Unseeable

A Major Qualifying Project
submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science in
Computer Science
and in
Interactive Media and Game Development

Authors:
Isaiah Cochran (IMGD)
Alexander Horton (CS)
Drew Tisdelle (IMGD/CS)
Tommy Trieu (CS)

Advisors:
Dr. David Brown (CS)
Brian Moriarty (IMGD)
Ralph Sutter (IMGD)

26 April 2018

Abstract

We designed, developed and tested *Unseeable*, a Web-based game with the goal of reproducing the subjective experience of a colorblind person for non-colorblind players to increase their understanding of the daily challenges caused by this form of visual impairment. The game was developed using HTML5, JavaScript, CSS3 and Three.js to build the front end, node.js and express.js for the server, and MongoDB for the database used to store test data. A majority of players reported increased empathy for people dealing with colorblindness.

Acknowledgements

This project would not have been possible without our advisors, Prof. David Brown, Prof. Brian Moriarty, and Prof. Ralph Sutter. We would also like to thank Tim Jones, Cassandra Salafia, and Bailey Sostek for lending their voices.

Authorship

Abstract	**Horton**
Acknowledgements	**Horton**
Authorship	**All**
Table of Contents	**All**
List of Figures	**All**
List of Tables	**All**
1. Introduction	**Trieu**
2. Background	**Tisdelle, Trieu**
3. Concept	**Tisdelle**
4. Gameplay	**Horton**
5. Design	**Cochran, Tisdelle, Trieu**
6. Technical Implementation	**Horton, Tisdelle, Trieu**
7. Art	**Cochran**
8. Audio	**Tisdelle**
9. Research Methodology	**Trieu**
10. Results and Analysis	**Trieu**
11. Project Promotion	**Tisdelle**
12. Conclusion	**Trieu**
References	**All**
Appendices	**Trieu, Horton**

Table of Contents

List of Figures

1. Introduction

Unseeable is a first-person game that requires players to solve a series of color-based puzzles from the perspective of a colorblind individual. The game takes place in a three-dimensional environment, but the controls reflect that of a two-dimensional game. The player is asked to solve puzzles without any hints indicating the identification of colors. When the player makes a choice or completes a puzzle, their success is reflected by the feedback they receive from the environment.

This Major Qualifying Project (MQP) was completed by four Worcester Polytechnic Institute (WPI) students. The project's artist was Isaiah Cochran, majoring in Interactive Media and Game Development (IMGD). The project's three programmers were Alex Horton, majoring in Computer Science (CS), Drew Tisdelle, majoring in IMGD and CS, and Tommy Trieu, majoring in CS.

The goal of *Unseeable* was to allow non-colorblind individuals to empathize with colorblind persons by seeing through their perspective and understanding some of the struggles they endure in their everyday lives. The hope of the *Unseeable* team was that this experience would spread awareness about color blindness and allow those without visual impairments to understand the stress of those that suffer from them. As an IMGD and CS MQP, the developers had a secondary goal of creating a game engine from the bottom up. This goal was necessary in order to fulfill the technical aspect required for a CS MQP.

2. Background

2.1. Color Blindness

2.1.1. What Is Color Blindness?

Color blindness is an often-overlooked visual impairment that poses significant challenges for those afflicted with it. The disability affects approximately 8% of men and 0.5% of women throughout the world. It is believed to be caused by malfunctioning cones within the eyes of those who have it, resulting in them seeing colors in ways that can be very different from those with fully functional cones. Differences in vision include, but are not limited to: two different colors appearing to be the same and entire colors being absent from a person's vision. The condition is usually inherited genetically, but can be acquired through various diseases and old age.

(a) **(b)**

Figure 2.1: Normal Color Vision (a) and Red/Green Color Blindness (b)

2.1.2. Why Color Blindness?

The focus on color blindness for *Unseeable* came about because this impairment affects a significant percentage of the population, but not many people understand the struggles it entails. From both research and from personal experience provided by a colorblind team member, when someone who is unfamiliar with color blindness first finds out that a person is colorblind, they will often start pointing at random objects and ask the colorblind person what colors the objects are. When the colorblind person proceeds to incorrectly identify the colors, the person who asked them to do so will often see it as "cool", "interesting", or "funny." This has become known as "the color game" to those who are colorblind, and is something that most colorblind people hate.

There are also many situations in which, even after knowing a person is colorblind, people will fail to take the person's disability into account. This includes getting frustrated when colorblind persons try to describe something to them using colors, or joking about how a colorblind person cannot tell certain things apart, such as when the person tries to do something

as simple as pick out their favorite candy flavor, since most flavors are color-coded. These situations can and do elicit a lot of stress, and cause those with color blindness to feel excluded. It can also make them feel as if they have done something wrong by not being able to complete simple tasks due to their inability to see in full color.

It is because of this lack of understanding of what color blindness actually is and how it affects those afflicted that we decided it would be an appropriate topic to try to shed more light on through a game.

2.1.3. Additional Information

While there are many different forms of color blindness, the one that is most common is red/green color blindness, in which different hues are very hard to distinguish between. Normally a person can see seven different hues of colors, but a person with this form of color blindness is limited to seeing only two or three, depending on the severity. It is because this form of color blindness is one of the most common forms, and the form that the group is most familiar with, that it was chosen to be the form of color blindness simulated within the game.

2.2. Empathy

Empathy can be defined as the ability to understand and share the feelings of another individual. This definition, however, only touches the surface of what empathy is. Empathy can be split into multiple different subcategories of itself, each with its own unique definition. The primary types of empathy that are usually examined are situational, dispositional, cognitive, and emotional. (Stueber, 2013; Lawrence et al., 2004)

Situational empathy encompasses the empathic reactions of someone in a specific situation. This type of empathy is displayed in reaction to specific events. Dispositional empathy is the opposite: dispositional empathy is empathy as a person's characteristic trait, meaning how empathic someone is at any time.

Besides types that cover when a person feels empathy, there are other sub-categories that define how an individual can be empathetic towards another. Cognitive empathy is an example of this: the understanding of someone else's feelings/mental state. Someone experiencing cognitive empathy will try to comprehend how another person is feeling and try to take on their perspective. Emotional empathy, however, is an emotional response to someone else's feelings. This includes either feeling the same emotion as someone else experiences it (i.e. getting angry when someone else gets angry), or going beyond that with the same emotion at a much higher intensity. While all of these definitions are unique, they also all fall within the overall definition of empathy.

Attempting to measure empathy has proven to be a unique challenge. While understanding what empathy and its sub-definitions are can be fairly straightforward, attempting to quantify empathetic feelings can be difficult. This is due to the fact that there are no clear, quantifiable items corresponding to empathy that are easy to measure.

Existing attempts at measuring dispositional empathy include subjective questionnaires. (Lawrence et al., 2004) A large suite of these questionnaires or scales exist and have been

used in empathy-related studies. The most well-known of these scales include: the Hogan Empathy scale, Davis' Interpersonal Reactivity Index, the Toronto Empathy Questionnaire, the Empathy Quotient, the Questionnaire Measure of Emotional Empathy, and the Balanced Emotional Empathy Scale.

While all of these scales are self-reports that ask how someone would react in specific situations, they differ in the way the data is scored and analyzed. For example, the Toronto Empathy Questionnaire produces a singular score rating a person's overall self-reported dispositional empathy. (Spreng et al., 2009) Davis' Interpersonal Reactivity Index, however, produces four different scores that representing fantasy, perspective-taking, empathic concern, and personal distress. (Davis, 1980, 1983) Despite these differences, all of these scales share the same limiting factor of only measuring dispositional empathy.

Measuring situational empathy is much more appropriate to seeing how an audience would respond to a specific situation. Scales like the Positive Affect and Negative Affect Scale are good for measuring differences in mood after exposure to a potential stressor, but is still limiting due to the fact that it measures mood, not empathy, and is also a self-report measure. (Watson et al., 1989)

The most accurate readings on situational empathy involve measuring real-time, physiological responses that correlate with empathy after being exposed to a specific situation. This can be accomplished through measures such as: facial response, vocal responses, gestural responses, heart rate (e.g. beats per minute), or skin conductance. (Stueber, 2013) The fallback of this approach is the difficulty in interacting with a participant and gathering these metrics. Not only do the expenses for gathering some of these metrics become rather high, but the convenience of taking these metrics can be rather low, since a participant would have to be connected to multiple devices in order to measure items such as skin conductance or heart rate. Facial, vocal, and gestural responses require a single camera and microphone with recording functionality, but analyzing this data requires specialized expertise.

Clearly, while situational empathy can be very accurate in measuring empathetic concern in a specific situation, it is much harder to implement than the questionnaires used to measure dispositional empathy.

3. Gameplay

The game that was developed in this project, *Unseeable,* is played through the perspective of a child named Sam, who is afflicted with red-green colorblindness. The game uses an algorithmic display filter to render full-color scenes in a limited spectrum similar to that experienced by a colorblind person. The game is split into 2 levels, one in a kindergarten classroom and another in a child's bedroom. All interactions in the game are controlled with the mouse only.

3.1. Level 1

3.1.1. Player Goals

The first level takes place in a kindergarten classroom. The level includes three tasks that the player must complete in order to proceed to the next level. The first task is to find the red table, the second task is to complete a color-by-numbers activity, and the last task is to place their artwork on the board.

