Increasing productivity by combining the Pomodoro Technique with 2D Idle Collection Games

Ledet Awano
Texas A&M University
400 Bizzell St, College Station, TX 77843
home01@tamu.edu

Grace Fan
Texas A&M University
400 Bizzell St, College Station, TX 77843
gracefan@tamu.edu

Brandon Neff
Texas A&M University
400 Bizzell St, College Station, TX 77843
brando_18@tamu.edu

Charles Wong
Texas A&M University
400 Bizzell St, College Station, TX 77843
charleswong@tamu.edu

ABSTRACT

Developed as a term project for the Spring 2019 semester of Computers and New Media at Texas A&M University, Rooted is an idle collecting game that incorporates principles from the Pomodoro Technique to enhance player's productivity when working or studying. Rooted is a web browser game developed using HTML5/CSS and JavaScript, designed specifically to avoid addicting or distracting game mechanics such as loot boxes. In this paper, we discuss our design decisions in development, the goals we hoped to accomplish, and how Rooted could be expanded in the future.

Keywords

Games, game design, game development, web games, zen game, idle-gaming, collecting games, casual games, indie games, gamification, games with a purpose, positive computing, positive gaming, Pomodoro Technique, entertainment, productivity, rewards, addiction

1. INTRODUCTION

"Rooted" is our team's 2-dimensional idle-collection game. It simulates a zen garden setting with an arrangement of decorations picked by the player. Each decoration can attract a visiting animal. The game was designed with a goal of accumulation: players gain money when visitors arrive and then spend money on more decorations to attract more visitors.

1.1 Game Purpose

The system was designed to be a fun time-management system that engages the user while also boosting their productivity. The core component of Rooted and other idle games is waiting time [9]. Idle game players can be stopped from interacting with the game during specific periods by using game mechanics such as automatic progression and slow spawn events. Requiring the player to wait for the arrival of visitors allowed us to exert control over the amount of interaction the player could have with the game in specific windows of time.

By tuning the wait times and dividing the game into stages of idle-play and idle-wait, we created overall stages of increased or decreased player attention to the game. The purpose of developing for this cycle was to follow the overall structure of the Pomodoro Technique for managing time and breaking down work. Our game seeks to encourage regular use throughout work times.

2. PROBLEM STATEMENT

Our game is designed to solve both the issues of difficult time management and distracting games. The primary motivation for addressing these issues is to find the right intervals to maximize focus, while minimizing stress from studying with deadlines. In educational environments, periods of procrastination and extended work damage overall productivity. Effective studying requires both freedom from distractions as well as effective time budgeting. Our overall goal for the product was to create a less distracting media source that would increase productivity through regular use.

2.1 Effective Time Management

Effective time managements skills is difficult to achieve since it is such a complex subject. It is possible that working nonstop for too long can decrease productivity [10], while smartphones and the internet can also cause frequent distractions that disrupt productive work altogether [4]. Some people have developed techniques to assist in improving productivity and effective time management; one such technique is known as the Pomodoro Technique [2]. As shown in Figure 1, the main characteristic of the technique is cycling working nonstop for 25 minutes followed by a short 3-5 minute break. Implementation of the Pomodoro Technique requires a timing system to alert the user to a switch from work to break. Current approaches involve using a analog timer or a Pomodoro program. With Rooted, we aimed to understand feasibility of using a game to run the Pomodoro cycles.

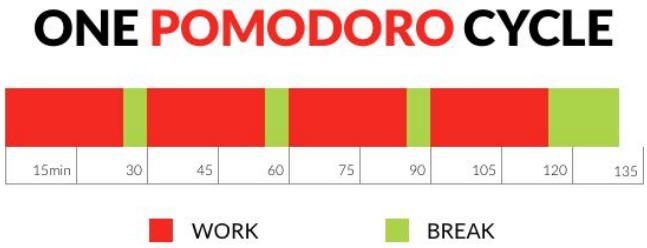

Figure 1. Illustration of a typical Pomodoro work cycle with 25-minute/5-minute work and relax periods.

