Using Monogame for Accessible Developmental Software

A Technical Report Submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Jared Conway
Fall 2024

Mark Sherriff, Department of Computer Science

Using Monogame for Accessible Developmental Software

Jared Conway

CCS CONCEPTS

• Human-centered computing → Accessibility; • Applied computing → Education;

KEYWORDS

Developmental software; education; concentration; game engine

1 INTRODUCTION

Between 2011 and 2013, the amount of smartphone usage among children under the age of two has risen from around 11% to 38% [1]. Although some parents may view this increase as a negative, others view it as a new opportunity to help raise their children. Developmental software, for example, is one such opportunity that can be used to train kids on a variety of different topics, including number sense, reflex, and writing [1]. One other such topic some software trains is concentration.

Concentration as a concept has been misunderstood for years. For example, news articles, such as TIME, have claimed that the human attention span is around 8 seconds long. TIME has even gone as far as to say that goldfish have better attention spans than the average person [2]. Regardless if attention spans are on the decline or not, the 8 second statistic has been heavily sensationalized. This only refers to the length of time a person is willing to spend on a webpage before clicking off, but not their attention spans in other contexts [3]. For example, lectures that run for as long as 50 minutes saw "no difference nor decline in retention rates of material" compared to 25 minute lectures [3].

Given that attention spans can vary wildly given different contexts, it is better to consider concentration not as the length of time someone is willing to devote to doing one thing. Rather, concentration is their expectations about the length of time, as well as their willingness to spend extra time. For example, someone with good concentration would not only spend longer than 8 seconds on a website before losing their patience, but also longer than 50 minutes in a lecture. Therefore, the goal of developmental software that seeks to improve concentration focuses more on improving their users' willingness rather than increasing a specific length of time.

2 RELATED WORK

One approach used to help train concentration is the usage of electroencephalography, or EEG, headsets. These specialized headsets are capable of scanning five particular brain waves to calculate the users' restfulness and attentiveness. Alpha waves cover the 8-12 Hz frequencies and correspond with restfulness and introspection [4]. Beta waves cover 13-30 Hz and correspond with "normal waking consciousness," or normal, everyday behavior [4]. Gamma waves are around 40 Hz and correspond with attentiveness [5]. Delta covers 1-4 Hz and occurs during sleep. Theta covers 4-8 Hz and occurs during deep meditation [4]. Combined together, researchers can determine whether the user is restful and willing to continue testing, or unfocused and bored.

One company which produces these EEG headsets for everyday use is Neurosky. This company has not only offered multiple different models to its clients, but has created a platform which allows developers to publish their own games that utilize their EEGs. The most recent model unveiled in 2018 was the MindWave Mobile 2, a headset specifically designed for comfort and usability; however, not all users agreed with this sentiment. Several reviews on Amazon revealed that the headset was "incompatible with newer operating systems like Windows 11" due to the drivers not being updated and maintained. Other reviews claimed that the headset had bluetooth connectivity issues, making the EEG hard to use and develop with. Although users could be asked to downgrade their operating system to support the outdated drivers, this unnecessary friction does not align with the instant gratification and ease of use people nowadays come to expect [6]. If the goal of this approach is to help people improve their concentration, requiring a high level of willingness and patience before training has started is not a good idea.

In spite of the issues with the hardware, the software has shown to produce favorable results. One of the main applications designed for Neurosky headsets is called Focus Pocus - a magic themed game designed for kids. Utilizing the different brain waves, the game's minigames are able to train multiple different types of concentration: focus, relaxation, and meditation. After 85 children were trained on each of these areas for 7 to 8 weeks, researchers found an increase in alpha wave activity, and a decrease in delta, suggesting an increase in attentiveness [7]. There was also a decrease in engagement over the training period [7].

Interestingly, not every minigame in Focus Pocus utilizes the EEG. "Hex Practice," for example, is a card matching game which only requires a mouse. Since certain minigames in Focus Pocus operate fine without the usage of the EEG, developmental software could be made without such equipment; therefore, the driver and bluetooth problems would be resolved. My technical project, for example, does not rely on such hardware. Instead, it relies on hardware the average computer user has: a keyboard, and mouse. The project additionally uses .NET and Monogame to allow the software to run on modern operating systems, such as Windows 10. Following the design principles of low coupling and high cohesion, the software contains a strong foundation other developers can reuse for their own games. However, as the main goal of the technical project is to provide an alternative to EEG based developmental software, the project also includes an example game to showcase the validity of this approach.