3.1.1.1. Selecting the Table

The first task that the player has to complete is to select the red table. Using the mouse, they have the options to either pick a table by clicking on one, or to look around the classroom by moving the mouse to the edge of the screen. There are four tables in the classroom that the player is able to choose from, and they have three chances to pick the correct one. Figure 3.1 shows the view during this task. In this figure, the player has the ability to select a table by clicking on it or look around the classroom by moving the mouse to either the left or right edge of the screen. When they click on a table, the camera will walk over to it. If the table that they selected is not the red table, they are told by the teacher that they are not at the correct table and will have another chance to pick the red table. After three wrong attempts, the teacher will guide the player over to the correct table and they will be seated.

Figure 3.1: Finding the Red Table

3.1.1.2. Coloring the Paper

Once the player is seated at the correct table, the teacher will instruct the class to color their papers, which are color-by-numbers. The camera will pan down to look at the table in front of the player. The camera angle here is fixed, so the player will not be able to look around the classroom at all during this activity. The view is shown in Figure 3.2. On the left of the screen is the coloring paper and on the right of the screen is the box of crayons. There are 7 different colors of crayons that the player can choose from and they have to use these crayons to color 14 sections of paper to complete this task. The player has the ability to select and pick up crayons by clicking on them. Once a crayon is selected, it will follow the mouse cursor. With a crayon selected, the player has the ability to color a section of the paper or return the crayon to the crayon box. If the player hovers the crayon over the paper, the section of paper is highlighted by changing to a light version of the selected crayon color, and clicking will color the paper the same color as the selected crayon. If the player clicks on the crayon box while they have a selected crayon, it will be returned to the box. During this activity, the player will not receive feedback about whether or not they have used the correct color crayon. Once they have colored all 14 sections of the paper, the next cutscene will start. The camera will pan to the teacher as she instructs the class to put their papers on the board. The screen will fade to black and then unfade in front of the board.

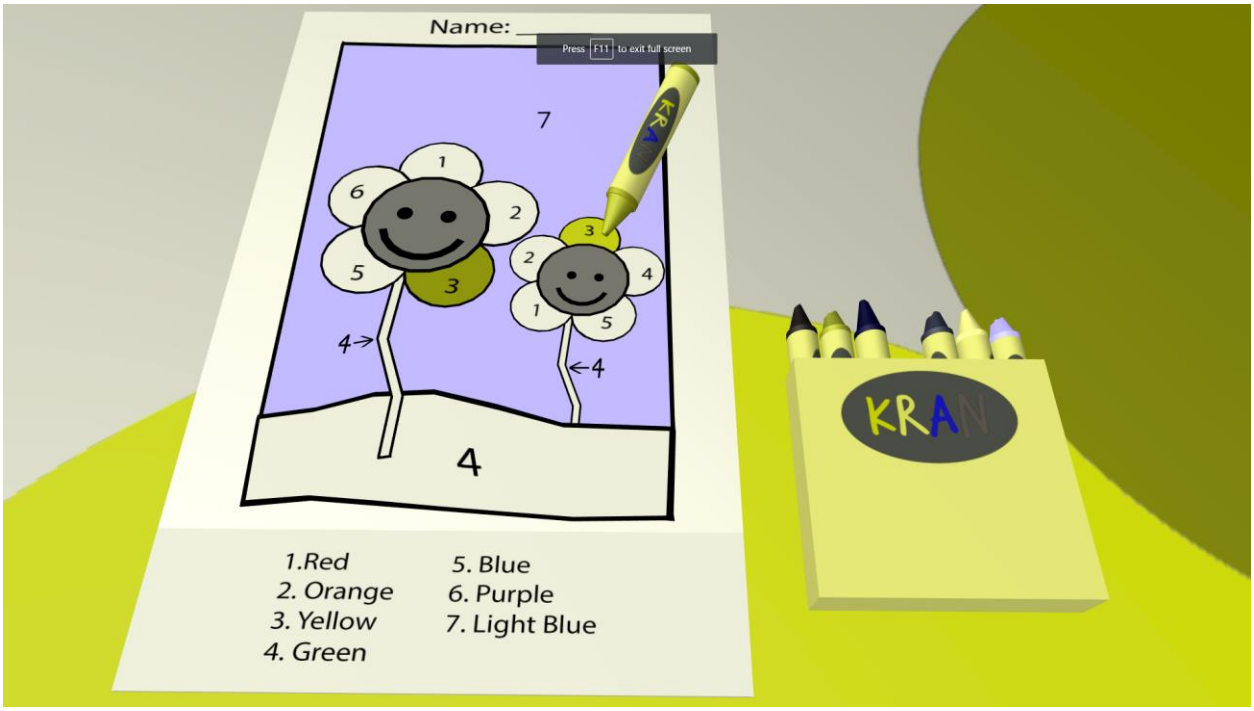


Figure 3.2: Coloring the Paper

3.1.1.3. Posting the Paper

Once the screen fades, the player will be located in front of the board. The paper that they just colored in the previous activity will be located at the mouse position on the board. Moving the mouse will move the paper on the board. When the player clicks the mouse, the paper will be placed on the board and the camera will zoom out, showing the papers of the rest of the class on the board.

Depending on how well the player did in the coloring activity, they will receive one of three endings. If they colored less than 10 sections of the paper correctly, they get the worst ending, where the children in the classroom make fun of the player where one child exclaims “Sam doesn’t know how to color!” and the rest of the children laugh. If they colored between 10 and 13 sections correctly, they get the ending where a child in the classroom says that some of the colors look a little bit weird. If they correctly color all 14 sections, they are not mocked at all and the level ends.

Figure 3.3: Posting the Paper

3.1.1.4. End of Level

When the player completes the level, the colorblind filter fades out showing the true colors and the end of level menu screen fades in. The end of level menu displays the amount of sections that the player colored correctly and buttons for the different choices that the player can make once they have completed the level. The options that they have here are to continue to the second level, replay the level in colorblind mode, replay the level with normal vision, and to return to the main menu.

3.2. Level 2

3.2.1. Player Goals

The second level takes place in a child's bedroom. The player's goal in this level is to build a block house without their friend getting annoyed at them for messing up too many times and ending the level before the house is finished. Playing on the floor of the bedroom is a child playing with blocks who invites the player over to play with them. The player will then walk over to a pile of blocks already on the floor and sit down to play with them. The cutscene ends here and the player will then have the ability to start interacting with the blocks and the instructions that have been laid out on the bedroom carpet.

3.2.1.1. Building the Blocko House

Once the cutscene is over, the player is seated right by the puzzle. Here, the player has the ability to pick up the first page of instructions or to pick up a block. If the player clicks on the instructions, the instructions will be moved in front of the camera showing the current building step in better detail. The instructions show what step the player is on, the block and color that they should pick and where they should be placed. Clicking anywhere on the screen with the instructions in front of the camera will return the instructions to their original position.

Similar to the crayons in the first level, hovering over a blocko will change the mouse cursor and raise the blocko slightly. Clicking on it selects it and it will follow the mouse around. The ghosts of the block in both the original position and where the block has to be placed to build the house will become visible to show the player that the block could be placed in either of these locations. When the block is moved close to one of it's ghosts that ghost will become more opaque to signify that clicking will place the block at that location. Clicking with the block near its ghost at its original position will deselect it and return it to its original position. Clicking with the block near its ghost at the final building position will attempt to place it there. If the block is not the correct shape or color to be placed, the friend will scold the player.

The player can make 3 incorrect block attempts or 5 incorrect color attempts before the friend gets angry and ends the level. After the first incorrect color attempt, the friend will point out which is the correct block, but will provide no more help after this. If the block is both the correct shape and color, it is placed in the final position and the top page of the instructions is pulled offscreen showing the next step for building the house. There are 10 blocks that have to be placed and 3 possible endings for this level. The first ending is where the friend gets frustrated with the player and ends the level before the house is finished being built. The other two endings are reached once the house is finished. If the player made 0-2 mistakes, then the friend will remark that the house looks good and if they made 3-4 mistakes, then they will remark that it took some effort to finish the house.

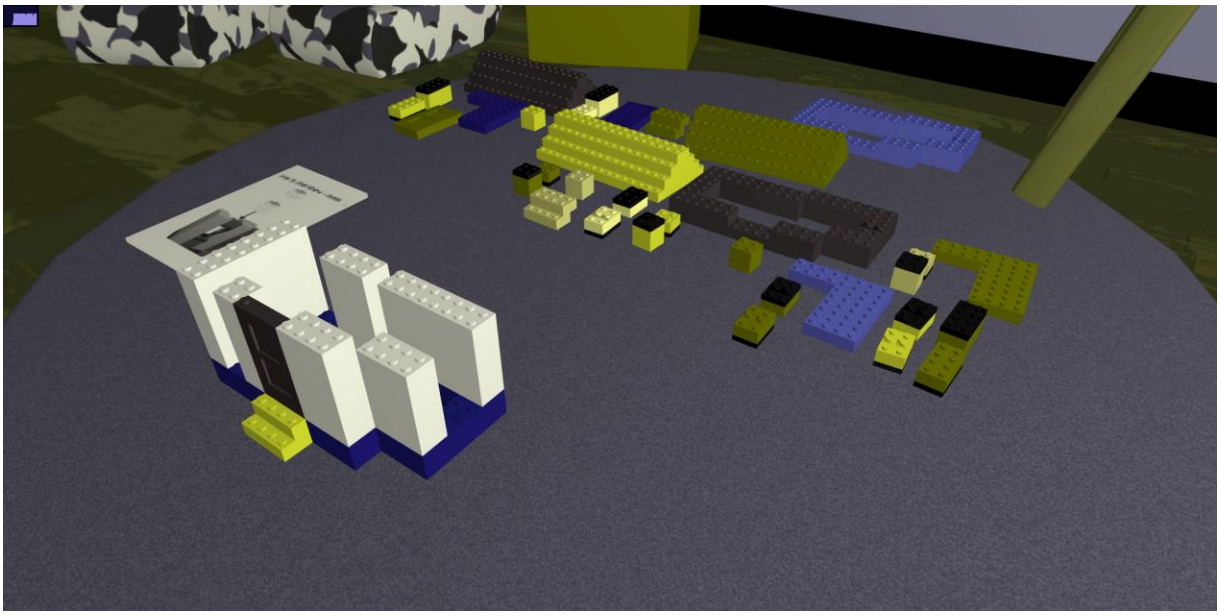


Figure 3.4: Building the Blocko House

3.2.1.2. End of Level

When the player reaches an ending, the end of level menu will be displayed as for the first level. This menu will display how many of the blocks they placed and the options that they can select at the end of the level. The player will have the same options as the first level in that they are able to replay the level with or without colorblind mode on and are able to return to the main menu. There is no next level option here because this is the final level.