2.2 Minimizing Distraction

The other issue we attempt to solve is the problem of games that are designed to be addicting and distracting. Many modern games, especially mobile games, incorporate mechanics similar to gambling to capture player's attention and entice them to spend

money on microtransactions [1][7]. Over the past several years, there has been a paradigm shift across the gaming industry as more and more companies integrate "loot boxes," microtransactions, etc., into their games. A loot box is an in-game item that can be opened for a rare chance of receiving a valuable item, such as a weapon or cosmetic tweak for in-game characters. In some games such as *Overwatch*, loot boxes cost money (Figure 2); in others, loot boxes are freely provided but players must purchase keys to actually unlock them. Increasingly, many games are free-to-play and receive all of their revenue from microtransactions made by players.

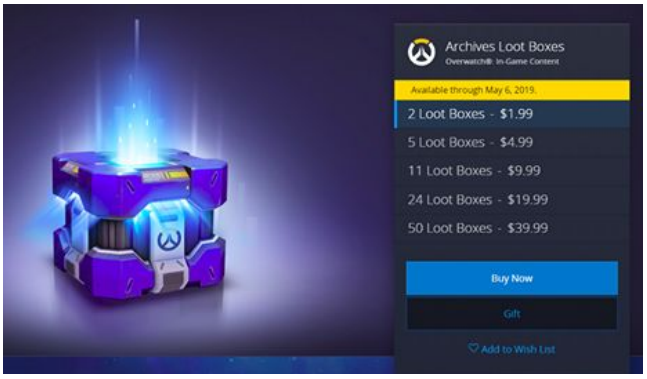

Figure 2. "Loot Box" prices from the video game *Overwatch* (Blizzard Entertainment, 2016).

Some governments have begun regulating these gaming features as gambling [6]. Our goal was to design a game that worked against these industry trends by aiming to not be addicting or distracting, and instead forcing users to take breaks through our design and gameplay.

2.3 Our Solution

To solve these two problems, we have developed our game as an idle collecting game that is meant to be played passively. Users will have a 5 minute window to take actions in the game, such as purchasing plants and placing them in the garden. At the end of the 5 minute window, the game goes into an automated state where no interactions can be made; the game continues to run in the background as visitors come to the garden and the user earns currency. During this 25 minute window, the user is supposed to have a distraction free opportunity to be productive. After the 25 minutes are up, the user can see which animal visitors have come to the garden and use their newly earned in-game currency to buy additional plants. Essentially, our game incorporates the Pomodoro Technique and limits the amount of time the user can play the game. We have also avoided using any features such as microtransactions, loot boxes, or excessive push notifications.

3. DESIGN & DEVELOPMENT ISSUES

Throughout the semester of working on this project, our team encountered a number of issues. In this section, we will overview which issues were the most challenging for us to overcome and how they affected our design decisions and development progress.

3.1 Phaser 3

One of the main issues that we encountered in the design of our application was the use of the Phaser framework. Our original idea was to use this game building framework that is written in JavaScript to implement all the design and functionality needed to have an interactive game. As we began to work on the project we realized after six hours of working that the framework would not be compatible for the application we were building. One of the main reasons that caused this was the lack of documentation that the framework had. The organization that built the framework had just build the 3rd edition of that framework and had changed many of the function calls and implementation details needed to use the game framework.

With that, as we began to do research and work on the project we noticed that they had not documented the current framework enough. This caused our group to have many difficulties integrating the framework into our application. We later choose to scrap the framework and work with JavaScript alone with the help of jQuery.

3.2 jQuery & Yarn

Because of our use of jQuery to grab JSON data about our visitors and items, we also developed a need to host our project on a server. In Google Chrome, fetch requests made for files are considered "cross-origin" if they are not from the same domain as the page [14]. Requests to the local filesystem are unsupported. To get the fetch to work successfully, we used the Yarn package http-server, which was made for local development of web pages. Use of Yarn and http-server is shown in Figure 3It allowed us to serve the project files to Chrome as if we were hosting them.

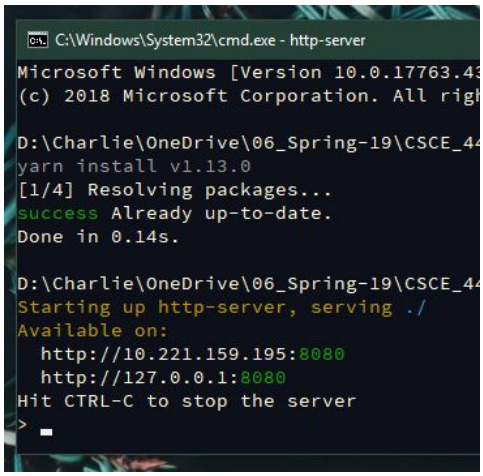

Figure 3. Process for locally hosting Rooted.