3 OVERVIEW

The example game for the project is called "Focus App" which showcases a character named Stario who can transform into a ring. The game requires the user to keep their mouse cursor over Stario while distractions are shown on screen. Once the user keeps their mouse over Stario for enough time, the user wins the level and receives their score. Afterwards, the user can press the escape key to return to the main menu to try again, or try a different level. This example game is complete with a level select system, a level editor, ranking system, and leaderboard.

The level editor was created to aid development as well as to allow users to create their own levels, thereby allowing users to be more involved with the game. The ranking system and leaderboard were created to appeal to agôn, or competition, which is one of the four elements of play [8]. Social based mechanics, such as leaderboards, are known to have one of the strongest correlations with video game addiction, having a β value of 0.2 [9]; therefore, by including a leaderboard, engagement for the game can be further strengthened. Ideally, games which are able to appeal to all four elements (agôn, alea, mimicry, and ilinx) [8] can lead to the most engagement, but that is out of scope for this project.

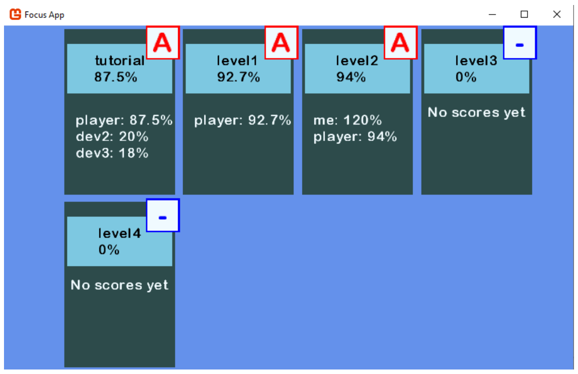


Figure 1: Select Select Screen. Users can select one of five different levels to play. Levels will show the best score and rank the player achieved, as well as the leaderboard scores.

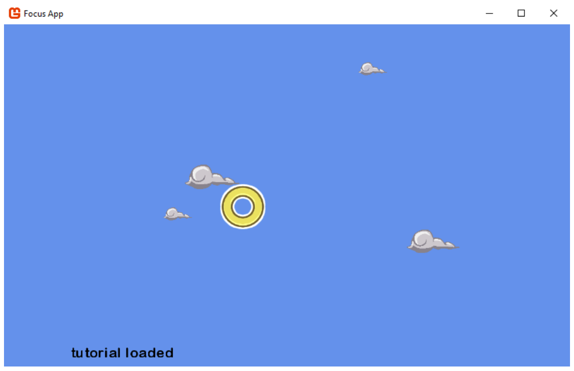


Figure 2: Tutorial Level. Clouds will appear and move left and right across the screen to try to distract the player.

4 IMPLEMENTATION

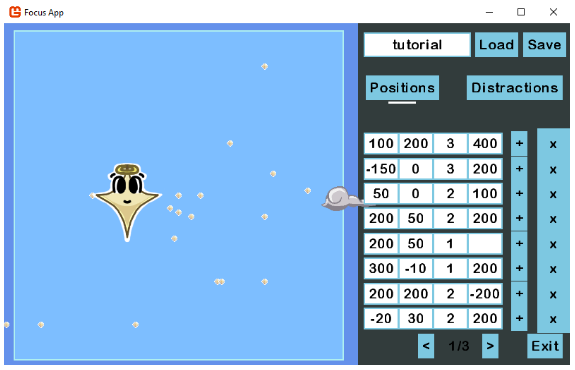
To have full control over the software, the Monogame framework was selected over conventional game engines, such as Unity. As it is the goal of this project to be as accessible as possible, using Monogame has allowed the build to be around 18 MB, uncompressed. Although less powerful, the cross-platform version of Monogame was selected to allow Mac and Linux developers to compile their games for non-Windows machines.

As this project was made in Monogame, most features normally seen in game engines, such as a user interface or sprites, were not implemented. Therefore, while working on the example project, foundational code was implemented when needed. Object Oriented Programming (OOP) was heavily used while implementing such foundational code in order to reduce redundancy and keep the project organized. UI buttons, for example, extend the UIElement class, which in turn extend the GameObject class. As GameObjects already have position, scale, and texture information, these variables were able to be reused for buttons.