4. Design

4.1. User Analysis

As mentioned before, the goal of *Unseeable* was to allow non-colorblind individuals to empathize with colorblind persons by seeing through their perspective and understanding some of the struggles they endure in their everyday lives. While anyone could play the game, this put some limitations on the types of users being considered during data collection and analysis.

In order to discern whether or not the game caused empathy towards the colorblind population, a differentiation had to be made between those with the vision impairment and those without. Non-colorblind individuals were considered as necessary participants in order to examine their stress and empathy towards the colorblind during and after gameplay. Additionally, it was decided that users would be college students of any gender in the United States. College students were the preferred population because it guaranteed that the participant had, at a minimum, a high school level education. This minimum level of education within the United States assumes that the participant had basic reading and English speaking skills, which was important when listening to the audio instructions paired with subtitles in the game. It was also assumed that at that education level the participant would have experience using a computer with a mouse and keyboard, which were the instruments required to play the game.

Narrowing the users to college students made it easier for the team to promote the game by allowing them to send the link to the study to their peers. There was no restriction on gender because the goal of the study did not focus on whether different sexes would display different amounts of empathy towards colorblind persons.

4.2. Task Analysis

The tasks that players must complete throughout the game were designed around tasks that appear to be normal, simple tasks to those with normal color vision but become difficult challenges to those with color blindness. There are many situations that this applies to, but it was decided that the tasks that should be chosen are those in which the player is able to get feedback from non-playable characters. The reason for this is that the feedback from the non-playable characters on the tasks the player completes would have a much larger emotional impact on the player than if they were placed in a situation where they were alone.

The first task that was decided on was one that would involve coloring. Due to coloring mostly taking place at a young age, it was decided that the player would be a small child. In addition, during this time in a color blind person’s life the person and those around them are often unaware that they are colorblind. This helped to make the scenario have a larger emotional impact as the children at this age would be more likely to tease the color blind person who they believe simply does not know their colors, rather than trying to understand why the

color blind person did not get the colors correct. The teasing the children would do in this case would help to show how embarrassed and singled out a colorblind person can feel, which is why the task was chosen for the game.

The second task that was decided on was one that would show how those who do not understand color blindness can get easily frustrated with a colorblind person and can make a colorblind person feel terrible for something that they cannot control. In order to do this, it was decided that a scenario was needed in which the player character would need to complete a task that a non-colorblind person would find simple with at least one other non-playable character who could become frustrated with the player. It was decided that building with Legos, called Blockos in our game, would be an ideal scenario. This is because building with these blocks is a simple task, but a colorblind person would not be able to tell the different colored pieces apart. When building with a person who does not understand color blindness, the person can easily think that the colorblind person is placing the wrong pieces on purpose and get very frustrated that such a simple task is being done incorrectly multiple times. It is because this scenario is one that is quite common for those who are color blind and because it would be able to display the emotions we wanted players to experience that we chose this task for the player to complete.

4.3. User Interface

Throughout the development process, the user interface for *Unseeable* has undergone many changes in order to make it easier for players to easily play and understand the game with minimal information provided before starting the game. In order to successfully do this the user interface was divided into five separate parts: the Main Menu, the Level 1 interface, the Level 2 interface, the End Level Menu, and the Credits Screen.

4.3.1. Main Menu

The original design for the main menu, pictured below in Figure 4.1, displayed the first level of the game with half of the screen in full color vision and the second half displayed through a colorblind filter. The thought behind this design was that it would show players the stark contrast between how those with colorblindness see the world when compared to how those without the vision impairment see it. At the bottom of the screen the player was given the choices to view the Credits, Replays, and Options of the game. Players were also given the options to start the game by pressing Play and to close the game by pressing Exit.

However, this design was not the one that was chosen for the final version of the game. The final design, pictured below in Figure 4.2, was created due to the original design conflicting with the goal of the game and due to features being cut throughout the development process. The first key difference of the new main menu design is that the screen is now fully displayed through the colorblindness filter. This was done because players would be able to see the actual colors of some of the tables in the first level, which would have given players the answer to the first puzzle in the first level. In addition, the change was made so that players would not know

how different the colors would be until the end of the first level, creating a larger emotional impact when the true colors are revealed after struggling through the level.

The other changes that can be seen in the final version of the title screen below include the removal of the Replays, Options, Credits, and Exit buttons. The Replay button would have taken players to the menu where they could view replays of their playthroughs of levels of the game. However, it was determined that adding in this feature would not add much to the game and would take a substantial amount of time to implement and therefore the idea was dropped and button removed. The Options button would have brought up a menu where players could change in-game options, namely the size of the screen. However, due to the game being browser based, players only would need to resize the browser window to change the screen size, removing the need for an options menu altogether and thus resulting the removal of the Options button. The Credits button originally took players to a screen that showed the names of the project group and advisors on it. This option was given only because the game did not have an ending to display the credits after during this stage in development. Once an ending was added to the game, the credits could be placed at the end of the game removing the need for the Credits button. The final button, the Exit button, would have taken players back to the website the game was going to be hosted on. However, due to the website not being implemented due to time constraints, the Exit button was no longer needed as players could simply close the tab the game was running in to exit the game.

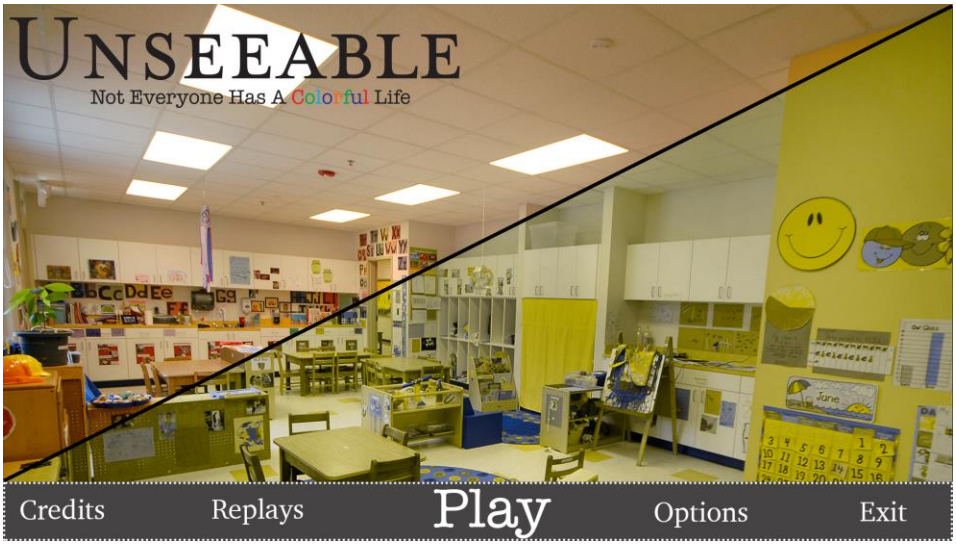


Figure 4.1: Original Design

Figure 4.2: Final Design

4.3.2. Loading Screen

Throughout most of the development process, loading screens were not part of the project and were not even planned to be in the game. However, as more assets were added into the game over time, it began to take longer to load each level. The result was the first few seconds of the level containing a black screen with individual assets suddenly appearing until all had been fully loaded in. While this was fine during development, it was seen as very off putting to eventual players of the game. This resulted in the idea that we should have a loading screen put in the game so that players would not be able to see the assets as they load into the level.

The loading screen, shown below in Figure 4.3, displays a simple, crayon background to reflect the challenge that will be faced in the first level of the game. On top of the background, a grey loading bar is displayed which indicates the percentage of assets that have been loaded into the game. As more assets are loaded, the percentage increases and the bar fills up until it is completely yellow instead of grey. After the bar is fully loaded, a continue button is displayed that the player can click in order to start the level. In addition, the loading screen contains a short summary of the level in order to provide the player with context about the situation that they will be put in before the level starts.

