annoying because I wasn't sure if it was a fault on my part, if it was the BCI just not working correctly, picking up signals. I always just assumed it was on my part, I just would always try to reposition it."

These self-blaming type of responses were more common with females, which supports observations reported in previous BCI literature (Hjelm and Browall 2000). Although the BCI hardware is beyond the scope of this work, it is important that this observation supports our goal of providing participants feedback about the current state of the BCI apparatus. Otherwise, students may run into invisible issues related to hardware that may hinder the overall experience of novice users building neurofeedback applications.

Along with paying closer attention to the sensor quality feedback, participants were also more familiar with the interface. As discussed previously, participants expressed issues interacting with blocks in the interface during Session 1. During the Session 2 interviews, participants began to express sentiments related to familiarity. One participant stated:

> "I think it's just because I've done it before, like last time. And last time I forgot where some of the stuff were, so usually I spend a lot of time trying to figure it out what does what, and where to find the'If Bounce' thing. But now I knew exactly where it was, so it made it quicker this time."

A second participant stated:

> "Well, I was already familiar with the system. So the second I sat down at the computer, I already knew what I was gonna do."

Other statements seemed to reflect participants trust in the system. For example, one participant shared:

> "Especially now that I have more experience with it, and kind of understanding exactly what each level will do to it, and kind of what to expect from it."

In general, participants familiarity with the system during Session 2 seems to suggest core features were easy to learn. This is also supported by the drop in observed selection barriers during Session 2.

One of the Session 2 subtasks required participants to define factors and thresholds related to relaxation and attention. Factors were used to manage the intensity of an affective state influence. Factors and threshold values greatly influence how shifts in the participants affective state effects objects' motion. While completing the neurofeedback application, participants often adjusted these factors until a desired effect was acquired. During the Session 2 interviews, participants often shared this process. One participant stated:

> "It wanted me to select a value that was high, and it didn't say what specifically is high or low. So I originally put it, I was thinking like maybe 0.75 something like that, but I decided ultimately to go with 0.6. And I left it as that, and when I went to try the program, I found it was kind of… I mean it wasn't easy to get it to 0.6, like I set it to. So I decided to change it to 0.5, and that seemed to work.."

A second participant stated:

> "I liked the freedom that I got with it. When it was high, I can make it move however fast I wanted, and I can determine what was high and what was low for all the levels of activity."

The ability to design the influence seemed to engage the participants. This observation supports the goal of allowing users to create custom feedback applications that leverage data provided by the BCI device. Students also shared their experiences controlling the direction of the objects using factors. For example, one participant stated:

> "It was fun figuring out how to, making it move to the left, I think I pretty much figured out, there was more than one way to approach it. How I did it, since it's the alpha [that] was changing the direction, I just made it negative so it would go to the left."

One important point shown in the response above is the connection the participant made with the affective state data. Although the nature of data provided by the BCI device is different than traditional input modalities such as mice and keyboards, the system allowed the participant to use the BCI commands to generate similar output. The participants also did not express much trouble figuring out how to create custom factors and thresholds to produce the desired effect. Students also shared how they used the objects motion as a way to confirm the impact of factors and thresholds. One participant stated:

> "Once I changed it and I could visually see them moving in larger directions, I guess I, I don't wanna say it made me happier, but, it made me feel like I was actually on the right track."

Although user-defined factors and thresholds are basic components of neurofeedback applications, it is fundamental to getting started with closed-loop control systems that leverage information about the users' affective state. According to the Session 2 interviews, the system supported participants with accomplishing tasks involving these components.

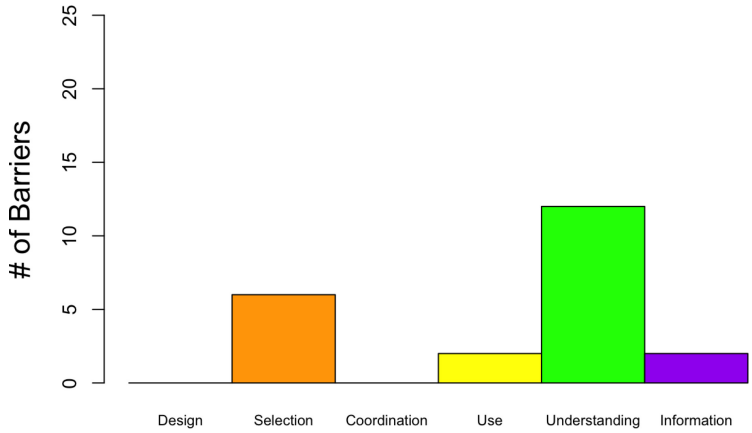


Fig. 14. Screenshot of the stage component during Session 1 task.