The level editor was one of the first features added to the game, as well as one of the longest to implement given the lack of foundational code. Level data is stored as Json to allow new levels to be easily added, as well as to add in additional tags when new mechanics are created. Each level Json object is of the form:

```
“level-name”: {
        “author”: “author-name”,
        “difficulty”: 0
        “positions”: [[x1, y1, time1,
        offset], [x2, y2, time2, offset2] …],
        “distractions”: [[x1, y1, time1],
        [x2, y2, time2] …],
        “accuracy”: accuracy
}
```

Note that difficulty is unused. For both positions and distractions, x and y correspond to the x and y positions of the ring or distraction at a given time. For positions, the ring linearly interpolates between (x1, y1) and (x2, y2) based on the current time. For example, a gametime of 1.5 with a time1 of 3 would have the ring equally distanced between the points (x1, y1) and (x2, y2). Additionally, a non-zero offset value will cause the ring to arc between the two points, rather than moving straight. This is to provide more interesting movement. Providing a null value for the offset will cause the ring to teleport from point to point instead. For distractions, when the current time reaches time1, a distraction will appear at (x1, y1), disappearing soon after. A different distraction will appear at (x2, y2) when the time reaches time2. Unlike the ring, multiple distractions can be seen on screen at once.



tutorial

100	200	3	400
-150	0	3	200
50	0	2	100
200	50	2	200
200	50	1	
300	-10	1	200
200	200	2	-200
-20	30	2	200

Figure 3: Level Editor. Users can click on the Positions and Distractions subtabs to edit Stario’s data or distraction data. Columns 1 and 2 are x and y, 3 is time, and 4 is offset.

The leaderboard system was implemented by using Firebase’s realtime database as well as Representational State Transfer (REST). Firebase was selected as the leaderboard database as it is both free to use and updates in real time. Sending a score to the database can be done using a PATCH request with the level name, player name, and score. An example score of 87% with a player name of “player” for the level “lvl1” can be done with the following:

```
var json = $"{{\"player\": 0.87f}}";
var content = new StringContent(json, System
.Text.Encoding.UTF8, "application/json");
await Registry.CLIENT.PatchAsync(Registry
.LEADERBOARD_URL + "lvl1.json", content);
```

Registry.CLIENT corresponds to the HTTP Client created at runtime, while Registry.LEADERBOARD_URL is the URL for the leaderboard (ie: https://[leaderboard-name]-rtdb.firebaseio.com/). This may alternatively be called from the LevelReader class, using:

```
LevelReader.UpdateLeaderboard(“lvl1”,0.87f);
```

This method will run asynchronously, allowing users to continue to use the application while the leaderboard updates. Note that the above method has “player” hard coded as the player name. This is due to not having enough time to implement a username system, as well as to mitigate users from entering obscene language as their name. Leaderboard information can be retrieved using:

```
var info = await
Registry.CLIENT.GetStringAsync(Registry.LEADE
RBOARD_URL + ".json");
```

Such information is then parsed as a JsonObject and separated by level names. As a benefit of using Json to store scores, custom levels created with the level editor will automatically have their own section in the leaderboard added as soon as any user completes it, following a data-driven design.

5 RESULTS

The final project is around 18 MB in size and only requires around 200-250 MB of RAM, making the project usable on lower-end devices. The example game is complete with 5 levels as well as leaderboards for each. Unlike EEG based games, this example game does not require any hardware the average computer user does not already have, nor require an outdated operating system. Developers will have access to the bare essentials to creating any simple game using this project's engine, whether it be for developmental software or not. Additionally, due to the OOP design of the foundation, it is easy for developers to add additional features to the codebase if needed as they may be able to extend already existing features.

Originally, the project would have also used an eye tracker instead of the mouse cursor to follow the ring. The underlying logic was that if the user was not looking at the ring, then they likely were not concentrating, and vice versa. However, I realized early on that the project could operate fine without the usage of the eye tracker; therefore, it was decided to save the eye tracker for later if there was still enough time to implement it. While researching various eye trackers online, most were either not in the budget (free), or used the GNU license. Had I used any of the GNU licensed eye trackers, I would not have retained any ownership of the example project or engine if it was used. Eventually, I settled on a Github repository which implemented an eye tracker in Python with an MIT license. After tweaking and compiling the eye tracker as an executable, I was able to run the file as a process and redirected the standard output from the console to the game.

Unfortunately, as the eye tracker was compiled by an unknown author and tried to access the computer's camera, my antivirus software repeatedly deleted the eye tracker every time the game was run. To mitigate this, I disabled the antivirus every time I decided to run the game. Besides the problem with the antivirus, there were also problems with properly lighting the room in order for the eye tracker to register my eyes. I also had to sit perfectly still and look straight ahead as any deviation threw the tracker off. Even once my lighting and posture were perfect, the eye tracker was still extremely variable in where it thought I was looking - it was completely unreliable. After this, I decided that the mouse was adequate for the project as users would still need to concentrate to keep the mouse on the ring.