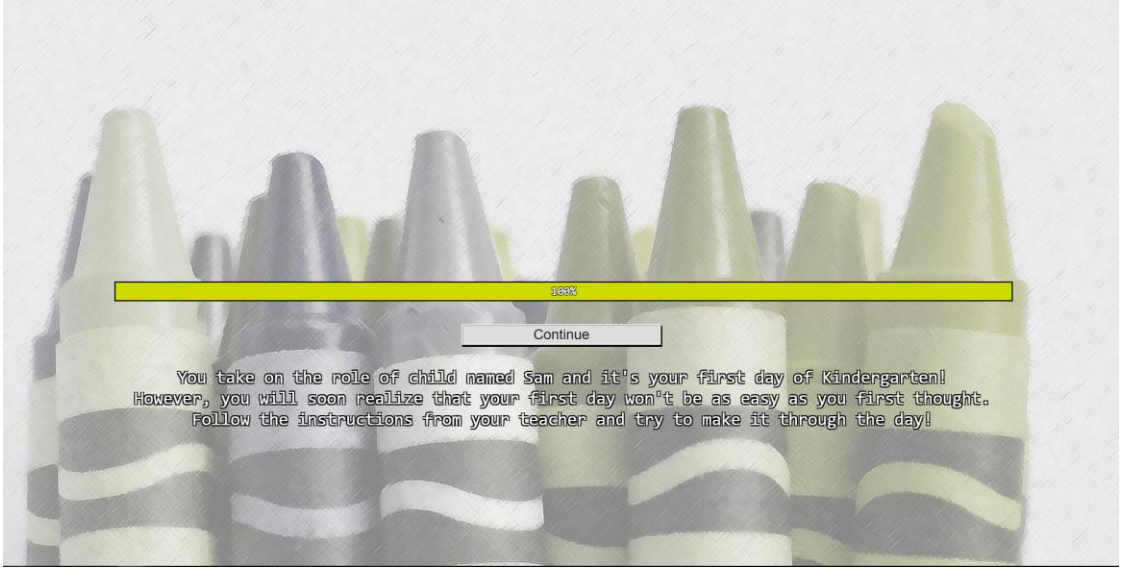


Figure 4.3: Loading Screen

4.3.3. Level 1 Interface

The design of the interface for Level 1 changed greatly over time in order to make the game easier to play. Originally, the title interface consisted purely of the scene in front of the player as well as the default mouse being displayed in order to interact with the screen. This simple interface did not explain to players how to interact with the game. Once the basic mechanics of the game were implemented, it was decided that the interface needed to be changed in order to provide the player with additional information about the level.

The first change that was made to the interface was the addition of a letterbox format (black bars at the top and bottom of the screen) to the interface in order to indicate to the player that a cutscene is currently taking place. This change made it apparent that the player could not currently interact with the game due to the cutscene that was taking place.

The second change that was made to the interface was the addition of vertical, transparent black bars with arrows on the left and right side of the screen that would appear after the first cutscene was complete. These were put in place in order to indicate to the player that, if they move their mouse to the edges of the left and right sides of the screen, they could rotate the camera view in order to look around the room. Once the player tried rotating the screen for the first time, the bars would fade away so that the player could focus on the game. Moving the mouse to the edge of the screen would make the bar visible again as they rotated the camera.

The third change that was made to the interface was to display subtitles at the bottom of the screen. The reason for this is that audio instructions are given to the player through the teacher, but these instructions could potentially be misheard or the player may not be able to hear it for a number of reasons. In order to prevent the player being unable to receive instructions due to these reasons, it was decided the best option would be to have subtitles

displayed so that the player could also read the instructions in addition to hearing them. The subtitles were placed at the bottom, center of the screen due to this being the standard for subtitles in film and games.

Figure 4.4: Letterbox and Subtitles

The fourth change that was made to the interface was to briefly display the current objective at the top, center of the screen. This was done in order to make sure that the player understood what task they were supposed to be doing in the game. It was placed at the top, center of the screen due to this being a standard among games. After a few seconds, the objective would fade away so that the player could focus on the game.

Figure 4.5: Side Bars and Current Objective

The fifth change that was made to the interface was how the player interacted with the crayons in the coloring section of the level. Originally, the player received no indication that they could pick up a crayon, could place a crayon back, or that they had picked up a crayon. In order to remedy this, when the mouse is over a crayon it will stick up out of the box slightly and the mouse will change to a selection cursor to indicate that it can be selected. Once the crayon is selected, the cursor disappear and the crayon will move to the cursor's position in order to indicate to the player that selected a crayon and that they are currently able to color using it. Finally, when the crayon is moved over the crayon box, a transparent version of the crayon will display in the spot where it once was to indicate that it can be returned to the box. This helped to indicate to the player how they could interact with the crayons during this part of the level.

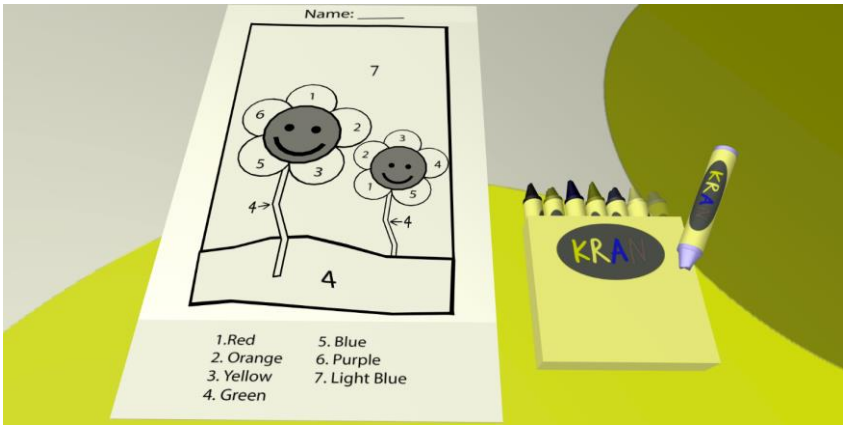


Figure 4.6: Crayon Cursor and Placing Back Crayon

The final change that was made to this interface was that, when the player hovers their crayon over an uncolored section of the paper, the paper will turn a lighter shade of the crayon color to indicate that the player can color that section of the paper. This allowed the player to more easily see which section they were currently over so that they could accurately color the section of the paper that they wanted.

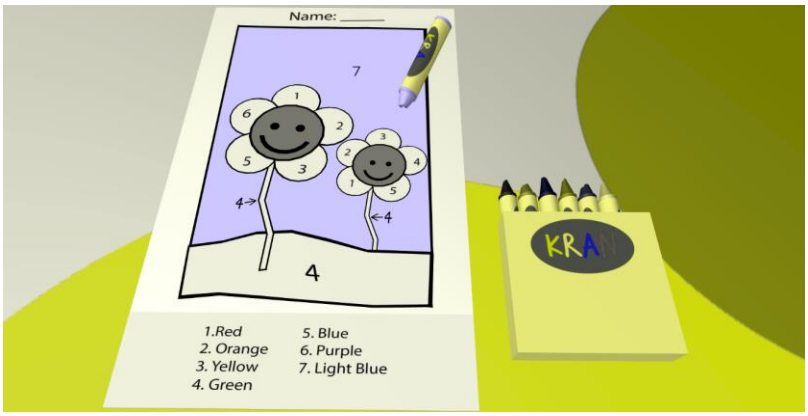


Figure 4.7: Color Selection

Other than the changes that were made to the interface overtime, the final section of the interface was kept relatively the same throughout development. This last section displays a white board along with the player's colored sheet, which becomes their cursor. Once the player places the drawing, the camera will zoom out to show the entire board containing all the drawings. The interface here was designed to have a large impact on the player as it is the point in the game where they will realize just how different their colored sheet looks from all the others.

Figure 4.8: Paper Placement

4.3.4. Level 2 Interface

The interface of Level 2 is very similar to the interface from Level 1. When a cutscene is taking place, the interface will also display black bars in a letterbox format in order to indicate that the current part of the game is a cutscene. Subtitles and level objectives will also display on the screen in the same locations as they did in the previous level.

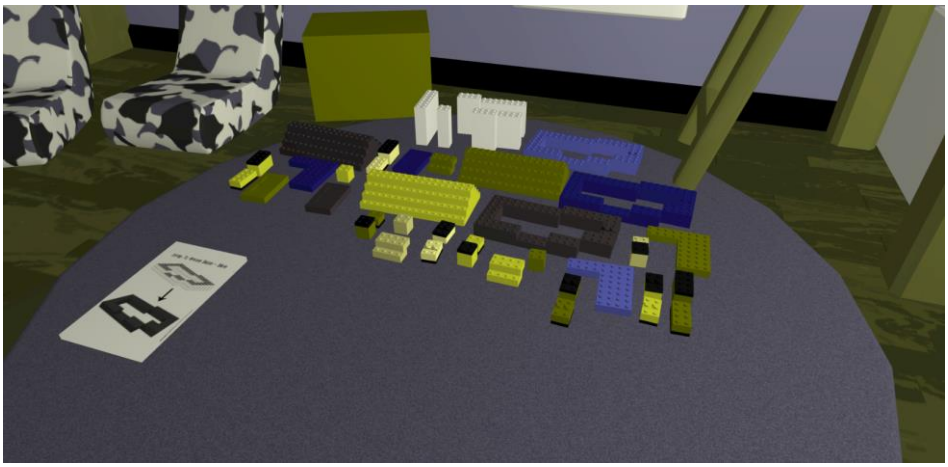


Figure 4.9: Blocko Building Area

The first new interface element of Level 2 is that the player is able to click on the instructions that are lying on the ground. When this is done, the top sheet will come forward and be displayed in the center of the screen showing the player the instructions they need to follow in order to advance in the level. When clicked again, the instruction sheet will return to its previous location.