4.4 Session 3

During the Session 3 pre-task, participants created a hybrid neurofeedback application featuring an amoeba and multiple bacteria objects. Participants were asked to design the application so that it supported control from both the BCI device and keyboard. The goal of the application was to use the keyboard to move the amoeba to the right and left sides of the stage to catch falling bacteria. Participants were instructed to design the application so that amoeba moved faster during high levels of relaxation. They were also instructed to make the bacteria fall faster during high levels of relaxation. Participants were asked to create a score variable that increased when collisions between the bacteria and amoeba occurred. During the session three task, participants created a hybrid neurofeedback application featuring an amoeba, multiple bacteria characters, and multiple shrimp objects as shown in Figure 11. The objective of the neurofeedback application was to eat as many bacteria objects as possible while avoiding the shrimp objects. To create this application, participants were asked to design features that supported control via the EEG apparatus and keyboard. Participants were instructed to positively map the speed of the amoeba to engagement levels. They were also instructed to create duplicates of the shrimp objects when high levels of relaxation were detected. The instruction also asked the participants to make the shrimp clones disappear when high beta band power session scores were sustained. Other instructions included reducing points when the amoeba collided with a shrimp sprite, returning the amoeba to the center of the stage when all points were lost, and increasing points when the amoeba collided with bacteria. This program featured three sprites, nine scripts, and five decision points.

4.4.1 Learning Barriers. As shown in Figure 14, understanding barriers were also the most frequently encountered barriers during Session 3. These barriers were responsible for 54% (12) of barriers during Session 3. Following understanding barriers were selection barriers, which accounted for 27% (6) of Session 3 barriers. Coordination and information barriers were the least common barriers during Session 3. Similar to Sessions 1 and 2, the understanding barriers observed in Session 3 were related to participants observing unexpected behavior related to objects movements. After using an incorrect motion block one participant stated:

> "For some reason it won't go pass the center point of the stage."

Although the participant needed blocks in the motion category, they had an issue finding other required blocks, which were in different categories. The reemergence of selection barriers

was related to issues finding blocks that caused the amoeba to exhibit a jittery motion. While attempting to implement this feature, one participant stated:

> "It has to be in the motion category but for some reason I can't remember any of these being a thing."

However, these instances were rare.

4.4.2 Interviews. Many concepts shared during the previous two sessions did not appear during Session 3. However, mounting issues were frequently shared during this session. One participant stated:

> "I thought sometimes, taking on and off the BCI device, sometimes it was difficult to get the sensors adjusted just right, or have to wait a couple seconds for it to connect to the computer."

A second participant stated.

> "So I was trying to focus but it didn't really kind of work. So I didn't know if it was my part or the device."

As mentioned previously, the hardware component is beyond the scope of this work but should be investigated further as researchers investigate ways to extend BCI to the general public.

5 DISCUSSION AND CONCLUSION

The goal of this work was to address the lack of novice-friendly (BCI) application development tools by investigating a block-based programming BCI approach. Specifically, this work focused on the design, implementation, and evaluation of a system that allows novice programmers to build basic neurofeedback applications. In total, the 40 study participants ran into 101 barriers across three sessions. Participants encountered an average of 3 barriers during the study. The total number of barriers experienced across all sessions ranged from 0 to 11 with a standard deviation of 1.96. The most common barriers across all sessions were understanding barriers. These barriers made up 54% of all errors experienced throughout the entire study. Selection barriers were the second most common barriers throughout the entire study. Selections barriers were responsible for 24% of all barriers encountered throughout the study.

Some participants expressed difficulties with self-regulation during user interviews. Insights gained from Session 1 suggest that providing sources of artificial EEG input for testing purposes may address challenges associated with self-regulation. It is not common for BCI development platforms to provide out-of-the-box features for generating synthetic EEG-based affective state data. Consequently, our first step toward exploring an educational BCI development platform tasked students with both programming and attempting to control their EEG while testing their programs. However, developers of future educational physiological computing platforms should strongly consider integrating an artificial EEG source. This approach will also assist future researchers to avoid confounding factors associated with evaluating students self-regulation and programming abilities simultaneously. It is still, however, important to provide novices experiences with live EEG data as the classic definition of a BCI involves an interaction between the central nervous system and its external or internal environment (Wolpaw and Wolpaw 2012).