6 FUTURE WORK

Further work may be done for the example game in multiple different ways. For example, the leaderboard does not have any security measures in place to authenticate users who submit scores, nor verify if the score submitted is real. To help with user authentication, RSA encryption could be used on PATCH requests to verify that the person who sent the score is who they say they are. Unfortunately, not much can be done for score verification as the game is client-sided. Users could be required to send a recording of their gameplay to validate their scores, although this would not scale well.

The example project may be further improved by including more levels. These levels would be longer than the starting levels and have different types of distractions. As the engine supports animations using the Animation and AnimationCollection classes, these levels could have distractions which visually change. These distractions would be more interesting to look at than static images, therefore making it harder to pay attention.

In addition to improving the example project, further work may be done for the game engine. For example, to make the engine more approachable for developers, a full engine API documentation could be written. This documentation would detail the structure of each class, class usages, as well as example implementations. Code from the example project could be reused for the examples, although additional examples would be beneficial.

To help developers find out about the engine, it would be useful to have a platform for developers to share their games made with the engine. One such existing platform which could be utilized is the Steam Workshop - a platform made for developers to publish mods and content for games. By uploading the game engine to Steam, other developers could upload their games as mods to the engine with little hassle. This approach would have the added benefit of allowing these developers to reach more users as Steam is one of the largest platforms in the world for gaming [10].

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Mark Sherriff at the University of Virginia for assisting me with revising this paper as well as for keeping development of the game engine on track. I am grateful for the time he has given me, as well as his support for the creation of this engine. I would also like to thank Professor Upsorn Praphamontripong, now with the University of Albany, for the time and support she has also given me for the early development of the engine.

REFERENCES

[1] Jodi Gold. 2015. *Screen-smart parenting* (1st. ed.). New York, NY: The Guilford Press. Retrieved from https://www-r2library-com.proxy1.library.virginia.edu/Resource/Title/1462515533.

[2] Kevin McSpadden. 2015. You Now Have a Shorter Attention Span Than a Goldfish. Retrieved from https://time.com/3858309/attention-spans-goldfish/.

[3] Neil A. Bradbury. 2016. Attention span during lectures: 8 seconds, 10 minutes, or more? *Advances in Physiology Education* 40, 4 (Nov. 2016), 509-513. doi: 10.1152/advan.00109.2016

[4] Jim Robbins. 2008. *A Symphony in the Brain : The Evolution of the New Brain Wave Biofeedback*, Grove/Atlantic, Incorporated. Retrieved from http://ebookcentral.proquest.com/lib/uva/detail.action?docID=5503798.

[5] John S. Barlow. 1993. *The Electroencephalogram: Its Patterns and Origins*, MIT Press. Retrieved from https://books.google.com/books?id=LPN-xm_POfMC.

[6] Paul Roberts. 2014. *The impulse society: America in the age of instant gratification*. Bloomsbury Publishing, USA. Retrieved from https://books.google.com/books?id=TgkbBAAAQBAJ.

[7] Stuart J. Johnstone, Steven J. Roodenrys, Kirsten Johnson, Rebecca Bonfield, and Susan J. Bennett. 2017. Game-based combined cognitive and neurofeedback training using Focus Pocus reduces symptom severity in children with diagnosed AD/HD and subclinical AD/HD. *International Journal of Psychophysiology*, 116 (Jun. 2017), 32-44. doi: 10.1016/j.ijpsycho.2017.02.015

[8] Roger Caillois and Meyer Barash. 2001. *Man, Play, and Games*, University of Illinois Press. Retrieved from https://books.google.com/books?id=bDjOPsjzfC4C.

[9] Damien C. Hull, Glenn A. Williams, and Mark D. Griffiths. 2013. Video game characteristics, happiness and flow as predictors of addiction among video game players: a pilot study. *Journal of Behavioral Addictions*, 2, 3 (Sept. 2013) 145-152. doi: 10.1556/jba.2.2013.005

[10] Robert Hoile. 2017. *Sparking a steam revolution: Examining the evolution and impact of digital distribution in gaming*. Charlottesville, VA: University of Virginia, English - Graduate School of Arts and Sciences, MA (Master of Arts). doi: 10.18130/V3Z369