3.3 Finding Appropriate Assets

One of the main features of our application was the use of assets to set the tone of the game and create a fun environment that would attract our users to continue to use or application. When we came up with our initial design we had decided to sketch and get a rough idea as to how we wanted our characters and inventory to look. After we completed our designs we went to open-source game art databases to see if we could find images that were similar to the sketches we had come up with. After a while of searching we were not able to find images that resembled the sketches that we had come up with.

Figure 4. Comparison of asset inspiration and final asset.

We collectively decided to create the assets ourselves using Adobe Illustrator. This would then allow us to mimic the designs that we created and continue on with the theme we had wanted to portray in our application (Figure 4). With all the characters and inventory items we had this task took quite a bit of time to complete. Our project team comprised of four people and the design of our assets was built by one of our team members. This task took them the majority of the time that we spent working on the application build, which overall caused our team to slow down in building efforts.

3.4 Responsiveness

Another challenge that we ran into was making our application responsive, both in terms of different browser sizes and mobile views. With our application being a idle game we were concerned with the format of the game when displayed on smaller screens such as tablets and cellular devices. We began by testing our application on various sized devices that our team members had and came up with a sizing factor that would maintain the appeal of our application while also allowing for easy navigation of the store and inventory log that users had access to. We solved the issue of mobile responsiveness by simple using media queries to display a different type of layout depending on the width of the screen. However, a more long term solution would be to simply develop this app through Unity or another game development platform so it can be specifically tailored for mobile.

Another issue with responsiveness is shown in Figure 5 below. We were using absolute positioning at first for easy layout, but the orange circles shifted away from their relative positioning on the background image.

Figure 5. Example of a responsiveness issue found in early prototypes.

4. DESIGN SYSTEM & MOTIVATION

In order to make an appealing game that would be accessible to a wide audience, our team decided to make the garden display the main focus of the interface. To bring the focus to the garden, we made the inventory/store menu collapsible. We used a drawer-style menu for listing the items in the inventory and store so that they could be easily browsed by the player. The menu itself was made with compact spacing so that more screen space could be allocated to showing the garden.

Figure 6. Collapsed state of Inventory/Store menu.

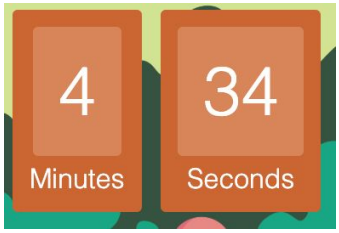

Figure 7. Five minute timer counting down.

The timer is placed in the top left position so that it is unobtrusive during gameplay. Once the five minute timer finishes, a semi-opaque black overlay is placed over the screen and the 25-minute timer begins. This interaction indicates to the user that they must begin their productive cycle again, however, the game will still continue to run in the background.

Figure 8. Semi-opaque 25 minute overlay.

4.1 Control Scheme

The control scheme for our system is an intuitive drag-and-drop by mouse. Both the inventory items and the garden slots act as large targets for mouse clicking. The decision to maximize click areas allowed for faster item placement [5]. Unoccupied garden slots display an orange disc to show players that an item can be placed (Figure 10). By doing this, we allow the player to check on the status of the garden at a glance, without needing to read text. To return an item to the inventory, the player only has to double click it in the garden. The motivation for this interaction was to shorten the time it takes for the player to rearrange their garden. Since the edition period only lasts for 5 minutes, we wanted to make sure that players would be able to create their arrangement and move on to completing their work.

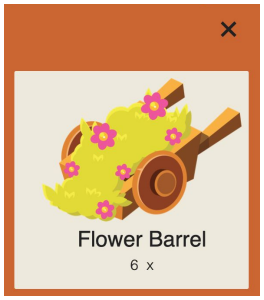

Figure 9. A single inventory item, where the image acts as a click-and-drag target.

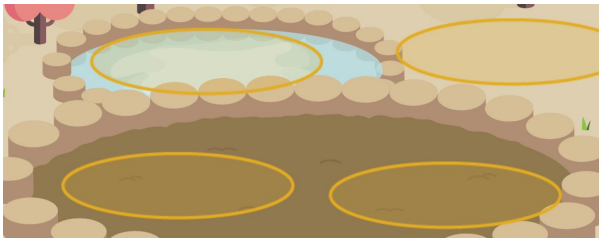

Figure 10. Collection of drag-and-drop targets showing open slots in the garden.