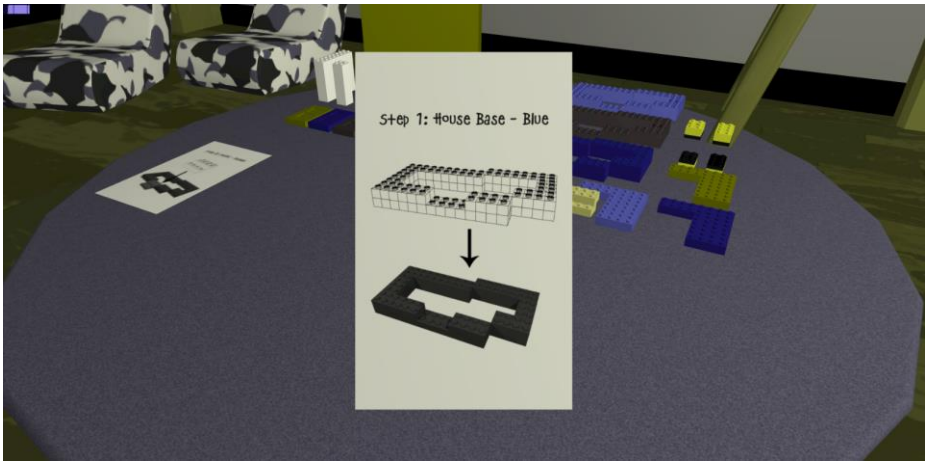


Figure 4.10: Building Instructions

The second new interface element is similar to the selection of crayons in Level 1. When the player hovers their cursor over a blocko, the blocko will shift upwards and the cursor will change into a selection cursor to indicate that the blocko can be picked up.

The third new interface element is that, when the blocko is moved close to the building area or to the place where it was picked up from, a transparent version of that blocko will appear in that space to indicate to the player that they can place it down in that location. This allows the player to easily see how they can interact with the blockos in the game.

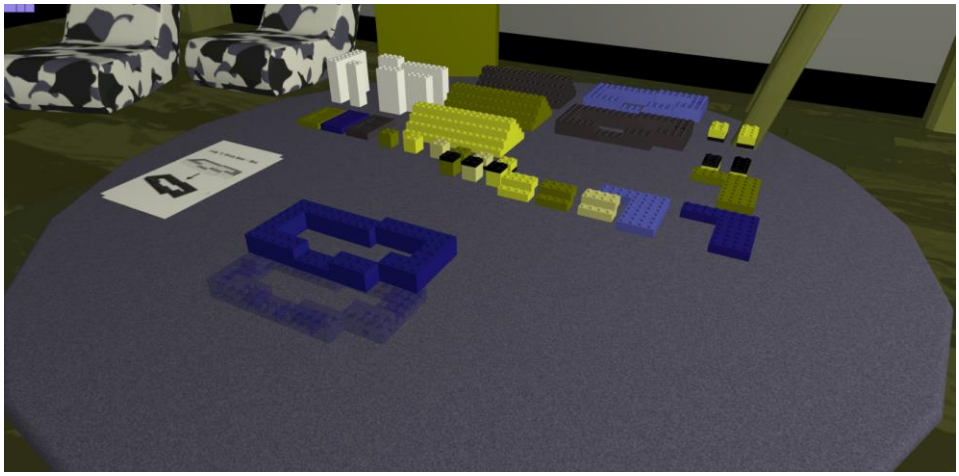


Figure 4.11: Transparent Blocko During Placement

The final new interface element is that, when the player makes their first mistake, the correct blocko will be highlighted to indicate which one was supposed to be chosen. This was done simply to give the player one free chance at getting the correct answer when placing pieces.

4.3.5. End Level Screen

The end level screen is a simple interface that is displayed at the end of each level. When the final cutscene for each level finishes, the screen will become blurred and a text menu will appear on the screen. The reason that the image is blurred for this interface is to make the text easier to read for the player.

When the menu is displayed, the player will have the option to retry the the level, retry the level with normal color vision, continue to the next level, or to return to the main menu. In addition to these options, there are two main differences between the Level 1 and Level 2 versions of the End Level Screen. The first difference is that the Level 1 version will display the number of correct colors on the coloring sheet that the player managed to get while the Level 2 version will instead display the number of steps in building the blocko house that the player managed to get through before the level ended. The second difference is that the Level 2 version contains a Continue option rather than a Next Level option. The reason for this is that Level 2 is the final level and the Continue option will instead take them to a black screen that says “Thank you for playing our game!”

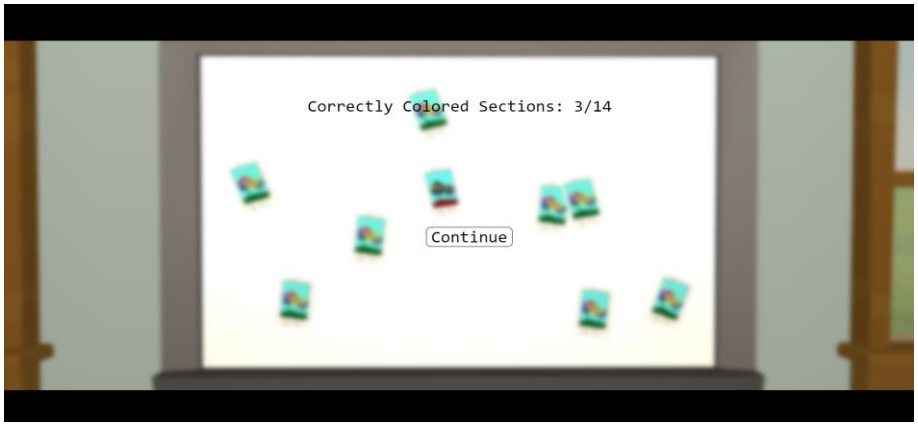


Figure 4.12(a): End Level Screen (Level 1 Version)

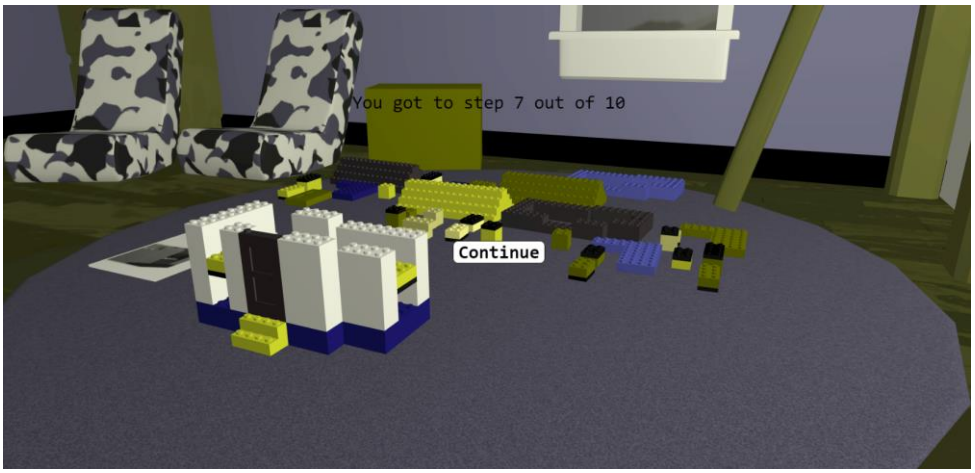


Figure 4.12(b): End Level Screen (Level 2 Version)

4.4. Art Style

Due to the fact that *Unseeable* was created in a custom game engine and was intended to run in a web browser, the art style we chose was one of simplicity with a cartoon-like style. This was chosen because we wanted the game to run quickly on lots of computers. For that reason, it could not contain excessive amounts of geometry and textures. As a consequence, the art was highly influenced by the game *That Dragon Cancer,* in which the players do not have faces and the objects which populate the scenes are very simplistic. In figure 4.13, the left image comes from *That Dragon Cancer* and the right is from *Unseeable*.

Figure 4.13: *That Dragon Cancer* (Left) and *Unseeable* (Right)

We also chose this style so it did not detract from the story being told. *That Dragon Cancer* is a very serious game, and its narrative conveys the message very effectively without the art style being the main focus, which is why we chose the minimal look for *Unseeable.*

4.4.1. Characters

Each character in *Unseeable* was created with the aim of simplicity and diversity. Since we had planned for there to be several characters, the characters were made in a way that would allow for them to be produced in various different versions rather quickly. We created two different body types and several different skin tones, as well as clothing colors and hairstyles. Each character has no distinguishable facial features, which allowed us to mix and match the pre-created character assets to create completely different characters quickly. A few character examples can be seen in figure 4.14. The clothing choices were meant to be colorful, as children tend to wear brightly colored clothing and we wanted there to be various different colors in a game about color blindness. The only character that is slightly different from the others is the teacher, seen in the first level. This character, seen in figure 4.14, does not share the same body model as the child models, but is in the same style as the other characters.

Figure 4.14: Character Models

4.5. Level Design

The design of the levels, very similar to the rest of development, was an iterative process. Various layouts, color schemes, and functionalities were considered and implemented over the four term production period; each iteration consciously tweaked in order to ensure our experience goal, producing a measurable amount of empathy, was achieved. We aimed to produce a realistic narrative in order to achieve this, therefore each level was designed to be grounded in reality, with stylistic choices that do not detract from the believability of each scenario.