Although students encountered barriers, our results suggest that students were excited to design BCI applications. During each session, students were able to apply basic concepts associated with testing and designing neurofeedback applications. One interesting observation from this study was a potential shift in how students viewed the application. The novelty of working with a BCI

system for the first time was apparent from insights gained from the user interviews. However, many students began to treat the application more like a tool to accomplish assigned tasks during later sessions as opposed to a game. Mounting issues caused by the EEG hardware may have contributed to this shift in perception. Further work is needed to better understand how novice programmers are influenced by hardware and software BCI components.

Students who seemed more experienced with programming often asked for text-based alternatives to access various blocks. Additionally, students were limited to using predefined EEG processing methods. Hybrid approaches that leverage text, flow (e.g., Simulink), and block-based programming may provide a higher ceiling for a wide range of students.

This work presented an approach that tasked novice programmers with designing passive BCI applications and testing the designs by collecting EEG data from themselves. Feedback from user interviews suggests that this approach may encourage self-regulation behaviors during programming exercises. However, this approach may also influence how students learn core computing concepts. The scope of this work focused on exploring the feasibility of designing a BCI application development environment for novice programmers. However, further research involving self-regulation practices and fundamental CS concepts could provide interesting ways to leverage novel physiological sensing technologies in future learning environments.

ACKNOWLEDGMENTS

The authors thank Chris Hundhausen and the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

Efthimia Aivaloglou and Felienne Hermans. 2016. How kids code and how we know: An exploratory study on the scratch repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, 53–61.

Alissa N. Antle, Leslie Chesick, Aaron Levisohn, Srilekha Kirshnamachari Sridharan, and Perry Tan. 2015. Using neurofeedback to teach self-regulation to children living in poverty. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 119–128.

M. Ángeles Serna, Cormac J. Sreenan, and Szymon Fedor. 2015. A visual programming framework for wireless sensor networks in smart home applications. In *Proceedings of the 2015 IEEE 10th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP'15)*. IEEE, 1–6.

Hans Berger. 1929. Über das elektrenkephalogramm des menschen. *Archiv für Psychiatrie Und Nervenkrankheiten* 87, 1 (1929), 527–570.

Niels Birbaumer, Nimr Ghanayim, Thilo Hinterberger, Iver Iversen, Boris Kotchoubey, Andrea Küler, Juri Perelmouter, Edward Taub, and Herta Flor. 1999. A spelling device for the paralysed. *Nature* 398, 6725 (1999), 297–298.

Alan F. Blackwell and Rob Hague. 2001. AutoHAN: An architecture for programming the home. In *Proceedings IEEE Symposia on Human-Centric Computing Languages and Environments, 2001*. IEEE, 150–157.

Tracey Booth and Simone Stumpf. 2013. End-user experiences of visual and textual programming environments for Arduino. In *International Symposium on End User Development*. Springer, 25–39.

E. Buch, C. Weber, L. G. Cohen, C. Braun, M. A. Dimyan, T. Ard, J. Mellinger, A. Caria, S. Soekadar, A. Fourkas, and N. Birbaumer. 2008. Think to move: A neuromagnetic brain-computer interface (BCI) system for chronic stroke. *Stroke* 39, 3 (Mar. 2008), 910–917. DOI: https://doi.org/10.1161/STROKEAHA.107.505313 [doi] LR: 20161124; GR: Intramural NIH HHS/United States; JID: 0235266; ppublish.

Tom Carlson and Jose del R. Millan. 2013. Brain-controlled wheelchairs: A robotic architecture. *IEEE Robotics & Automation Magazine* 20, 1 (2013), 65–73.

Guillaume Chanel, Cyril Rebetez, Mireille Bétrancourt, and Thierry Pun. 2011. Emotion assessment from physiological signals for adaptation of game difficulty. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41, 6 (2011), 1052–1063.

Baek Hwan Cho, Jong-Min Lee, J. H. Ku, Dong Pyo Jang, J. S. Kim, In-Young Kim, Jang-Han Lee, and Sun I. Kim. 2002. Attention enhancement system using virtual reality and EEG biofeedback. In *Proceedings IEEE of Virtual Reality, 2002*. IEEE, 156–163.

Nikolay Chumerin, Nikolay V. Manyakov, Marijn van Vliet, Arne Robben, Adrien Combaz, and Marc M. Van Hulle. 2013. Steady-state visual evoked potential-based computer gaming on a consumer-grade EEG device. *IEEE Transactions on Computational Intelligence and Ai in Games* 5, 2 (2013), 100–110.

Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, and Kevin Christiansen. 2000. Alice: Lessons learned from building a 3D system for novices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 486–493.

Juliet Corbin and Anselm Strauss. 2008. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (3rd ed.). Sage Publications, Inc, Thousand Oaks, CA, US.

Judith Bayard Cushing, Richard Weiss, and Yoshiya Moritani. 2007. CS0++ broadening computer science at the entry level: Interdisciplinary science and computer science. *Journal of Computing Sciences in Colleges* 23, 2 (2007), 51–57.

Matthew H. Dabney, Brian C. Dean, and Tom Rogers. 2013. No sensor left behind: Enriching computing education with mobile devices. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*. ACM, 627–632.

Zachary Dodds, Ran Libeskind-Hadas, and Eliot Bush. 2010. When CS 1 is biology 1: Crossdisciplinary collaboration as CS context. In *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education*. ACM, 219–223.

Zachary Dodds, Ran Libeskind-Hadas, and Eliot Bush. 2012. Bio1 as CS1: Evaluating a crossdisciplinary CS context. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*. ACM, 268–272.

Andreas Dzialocha. [n.d.]. osc-js. Retrieved on April 1, 2017 from https://www.npmjs.com/package/osc-js.

Emotiv. 2018. Retrieved on May 17, 2018 from https://www.emotiv.com/.

Bjørn Fjukstad, Nina Angelvik, Maria Wulff Hauglann, Joachim Sveia Knutsen, Morten Grønnesby, Hedinn Gunhildrud, and Lars Ailo Bongo. 2018. Low-cost programmable air quality sensor kits in science education. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 227–232.

Ferran Galán, Marnix Nuttin, Eileen Lew, Pierre W. Ferrez, Gerolf Vanacker, Johan Philips, and J. del R. Millán. 2008. A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots. *Clinical Neurophysiology* 119, 9 (2008), 2159–2169.

Christoph Guger, Alois Schlogl, Christa Neuper, Dirk Walterspacher, Thomas Strein, and Gert Pfurtscheller. 2001. Rapid prototyping of an EEG-based brain-computer interface (BCI). *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 9, 1 (2001), 49–58.

Hayrettin Gürkök and Anton Nijholt. 2013. Affective brain-computer interfaces for arts. In *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII'13)* IEEE, 827–831.

Mark Guzdial. 2003. A media computation course for non-majors. In *ACM SIGCSE Bulletin*, Vol. 35. ACM, 104–108.

Mark J. Guzdial and Barbara Ericson. 2009. *Introduction to Computing and Programming in Python, a Multimedia Approach*. Prentice Hall Press.

Mariam Hassib, Stefan Schneegass, Philipp Eiglsperger, Niels Henze, Albrecht Schmidt, and Florian Alt. 2017. EngageMeter: A system for implicit audience engagement sensing using electroencephalography. In *Proceedings of the 2017 Chi Conference on Human Factors in Computing Systems*. ACM, 5114–5119.

Sara Ilstedt Hjelm and Carolina Browall. 2000. Brainball-using brain activity for cool competition. In *Proceedings of NordiCHI*, Vol. 7.

Jin Huang, Chun Yu, Yuntao Wang, Yuhang Zhao, Siqi Liu, Chou Mo, Jie Liu, Lie Zhang, and Yuanchun Shi. 2014a. FOCUS: Enhancing children's engagement in reading by using contextual BCI training sessions. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1905–1908.

Jin Huang, Chun Yu, Yuntao Wang, Yuhang Zhao, Siqi Liu, Chou Mo, Jie Liu, Lie Zhang, and Yuanchun Shi. 2014b. FOCUS: Enhancing children's engagement in reading by using contextual BCI training sessions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1905–1908.

Interaxon. [n.d.]a. Available Data - Muse Developers. Retrieved on April 1, 2017 from http://developer.choosemuse.com/research-tools/available-data.

Interaxon. [n.d.]b. Band Power Session Scores. Retrieved on April 14, 2019 from http://developer.choosemuse.com/tools/available-data#Band_Power_Session_Scores.

Interaxon. [n.d.]c. MuseIO. Retrieved on August 1, 2018 from http://developer.choosemuse.com/tools/museio.

Interaxon. 2018. Retrieved on April 14, 2019 from http://www.choosemuse.com/.