4.2 Visitor Spawn Rate

The spawning of visiting animals was designed in order to give a sense of progression. Every five seconds, the game chooses a random garden slot and attempts to spawn a visitor. The visitor will fail to spawn if the slot is unoccupied with a decoration, or the decoration already has a visitor. This makes the game more appealing because it adds an element of chance. It also incentivizes the player to place more decorations in the garden to increase the odds of successful spawns. A chain of successful spawns can give the player a chance to progress quickly, while the overall game progress is still slow due to the maximum spawn rate of 1 visitor per 5 seconds.

4.3 Visual Style

The visual style of the game was designed to be soothing, using muted colours for the background and visitors to provide a relaxing experience. The colours of the inventory and store buttons are the same muted shade of orange for consistency. There are also icons in addition to the word labels for each button to be explicit about a button's purpose. We chose a variety of garden visitors to illustrate and developed animations for the visitors when they spawn. Idle animation occurs when an object is not being controlled and it adds interest to the screen in a game like ours.

There were several elements that we had planned before development to implement. In terms of visitors, we had listed: birds, bees, frogs, snails, worms, and snakes because they are common garden visitors. For the objects that the player can buy, we thought of bird feeders, potted plants, seeds, koi pond, and water fountain. One easy way to quickly produce more objects is to change the colours of the objects or minor details and advertise them as different options to buy.

4.4 Code Implementation

The implementation of the game was done in HTML5, CSS3, and JavaScript. We also used Yarn to manage our dependencies. We took advantage of native HTML5 support for mouse actions and click-and-drag to simplify programming for user interactions.

4.4.1 Front-end Implementation

We created the inventory objects and store objects with simple tags and styled them into cards with CSS. We kept a consistent styling for all of the buttons and cards. The background image is set to a static image that we lay orange drop-points. Each object, both visitors and garden items, are images or GIFs that we designed and rendered in the HTML with the <img> tag.

4.4.2 Database Implementation

One library feature we used in JavaScript was jQuery's getJSON(). The use of JSON for listing decoration and visitor properties was key to keeping our game dynamic and easy to expand. Anytime we want to implement a new visitor or item, we simply create a new entry in the JSON files hosted on the server. When a user loads the game, their browser will fetch the JSON files using Ajax and use the information to populate a list of items and visitors. The JSON files contain information such as item costs, descriptions, paths to art assets, and names.

The game begins when all of the content in the html <body> tags is loaded. The startGame() function initializes some global values, calls the JSON parsing, and begins the main game loop. From there, a function mainLoop() allows one update every 1000 milliseconds. This format worked well because we wanted to have the mouse inputs handled by the HTML rather than using JavaScript event listeners. The game also did not need to be redrawn very often because there were not complex animations or physics involved; the visitors are animated using GIFs.

4.4.2.1 User Money

User money is tracked as a global variable. Whenever a user attempts to buy an item, the game logic checks the item cost against their total currency. If the user has enough money, the count of the item in their inventory is incremented and their total money is decremented according to the item cost. If the user does not have enough money, the game alerts the user and no transaction is made.

4.4.2.2 Inventory & Store Objects

The item JSON file is used to populate an array of every item possible. Each entry in the array represents a kind of item the player can purchase and use, and includes variables representing how many instances of the item the player owns (which is initialized to zero), the item cost, and a general description about what the item is.

Whenever a player attempts to place an item from their inventory onto the board, the game logic checks the amount of the item in their inventory. If it is greater than zero, they are allowed to drag the item onto an open spot in the garden. Doing this will decrement the amount in their inventory by one. No action will occur if the number of items they are trying to drag onto the screen in their inventory is equal to zero.

Whenever a player double clicks a placed item, the game logic clears it from the garden and increments the count of the item in the players inventory.