4.5.1. Level 1

The first level underwent various iterations, mostly based on level layout and color choice in order to maximize the believability of each scenario. Each iteration was aimed at achieving our experience goal of creating situations designed for the player to feel empathy towards those with color blindness. This level is intended to make the player experience the confusion and potential humiliation a child with color blindness would experience in an early childhood classroom setting.

The first interaction with the level the player has is to seat themselves at a table of a particular color, as many American preschools and kindergarten classrooms have colored furniture items. The correct table for the player to choose is the red table. Red was chosen due to the fact that, in colorblind mode, it appears very yellow in hue. Yellow remains yellow in colorblind mode, making it a 50/50 chance of the player selecting the correct table if they have deduced that the correct one must be amongst the similarly colored tables. Blue and green were chosen as the other colors; green appears gray, and blue stays the same. This was done under the assumption that the player, before playing the game, had limited knowledge of the colorblind experience and would not have any rational basis to believe that either could not be red.

Feedback from initial playtesting showed that about forty-seven percent of players could not tell which of the four tables was the correct one on their first try. Originally, it was planned to have the first table the player selected to always be considered incorrect in order to ensure the experience goal of empathy. However, once playtesting data showed most players could not select the red table first, we allowed for the chance of selecting the correct table on the first try. If the player chooses correctly on the first try, they still experience the confusion when faced with having to choose the red table when apparently there isn't one on the screen, which is part of our experience goal.

Each table top surface has a polka dot pattern in a color that corresponds to the solid color which is seen on the sides of each table, as seen in figure 4.15.

Figure 4.15: Classroom Table

At the height of the character, each tabletop cannot be seen, so the player must rely upon the solid color on the table's side when faced with selecting the correctly colored table. It is for this reason that each chair surrounding the tables share the table's color. The tables have been laid out in a semi circle, in the center of which the player stands when tasked with making a selection. This was done in order allow for the most unobstructed view of each table that could be allowed without requiring to turn the camera to a different view, as seen in figure 4.16.

Figure 4.16: Initial Camera Position

In the early iterations of the game, seen in figure 4.17, the tables were laid out in a completely different manner, which didn't allow the player to see all of the tables clearly and would make gameplay unnecessarily difficult.

Figure 4.17: Original Table Layout

When the player clicks on a table, the camera moves toward that table. If the table is the correct one, they will sit down in the seat, allowing for the next cutscene to start. If they choose

the incorrect table, they are told verbally that the table they chose is incorrect, and will have to choose another. The camera then turns to allow all the tables to be seen which allows the player to choose another table. If the player selects the wrong table three times, they are then led to the correct table, as there are only four table options.

After the player is seated at the correct table, the camera pans to the teacher who explains the second task: to fill in a color by numbers sheet. Color by numbers is an activity that is often utilized in early childhood education, which is why it was chosen to represent the experience of those with colorblindness at a young age. Once the player receives verbal instructions from the teacher, the camera view shifts to view the workspace area, which contains various crayons in a crayon box, and a sheet of paper which contains a scene of two flowers. This can be seen in regular color mode in figure 4.18 and in colorblind mode in figure 4.19.

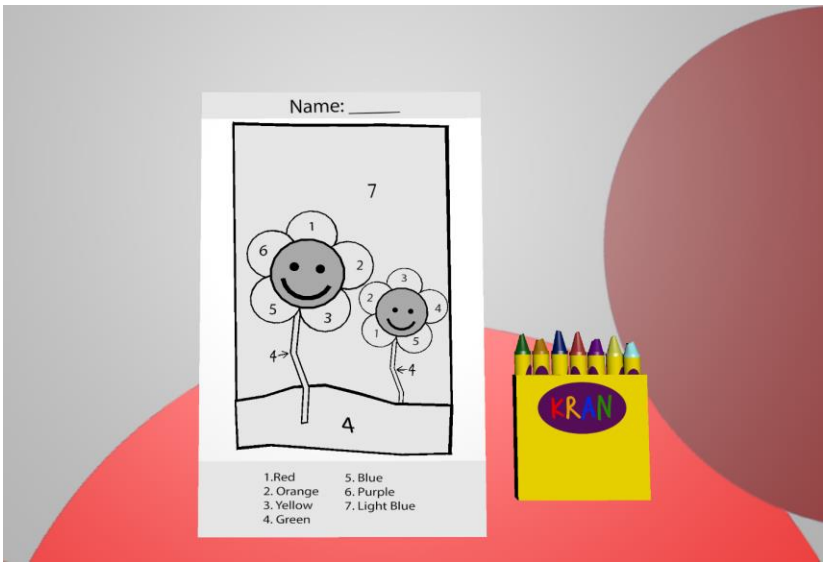


Figure 4.18: Paper and KRANs

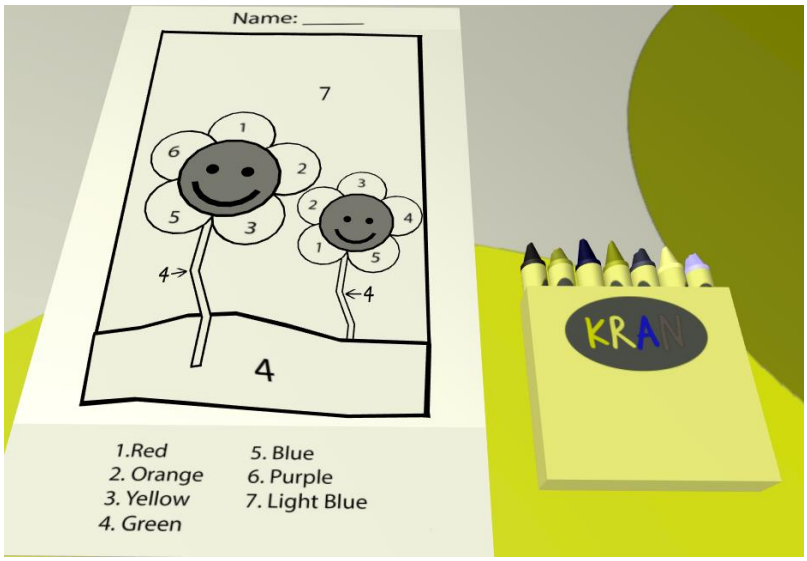


Figure 4.19: Paper and KRANs with Filter

In figure 4.18, it can be seen that the colors of the crayons are not laid out in typical ROYGBIV order, which was done in order to prevent the player from relying on a preconceived layout to figure out which color is which, as well as provide a level of realism to the level. The flower scene requires the player to use every color in the crayon box, maximizing the amount of possible error. There is a slight possibility of the player getting every color correct, but it is a very unlikely case. 1 out of 42 people were able to complete the level with 100% accuracy. Though the level can be completed correctly, our experience goal of empathy is still achievable due to the fact that completing the level correctly is not an easy feat; as well as the fact that seeing the correct colors is not possible.

The layout of the room was designed to mimic a kindergarten classroom and includes objects such as cubbies, backpacks, bookshelves, a colorful rug, books, and a whiteboard. Many of the objects in the scene do not serve any functional purpose in relation to gameplay, but instead are used to populate the scene to add to the believability of the classroom setting. The colors of each of the objects in the scene were selected with purpose, as to not give the player any hints when faced with making color-related decisions. For example, the carpet, seen in figure 4.20, does not have a ROYGBIV color layout, so players cannot reference it when completing the color by numbers

Figure 4.20: Rug

If they do make the assumption that the carpet is in the standard order, they will not complete the level correctly. Additionally, feedback allowed us to see that people were using the poster on the wall to give them a hint as to which table was the correct one. The poster, the “Food Dodecahedron,” which was designed to mimic the food pyramid seen in lots of American classrooms, contains various items, some not food related for humor. Amongst the items on the poster is a glass of red wine. Lots of players deduced that since wine is typically red, the color it appears in colorblind mode must correlate to the color of the correct table. We had been given feedback that it was easy to choose the correct table if they relied on the poster, so we changed the wine from red to green in order to not give any hints to players.

In the last scene of the level, the player must place their completed paper on a board amongst the drawings of other students in order to compare their success in coloring the paper to their fictional peers. If colored very incorrectly it can instantly be seen that the player’s paper is different than the others. All of the other papers are colored correctly so there is no confusion as to which colors the player got wrong. They can then see a screen which tells them exactly how many sections were colored correctly, but the initial shock of seeing their paper amongst the others provides an experience that contributes to the goal of empathy we wished to achieve.

4.5.2. Level 2

The second level is chronologically further along than the first, where the main character, Sam, is about 10 years old. The main goal of the level is to construct a model house using Blockos: a take on the brand Lego. This level was designed in order to further our experience goal by producing another scenario in which someone living with color blindness would be presented with a challenge.

The scene begins with Sam (the camera view) entering the bedroom of a friend, who is playing with the Blockos on an area rug. The friend then beckons Sam to join him and the camera moves toward the rug, then ends at a set position which gives the player a view of the whole workspace: all of the blockos laid out from the middle to the top of the screen, directions

at the bottom left, and an empty space at the bottom right, where the Blockos are to be placed. The Blockos are not laid out in a particular order, and are at slightly different angles to provide a level of realism to the game, as it is likely a child would not take the time to organize and straighten out game pieces before playing with them. They are laid out in such a way that they do not overlap each other and are far apart enough that they player can see each individual piece. Several of the blockos are exactly the same, yet have different colors, as seen in figure 4.21. It is the player's job to guess which of the Blocko pieces are the correct colors corresponding to the directions.