Iñaki Iturrate, Javier M. Antelis, Andrea Kübler, and Javier Minguez. 2009. A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation. *IEEE Transactions on Robotics* 25, 3 (2009), 614–627.

Betsy James DiSalvo, Sarita Yardi, Mark Guzdial, Tom McKlin, Charles Meadows, Kenneth Perry, and Amy Bruckman. 2011. African American men constructing computing identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2967–2970.

Andrew J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six learning barriers in end-user programming systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing*. IEEE, 199–206.

Roxane Koitz and Wolfgang Slany. 2014. Empirical comparison of visual to hybrid formula manipulation in educational programming languages for teenagers. In *Proceedings of the 5th Workshop on Evaluation and Usability of Programming Languages and Tools*. ACM, 21–30.

C. Kothe. 2014. Lab streaming layer (lsl). Retrieved October 26, 2014 from https://github.com/sccn/labstreaminglayer.

Christian Andreas Kothe and Scott Makeig. 2013. BCILAB: A platform for brain-computer interface development. *Journal of Neural Engineering* 10, 5 (2013), 056014.

Dave Krebs, Alexander Conrad, and Jingtao Wang. 2012. Combining visual block programming and graph manipulation for clinical alert rule building. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2453–2458.

Mikhail A. Lebedev, Adam Messinger, Jerald D. Kralik, and Steven P. Wise. 2004. Representation of attended versus remembered locations in prefrontal cortex. *PLoS Biol* 2, 11 (2004), e365.

Catherine Letondal. 2006. *Participatory Programming: Developing Programmable Bioinformatics Tools for End-users*. Springer, 207–242.

Lifelong-Kindergarten. [n.d.]. Scratch-vm. Retrieved on August 1, 2018 from https://github.com/LLK/scratch-vm.

John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 16.

Thomas J. McCabe. 1976. A complexity measure. *IEEE Transactions on Software Engineering* 4 (1976), 308–320.

Monica M. McGill. 2012. Learning to program with personal robots: Influences on student motivation. *ACM Transactions on Computing Education (TOCE)* 12, 1 (2012), 4.

J. d. R. Millán, Frederic Renkens, Josep Mourino, and Wulfram Gerstner. 2004. Noninvasive brain-actuated control of a mobile robot by human EEG. *IEEE Transactions on Biomedical Engineering* 51, 6 (2004), 1026–1033.

Amon Millner and Edward Baafi. 2011. 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, 250–253.

Christian Múhl, Hayrettin Gürkök, Danny Plass-Oude Bos, Marieke E. Thurlings, Lasse Scherffig, Matthieu Duvinage, Alexandra A. Elbakyan, SungWook Kang, Mannes Poel, and Dirk Heylen. 2010. Bacteria hunt. *Journal on Multimodal User Interfaces* 4, 1 (2010), 11–25.

Gernot R. Müller-Putz, Reinhold Scherer, Gert Pfurtscheller, and Rüdiger Rupp. 2005. EEG-based neuroprosthesis control: A step towards clinical practice. *Neuroscience Letters* 382, 1 (2005), 169–174.

Neurosky. 2018. Retrieved on August 1, 2018 from http://neurosky.com/.

Grace Ngai, Stephen C. F. Chan, Hong Va Leong, and Vincent T. Y. Ng. 2013. Designing i* CATch: A multipurpose, education-friendly construction kit for physical and wearable computing. *ACM Transactions on Computing Education (TOCE)* 13, 2 (2013), 7.

OpenBCI. 2018. Retrieved on August 1, 2018 from http://openbci.com/.

Gert Pfurtscheller, C. Guger, G. Múller, G. Krausz, and C. Neuper. 2000. Brain oscillations control hand orthosis in a tetraplegic. *Neuroscience Letters* 292, 3 (2000), 211–214.

Matthew Pike, Richard Ramchurn, Steve Benford, and Max L. Wilson. 2016. # scanners: Exploring the control of adaptive films using brain-computer interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 5385–5396.

Robert T. Pivik, Roger J. Broughton, Richard Coppola, Richard J. Davidson, Nathan Fox, and Margreth R. Nuwer. 1993. Guidelines for the recording and quantitative analysis of electroencephalographic activity in research contexts. *Psychophysiology* 30, 6 (1993), 547–558.

Shaileen Crawford Pokress and José Juan Dominguez Veiga. 2013. MIT app inventor: Enabling personal mobile computing. *arXiv preprint arXiv:1310.2830*.