4.4.2.3 Visitor Objects

Visitors are also kept in an array, which keeps track of information such as the value of the visitor, whether the visitor is

present in the garden, if the visitor is allowed to spawn, and what type of visitor it is. The value of the visitor is the amount of money it gives to the user if it spawns. This game mechanism was designed so that the user is rewarded for putting down objects and is incentivized to buy more. The type of visitor is randomly chosen from an array of four different types of visitors: frog, rabbit, duck, or butterfly. The visitor object works closely with the inventory objects to determine whether a visitor can spawn or not. When there are no objects placed in the garden, no visitors may spawn. Once we add it to an array of "available" spaces. Every 1000 ms, the game will randomly try to generate a visitor on an available space. This controls the speed at which the visitors generate so there is not an overwhelming amount, nor a very sparse amount of visitors.

5. DESIGN PROCESS

Our team began developing the idea of our game using in-person meetings. We weighed initial ideas based on feasibility and originality. As we began to create and improve prototypes, we relied on online services such as GitHub and Messenger to coordinate the implementation.

5.1 Game Mechanics

Our team started by designing the game mechanics for Rooted. Once we had decided on the necessary menus, we experimented with placements and menu types. We decided on a single menu location that would have an inventory and store tab.

We initially planned to have a garden theme with seasonal cycles that would change the artwork for the game. In the end, we chose to use a single theme to reduce the amount of assets that was required. With our theme decided, and a list of desired assets, we started producing the art.

5.2 Frameworks

Meanwhile, we explored different game frameworks for HTML5/JavaScript. The one we initially decided on was Phaser 3. We expected that a gaming framework would make object interactions with the mouse easier to implement, as well as the rendering and triggering of game events. However, the most recent version of Phaser (3.16) did not have much documentation. We also discovered that the focus of most of the framework functions was related to keyboard controls, animations, and physics. Since our game didn't use most of this, we decided to move from Phaser to a plain HTML/CSS/JavaScript approach.

5.3 Final Implementation

With the plan for implementation decided, we worked to separately implement the game display, mouse interactions, game clock cycle, and JSON parsing. In the final version, we combined all the parts to create event triggers for buying and placing items, as well as random visitor spawns. We experimented with the cost values for items and the spawn rates for visitors to solidify a sense of progression in the game that would allow users to feel invested for extended periods of time.

6. EXISTING COMPETITORS

There are numerous mobile apps and websites that partially overlap with the purpose of Rooted. The two main categories of these apps are existing Pomodoro Technique timer programs, as well as idle collecting games.

6.1 Pomodoro Apps

Applications that implement Pomodoro timers are not new; there are many mobile applications, along with web browser extensions, that do this already [8]. However, the majority of these apps are simple timers integrated with to-do lists and charts. In spite of the lack of diversity in Pomodoro apps, the technique itself has still proven to be effective in increasing productivity and it is important to see how it is currently being implemented. We looked at two popular models, Focus Booster and Focus Keeper Pro as case studies into existing products.

6.1.1 Focus Booster

Focus Booster is a Pomodoro timer that automatically records sessions into a file. Its primary purpose seems to be to replace timesheets in the workplace. It creates bar graphs and pie charts based on your productivity patterns. It claims to create better work habits by helping you visualize your progress and find a work/life balance.

6.1.2 Focus Keeper Pro

Focus Keeper Pro is another Pomodoro timer. Its main feature is that it shows charts generated based on sessions over the past 30 days (Figure 11). It allows you to customize focus sessions, goals, and the interface itself. The basic steps listed in their manual are: choosing a task to complete, setting a 25 minute timer, working until the timer elapses, and then taking a 5 minute break.

Figure 11. Interface of Focus Keeper Pro, showing the timer, trends graph, and settings

6.2 Idle & Collecting Games

On the other hand, there are numerous idle and collecting games, but many are addictive. Idle games play themselves without user being present, while collection games allow users to gather game objects to create an inventory or accumulate points. We believed that idle-collecting games would be a good complement to the off-time in the Pomodoro technique because some games have a relaxing quality to them that we wanted to tap into.

6.2.1 Cookie Clicker

One in particular that we looked at is called Cookie Clicker. The premise is to click a button to generate cookies. Once you have generated enough cookies, you can buy upgrades to make generating cookies easier. However, there is an increase in playtime because in the beginning, you have to constantly click the button to generate enough cookies. Once you have a steady mechanism for generating cookies, you have to monitor when you have enough to buy upgrades (Figure 12). This endless cycle is what makes collecting games too addictive and ultimately, time-consuming. Additionally, these types of games can be

stressful if the player does not keep up with the latest updates, which further feeds into their addiction.