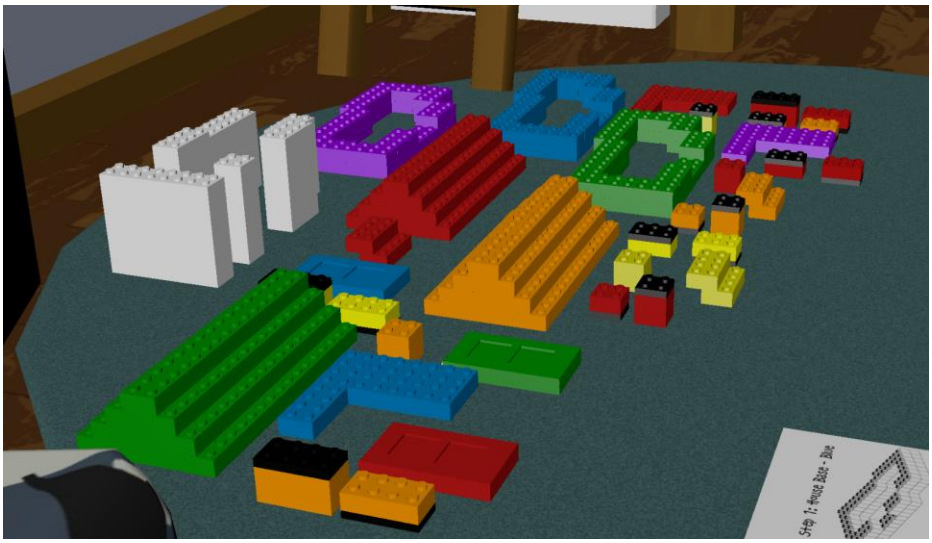


Figure 4.21: Blockos

The directions, which contain step by step instructions regarding the correct order and color of the pieces, may be clicked upon to bring them to the center of the screen, where they can be seen more easily. Each page of the directions contains three sections, which can be seen in figure 4.22, the first being the top of the page which tells the player the step number, piece type, and color.

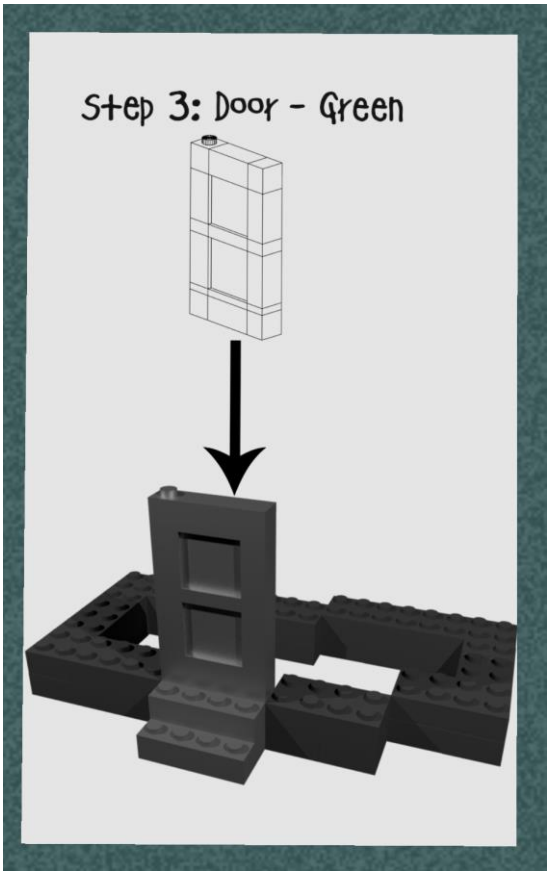


Figure 4.22: Instructions

Below this, is an orthographic view of that piece, taken at a forty-five degree angle. This was done in order to give the player the most surface area of each piece to view, to make it easier for them to distinguish each Blocko shape when selecting a piece off of the rug. This orthographic view was taken as a line render, which serves two functions: the first being that it, as well as the angle, allow for the player to distinguish the pieces more easily. Secondly, a line render has no color to it, so the player cannot compare the color of the piece in the directions to the color they will select from the rug. The last section of the directions is the bottom which shows, in black and white, where the piece is to be placed. The black and white also gives the illusion of a brand which cuts corners and is relatively inexpensive, as a “ripoff” Lego brand would assumedly be. If the player chooses the correct piece and places it in the working area, the current page will slide off of the screen, revealing the next set of directions.

The Blocko house that is to be constructed contains various colored pieces and can be built in ten steps. The first piece to be placed down is the foundation, which is blue. As previously discussed in Section 3.2 the first piece of the puzzle is highlighted for the player to select, move, and place down in the work space. It was for this reason that there is only one blue piece that belongs on the house; the player is explicitly told which piece is blue, and will have that information going forward when placing other pieces. The other step that is a

giveaway to the player is adding the walls, which are white. There is only one set of walls for this reason. The other pieces, however, are in different colors.

Having experienced the first level, we assumed the player will have drawn some parallels in regards to color perception. For example, in the first level, the red table appears yellow in colorblind mode, so player may try to use this information to solve the Blocko puzzle. It was with this in mind, that instead of providing options that were greatly different in colorblind mode, they were very similar in hue. For steps that required the player to choose a red piece, for example, the three options provided for choices are red, yellow, and orange. These were chosen due to the fact that they look very similar in colorblind mode, which is shown in figures 4.23 and 4.24.

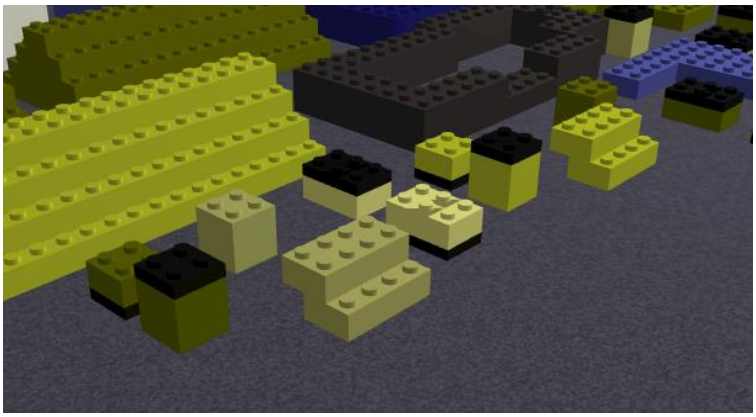


Figure 4.23: Colorblind Filter Blocks

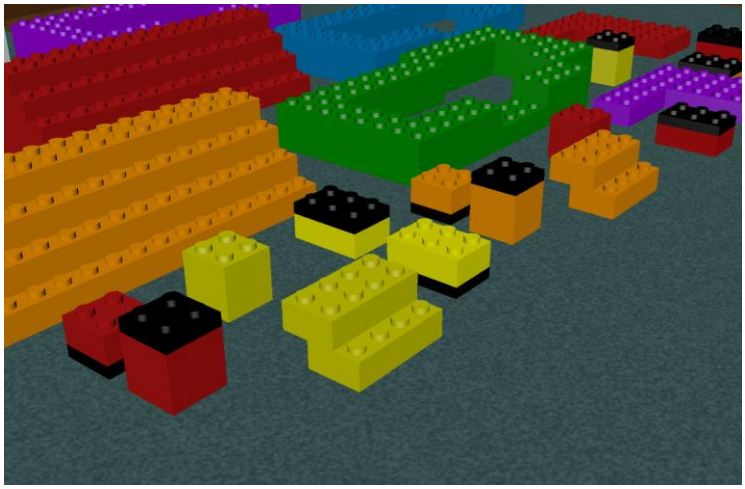


Figure 4.24: Normal Vision Blocks

The level was designed keeping the age group of the main character in mind. The scene is populated with things such as camouflage gaming chairs, a television, superhero poster, toy chest, bunk bed, and desk, all designed to highlight the youth of the character, as the player is not told how old the main character is in the level. The overall layout is shown in figure 4.25, which also shows that nothing in this scene can give the player a hint with the puzzle.

Figure 4.25: Bedroom

Off to the side of the carpet is the Blocko box, which is in grayscale so the player cannot reference it when trying to select the pieces. The rest of the level contains muted colors, such as browns and blacks to not overwhelm the eye.

5. Technical Implementation

This section will describe the technical aspects of the project. Figure 5.1 shows an overview of the implementation of the implementation of Unseeable. Since the game is implemented in the browser, Three.js was used to display the levels. The 3d assets and audio assets were loaded into a Three.js **scene** (defined in Appendix F) and an HTML color filter was layered over it to achieve red-green colorblind vision. HTML event listeners were used to listen to mouse events and react to user interaction. In order to deploy the game on the web, a server was implemented using Node.js and Express.js. The server was also necessary to connect to a MongoDB database, which was used to record game information and survey results from playtesting. Unseeable is hosted on Heroku at unseeable.herokuapp.com.