Alan T. Pope, Edward H. Bogart, and Debbie S. Bartolome. 1995. Biocybernetic system evaluates indices of operator engagement in automated task. *Biological Psychology* 40, 1 (1995), 187–195.

Yann Renard, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand, and Anatole Lécuyer. 2010. Openvibe: An open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments. *Presence: Teleoperators and Virtual Environments* 19, 1 (2010), 35–53.

Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, and Brian Silverman. 2009. Scratch: Programming for all. *Communications of the ACM* 52, 11 (2009), 60–67.

Ricarose Vallarta Roque. 2007. *OpenBlocks: an extendable framework for graphical block programming systems*.

Gerwin Schalk, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R. Wolpaw. 2004. BCI2000: A general-purpose brain-computer interface (BCI) system. *IEEE Transactions on Biomedical Engineering* 51, 6 (2004), 1034–1043.

Alois Schlogl, Clemens Brunner, Reinhold Scherer, and Andreas Glatz. 2007. Toward brain-computer interfacing. *BioSig: An Open-Source Software Library for BCI Research*. https://ieeexplore.ieee.org/servlet/opac?bknumber=6267251.

Robert H. Sloan, Cynthia Taylor, and Richard Warner. 2017. Initial experiences with a CS law introduction to computer science (CS 1). In *ITiCSE'17*. https://ssrn.com/abstract=2945687.

Jay Summet, Deepak Kumar, Keith O'Hara, Daniel Walker, Lijun Ni, Doug Blank, and Tucker Balch. 2009. Personalizing CS1 with robots. *ACM SIGCSE Bulletin* 41, 1 (2009), 433–437.

Daniel Szafir and Bilge Mutlu. 2012. Pay attention!: Designing adaptive agents that monitor and improve user engagement. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 11–20.

Daniel Szafir and Bilge Mutlu. 2013. ARTFul: Adaptive review technology for flipped learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1001–1010.

Desney Tan and Anton Nijholt. 2010. *Brain-computer Interfaces and Human-Computer Interaction*. Springer, 3–19.

Bastian Venthur and Benjamin Blankertz. 2012. Mushu, a free-and open source BCI signal acquisition, written in Python. In *Proceedings of the 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'12)*. IEEE, 1786–1788.

Bastian Venthur, Sven Dähne, Johannes Höhne, Hendrik Heller, and Benjamin Blankertz. 2015. Wyrm: A brain-computer interface toolbox in python. *Neuroinformatics* 13, 4 (2015), 471–486.

Bastian Venthur, Simon Scholler, John Williamson, Sven Dähne, Matthias S. Treder, Maria T. Kramarek, Klaus-Robert Müller, and Benjamin Blankertz. 2010. Pyff—A pythonic framework for feedback applications and stimulus presentation in neuroscience. *Frontiers in Neuroscience* 4 (2010), 179.

Jacques J. Vidal. 1977. Real-time detection of brain events in EEG. *Proceedings of the IEEE* 65, 5 (1977), 633–641.

David Weintrop and Uri Wilensky. 2015. To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 199–208.

John Williamson, Roderick Murray-Smith, Benjamin Blankertz, Matthias Krauledat, and K-R Müller. 2009. Designing for uncertain, asymmetric control: Interaction design for brain-computer interfaces. *International Journal of Human-Computer Studies* 67, 10 (2009), 827–841.

Jonathan Wolpaw and Elizabeth Winter Wolpaw. 2012. *Brain-Computer Interfaces: Principles and Practice*. OUP USA.

Jonathan R. Wolpaw, Niels Birbaumer, Dennis J. McFarland, Gert Pfurtscheller, and Theresa M. Vaughan. 2002. Brain-Computer interfaces for communication and control. *Clinical Neurophysiology* 113, 6 (2002), 767–791.

Matthew Wright. 2005. Open sound control: An enabling technology for musical networking. *Organised Sound* 10, 03 (2005), 193–200.

Shuo Yan, GangYi Ding, Hongsong Li, Ningxiao Sun, Yufeng Wu, Zheng Guan, Longfei Zhang, and Tianyu Huang. 2016. Enhancing audience engagement in performing arts through an adaptive virtual environment with a brain-computer interface. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, 306–316.

Yueran Yuan, Kai min Chang, Jessica Nelson Taylor, and Jack Mostow. 2014. Toward unobtrusive measurement of reading comprehension using low-cost EEG. In *Proceedings of the 4th International Conference on Learning Analytics and Knowledge*. ACM, 54–58.

Received September 2018; revised April 2019; accepted May 2019