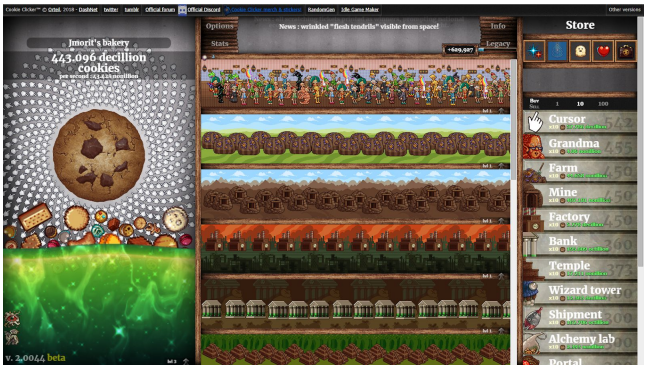

Figure 12. Cookie Clicker game interface mid-game after collecting numerous items

We did think it was important to see how Cookie Clicker laid out their application, seen in the figure above, because it is also a browser game. Because this is such a popular game, we wanted to analyze why it became popular and how it utilizes the interface to maximize user play-time.

6.2.2 Neko Atsume: Kitty Collector

Neko Atsume is an idle-collecting game that served as the inspiration for our product. The general idea of this game is to put down food for cats to visit, with over 60 cats to collect overall. These cats are all unique in name, personality, and appearance. The simplicity of the game makes it accessible to people worldwide. Not only is its art style cute, but it is also laid back. It is unobtrusive because users usually check in for less than 5 minutes as cats spawn. We wanted to emulate this casual playstyle that makes users relaxed when playing and combine it with the Pomodoro technique.

Figure 13. (Left) Inventory and Store in Neko Atsume; (Right) Catalog of visitors both obtained and yet to obtain

In Figure 13 above, the inventory and store are combined in Neko Atsume. It seems that other than replenishable food, you can only have 1 quantity of an object, such as the Rubber Ball (Red). We have to keep in mind, this game is optimized for mobile play and game controls rely heavily on taps instead of clicking or dragging. On the right side of the figure, we see the catalog of visitors, which is essentially what the user is trying to "collect" and in this situation it is cats. By hiding the ones that the user has not encountered, there is an element of surprise or excitement when a new cat is obtained. The user is also able to obtain more information about a specific cat once it appears in their catalogue.

7. FUTURE IMPLEMENTATION

In this section, we will discuss potential improvements if we were to develop Rooted into a full product. Given the constraints of only working on the project for a few months, there were only so many features or art assets we could implement. Additionally, we never purchased a host for Rooted and have been developing it by running a local server on our own computers.

7.1 Stretch Goals

Once we had the basic framework designed, we also listed several stretch goals we wanted to complete if time allowed. This included parts of the visual system, such as changing the colours and adding more visitors. Other features we want to implement include having a wait time to grow special plants. Instead of buying already grown objects, instead the user could buy "seeds" to customize their garden and have the entire experience. These time features could even be expanded into keep track of the date and having different seasons available to the user, such as summer, fall, winter, and spring. Because of these time-dependent additions, it would also be nice to design browser notifications to alert the user when their 25-minutes of work has elapsed or when a special event has occurred.

7.2 Expanding Game Mechanics And Interaction

To improve our application for the next design iteration we have considered to implement various features for users to interact with. Currently, users have access to 6 items that they can purchase in the plant store. With our next iteration of updates we would like to add 4 more plant items for users to choose from. The plant types that we would add are cactus, strawberry, grape and fern. These inventory items would be hand-designed in Adobe Illustrator to maintain the unique theme of our application. The addition of more items added to the store will incentivise users to maintain their progress and keep up with their goals.

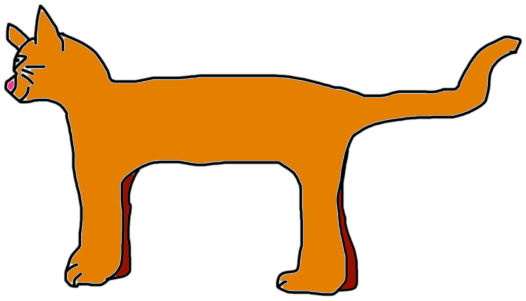

Figure 14. Concept art for a mountain lion (*Puma concolor couguar*).

Additionally, we could relate the animal visitors to the types of plants to add an educational aspect to the game. As users place new items, animals that have a relation to that kind of item could show up, such as panda bears being attracted to bamboo plants or koalas being attracted to eucalyptus trees.

7.3 Cross-Platform Compatibility

Currently our application only runs in web browsers, but with our next redesign we would like to make our application compatible with iOS and Android. By creating our application in React Native we could turn our game into a mobile application. This would allow users to maintain their progress and achieve their weekly goals on the go. In order to complete this implementation we would need to create our application in React Native instead of vanilla JavaScript. The benefits of this would be simpler distribution of the game, as well as access to another user group.

7.4 Backend Development & Storage

Another feature that we would like to implement is a user login system that would allow users to save their progress and create weekly goals. This would allow users to keep track of their progress and hold themselves accountable with the time they have to get their tasks done. When each user has a account they can then interact with other users and give their current inventory to their friends. Friends can hold each other accountable and ensure that their friend group is maintaining their progress. We would then also be able to provide feedback to the user in the form of charts and graphs, similar to a stats page that the other Pomodoro apps offer.

If user progress was persistent, we could also implement online trading similar to the Steam Community Market [3]. Since we are aiming to avoid addicting monetization methods, we would likely forgo the use of real money and instead only allow users to barter items or use their in-game currency. The introduction of real money would give us perverse incentives to attempt to "hook" players into the game and rely on so-called "whales" to generate large amounts of revenue from a small percentage of the player base [13]. Additionally, it is possible we would run into legal issues with the trademark holder of the Pomodoro Technique if we began monetizing Rooted.

8. EVALUATION PROCEDURE

Our team thinks that testing is valuable for our game because we aim to appeal to a wide population of users and seek to make a product that actually solves the two problems we have identified. Running user studies to test usability of our project would give us better understanding of the interactions we designed to influence future updates and improvements. As the game currently stands, it is more of a high-fidelity prototype than a polished final product. If we were to continue development, we would need to figure out areas of the interaction design to focus on.

We would use user testing to determine future improvements and current performance of our game by using studies to get evidence of strengths and shortcomings in our current design. In order to do this, we would design the studies in a way that isolates specific variables in the design. We would test different versions against our current (baseline) version, as well as other competitor Pomodoro apps and idle games.

One of the most important features of the game is the rate at which the game progresses. This rate includes visitor spawn rate and how the user gains and spends money. This feature is important to test because if the rate is too slow, it will not motivate people to continue playing the game because there is not enough reward for the amount of effort they put in. If it is too fast, the game itself will not be able to keep up with the user because they will run out of new items quickly, further prompting the user not to play anymore.

The subjects we would focus on the most would be university-age students because they are under the most stress for studying. We would consider various educational background factors such as year of study and area of study. It would be useful to search for users that have specific experience with Pomodoro Apps or the Pomodoro Technique in general. Since we are at Texas A&M, there would likely be enough students nearby to conduct the studies locally.

8.1 Gathering Quantitative Data

The easiest way to gather lots of qualitative data points on user experience would be to host a live version of the game online and attach a survey for users to submit. The surveys could give information about appeal, user demographics, and level of interest. Additional quantitative data could be gathered by observation of subjects in natural environments using the game to study. While subjects would have to be informed beforehand, there could be a way to run field tests in environments such as private study rooms to observe the effect of common distractions on users of game.

In our observation of users in field tests, we would aim to find frequent users of the Evans Library, Annex, and SCC. Although this means that there would be less control over environment variables, we could get a much better understanding of how common everyday distractions affect the performance of our game as a productivity tool. Ideally, there would be no setup needed. The observers could arrive at the same time as the users in the study settings and record their interactions using the game to help study.

8.2 Gathering Qualitative Data

If we were to publish our game online, social media could be a good way to gather qualitative data, using open codes to break down tweets and other posts. Our team could establish a social media presence for Rooted, simplifying the gathering of reviews and user testimonies.

There are also sites which may host our game for wider audiences, such as itch.io and html5games.com. We could gather comments from users and analyze them as well.

From gathered qualitative data, we would seek to understand suggestions and feedback not reached through our quantitative data gathering. We would use that understanding to improve the usability and the gameplay of Rooted.

9. ACKNOWLEDGMENTS

Our thanks to Dr. Richard Furuta for providing project feedback and guidance during the development processes. Also to Aaron Lee for being an invested and knowledgeable TA throughout the semester. Additional thanks to services GitHub and Google Docs for allowing easy collaboration on this project.

10. REFERENCES

[1] Addictions.com. 2019. A New Addiction On The Rise: Mobile Game Addiction. Retrieved from https://www.addictions.com/blog/a-new-addiction-on-the-rise-mobile-game-addiction/.

[2] Cirillo, Francesco. 2019. The Pomodoro Technique. Retrieved from https://francescocirillo.com/pages/pomodoro-technique.

[3] Conditt, Jessica. 2012. Steam Community Market enables buying and selling with Steam Wallet. (December 2012). Retrieved from https://www.engadget.com/2012/12/12/steam-community-market-enables-buying-and-selling-with-steam-wal/.

[4] Elgan, Mike. 2017. Smartphones make people distracted and unproductive. (August 2017). Retrieved from https://www.computerworld.com/article/3215276/smartphones-make-people-distracted-and-unproductive.html.

[5] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, 47(6), 381-391. http://dx.doi.org/10.1037/h0055392.

[6] Hafer, T.J. 2018. The legal status of loot boxes around the world, and what's next in the debate. (October 2018). Retrieved from https://www.pcgamer.com/the-legal-status-of-loot-boxes-around-the-world-and-whats-next/.

[7] Kelly, Makena. 2019. How loot boxes hooked gamers and left regulators spinning. (February 2019). Retrieved from https://www.theverge.com/2019/2/19/18226852/loot-boxes-gaming-regulation-gambling-free-to-play.

[8] Kennedy, Sean. 2018. The 10 Best Pomodoro Timer Apps to Boost Your Productivity. (September 2018). Retrieved from https://zapier.com/blog/best-pomodoro-apps/.

[9] Kibble.net. 2016. What Are Idle/Incremental Games? (May 2016). Retrieved from https://www.kibble.net/blog/index.php/2016/05/23/what-are-idleincremental-games/.

[10] Sullivan, Bob. 2015. Memo to work martyrs: Long hours make you less productive. (January 2015). Retrieved from https://www.cnbc.com/2015/01/26/working-more-than-50-hours-makes-you-less-productive.html.

[11] Sultan A. Alharthi, Olaa Alsaedi, Zachary O. Toups, Joshua Tanenbaum, and Jessica Hammer. 2018. Playing to Wait: A Taxonomy of Idle Games. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA, Paper 621, 15 pages. DOI: https://doi.org/10.1145/3173574.3174195

[12] Tagliaferri, Lisa. 2016. How To Work with JSON in JavaScript. (December 2016). Retrieved from https://www.digitalocean.com/community/tutorials/how-to-work-with-json-in-javascript.

[13] Takahashi, Dean. 2014. Only 0.15 percent of mobile gamers account for 50 percent of all in-game revenue (exclusive). (February 2014). Retrieved from https://venturebeat.com/2014/02/26/only-0-15-of-mobile-gamers-account-for-50-percent-of-all-in-game-revenue-exclusive/.

[14] Xenos. 2018. CORS is Layered over HTTP. (July 24, 2018). Retrieved from https://security.stackexchange.com/a/190269.

About the authors:

Ledet Awano is a Junior Computer Science major at Texas A&M University. She is currently pursuing a business minor and plans to attain her MBA and work as an engineering manager within the next 4 years. This summer she will continue learning as a software engineering intern at Linkedin where she will be working on the Premium Subscriptions team in Sunnyvale, California.

Grace Fan is a Junior Computer Science major at Texas A&M University. She is currently pursuing an Art minor and hopes to work as a User Experience Designer after graduation. She has had previous experience working at Sandia National Laboratories and will continue her work experience at Credera this upcoming summer.

Brandon Neff is a graduating Computer Engineering major at Texas A&M University. After graduation, he will be working as a software engineer developing embedded software at Collins Aerospace in Cedar Rapids, Iowa where he previously interned. His hobbies and interests include playing tennis, listening to podcasts, and watching Star Trek.

Charles Wong is a Computer Science student at Texas A&M University. His coursework in Computer Science follows an emphasis in Human-Computer Interaction, as well as a Minor in Art (New-Media). He currently is searching for summer research opportunities in HCI and aims to enroll in a graduate program following his final 2 semesters of undergraduate study. During breaks from school, Charles resides in San Diego, California. (people.tamu.edu/~charleswong)