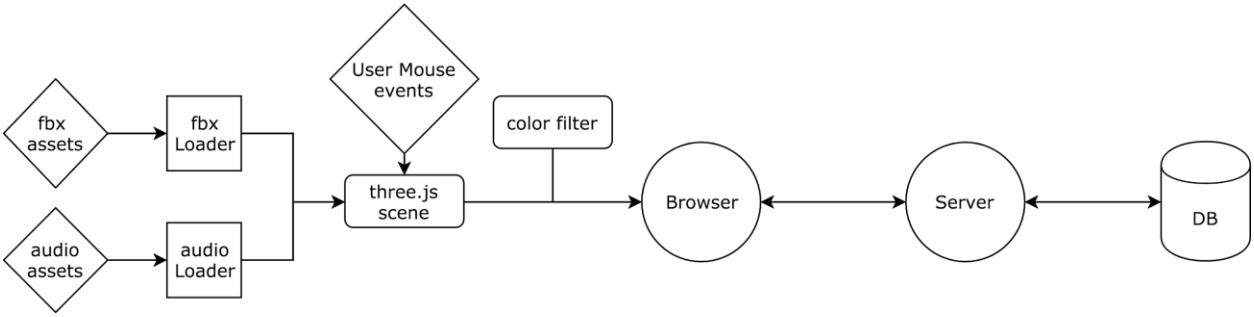


Figure 5.1: Web Application Diagram

5.1. Application Flow

Before discussing the specific tools or implementation techniques used to create the game, an overview of the application's system should be understood. The system consists of two main parts: front-end and back-end. These parts communicate with each other in order to display the game to the user and send feedback to the database for developers to analyze later.

The front-end of an application is the client-side code, responsible for visual elements that can be seen and interacted with by the user. The front-end is where all of our assets are loaded and rendered for the user to interact with. Both the models (**FBX** files) and audio files are loaded into a level through independent loaders. Interaction logic is also defined during load time. All of these assets are integrated into a Three.js **scene**, which, when rendered, will be displayed in the web browser for the user to see. The color filter is applied to the entire HTML page. All of these aspects and processes are contained within the front-end.

The back-end of an application is the server-side code, responsible for site functionality and database communication are the main responsibilities for the server in *Unseeable.* Without the server, the application would not be hosted on the cloud. The back-end of our application defines how the user is able to navigate between different pages in the application. Additionally, the server creates a secure connection to our database, where relevant game and survey data is recorded for later analyses. The server is the line of communication between the database and browser that keep the site functionally stable.

All of these components of our application have particular tools integrated into them. This was done in order to facilitate the implementation process. A description of the tools as well as specific implementation techniques are discussed later in this section.

5.2. Tools

5.2.1. Three.js

This project utilized Three.js, which is a JavaScript library for displaying 3D computer graphics in web browsers. The original project proposed to create the game in Unity, but this idea was discarded because it would detract from the programming aspect of the project. Making the game in Unity would have required much less programming because of how much it is capable of doing. The reason for this is that Unity completely takes care of importing assets as well as rendering graphics on the screen. It also has many built in functions that would have allowed us to implement many of the features that were in our game with minimal to no programming at all. It is because of this that we decided on Three.js because we would be able to use it to implement our own game engine.

Three.js uses WebGL, which is a JavaScript API for rendering interactive 2D and 3D graphics in any browser. It is integrated into all major browsers. Since WebGL allows execution on a GPU, Three.js also runs on one. The source code for Three.js is hosted and available in a public repository on GitHub. The Three.js website includes extensive documentation for most of its classes as well as many examples. We used these both extensively to develop many of the features of the game: such as scenes, lights, cameras, and animation. Three.js has the ability to load in external models, which allowed all models to be exported as FBX files which could be easily loaded in. All that is needed to have Three.js run in a browser is to include the “three.min.js” script in the desired web page. Other scripts that provided extra functionality, such as “FBXLoader.js” had to be loaded in addition to this.

Three.js has performance utilities that can be used to measure how fast it is running. Using these, we determined that it ran fastest on Google Chrome, so we decided that our game would be developed to be played using the Google Chrome browser.

5.2.2. JavaScript

The JavaScript programming language was chosen for this project due to it being the basis for the Three.js API. JavaScript is the primary language used to program functionality in web browsers, and is employed by nearly every site on the Web. Due to JavaScript being widely supported, creating a browser-based game would become a much easier task by basing the entire project around it. In addition, JavaScript comes with built in memory management, making the creation of the game easier due to not needing to make our own memory manager in the game engine.

5.2.3. Node.js

Node.js (or Node) was utilized in order to create the server for *Unseeable*. Node.js is “an application runtime environment that allows you to write server-side applications in JavaScript” (Rachowicz, 2017). What makes Node.js different from other server-side scripting languages was that Node.js is its non-blocking, event-driven I/O (Capan, n.d.; Simoneau, 2010). More specifically, Node.js operates on a single thread. Whenever an event occurs (e.g. an xmlhttprequest), a request is sent to that thread. The request is then sent to the appropriate processes. A callback is set up to return the data output from those aforementioned processes. Since Node.js is non-blocking, the server will not wait for the request to finish before accepting other requests, but will instead take the next request retrieved, send it to the appropriate processes, and set up a callback even before the previous request has been completed. This way the user is not waiting on multiple requests to be completed before being able to perform other actions. Instead, the user will make requests that will be completed independently from each other. Examples of the actions that can be performed by Node.js include: generating dynamic page content, creating files on server, reading files on server, collecting data, adding database files, reading database files, modifying database files, etc. (W3Schools, n.d.). These features make Node.js an extremely efficient environment to implement application servers.

It was determined during the design phase of the project that having a server implemented was necessary for a multitude of reasons. First, the implementation of a server was necessary in order to host the game over the web. All websites require some kind of server side code be implemented in order to be hosted. Additionally, having a server implemented creates a level of security. Unlike client-side code, server-side code can not be seen by the user through the developer console. This feature can be taken advantage of in order to send more confidential data to the server to be uploaded and retrieved later, rather than passing that data around in the client where a user could access it through the developer console.

Lastly, but most importantly, was the database connection that a server provided. As stated before, a server is able to read, write, and modify database files. This feature was very important to the team because the user data was collected had to be stored somewhere to be accessed later. Node.js allowed for a connection to a database. Through this connection, the team was able to send user data to the database to be stored and retrieved later for analysis. Accessing a database via the front-end is possible, but highly discouraged. Passing data that way is extremely unsafe and could give malicious users the opportunity to expose these security flaws and tamper with the data. Having a server not only grants access to a database, but also passes the data into and out of it in a secure fashion.

When considering how the server-side code would be implemented, it became clear to the team that using Node.js was the correct choice. The developers of the team have completed project work utilizing Node.js, so there was a substantial amount of familiarity with the environment. The team also had prior experience with Javascript, which made coding and debugging much easier when server-side issues appeared. Being able to use Javascript for both the front-end and back-end of the application made the code much easier to comprehend due to its consistency, adding another level of familiarity with the environment. If any issues

arose due to unfamiliar aspects of Node.js, the documentation could be read in order to gain a better understanding of the problem. Due to its popularity, the documentation for Node.js is vast. This made it relatively easy to find solutions to problems that the developers encountered. Besides being easy to understand and use, Node.js was actually good for the team's use case. When playing the game, the user should not have to wait on server-side requests to finish in order to proceed with the task they were repeating. Node's non-blocking, event driven I/O removes this latency so that the user can play the game while data is being collected and uploaded to the database in real-time.

5.2.4. Express.js

We used Express.js in order to more efficiently create the architecture for the server-side code. Express.js is defined as "a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile application" (Holowaychuk et al., n.d.). Using Express.js with Node.js allowed for easier management for routing and handling requests in the server.

5.2.5. MongoDB

MongoDB is a document-oriented, NoSQL database program (Medlock, 2017). The documents stored in the database follow binary **JSON**, or **BSON**, format. BSON documents are a "binary-encoded serialization of JSON-like documents" (BSON Specification, n.d.). What makes BSON different from JSON is that BSON also contains extensions that allows for data types that are not part of the JSON specification (e.g. the Date object type). Figure 5.2. is an example of one of the BSON documents in the team's own database.


```
{
   "_id": {
      "$oid": "5ab9685f0e19c80014de46a3"
   },
   "userId": "c9026bb1-05e9-42a3-b8f4-15b898f246c7",
   "answers": [ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3 ],
   "positiveAffect": 30,
   "negativeAffect": 30,
   "__v": 0
}
```


Figure 5.2: PANAS JSON Example

The database is split into different collections in which the documents can be stored. These collections are defined by the Unseeable team's developers and their purpose is to add compartmentalization to the database when necessary. For example, a retail store using a MongoDB database could have a collection containing inventory items and and a collection listing employees. The *Unseeable* project's database was split into several collections for the

following categories: informed consent, first PANAS, IRI, pre-survey, game logs, task logs, second PANAS, and post-survey. Descriptions of these collections are discussed in Section 8.1.

The main feature of MongoDB databases is that they use a non-relational, or NoSQL, data-structure (Xplenty, 2017). This means that the documents stored in a MongoDB database do not have to have a predefined structure, and that the structure of the documents can change without having an impact on the collection or entire data collection (MongoDB, n.d.). This was one of the main reasons our team chose to use MongoDB as our database program; being able to dynamically change data schemas was an important characteristic as the game and data collected rapidly evolved. Members of *Unseeable*'s development team were also very familiar with MongoDB having used it in previous projects. Additionally, Mongo's BSON documents were consistent with the JSON documents that Three.js produced to represent different objects, making it much easier when integrating the two technologies.

The team used mLab as the service to host their MongoDB database. MLab is a cloud database service that hosts MongoDB databases (mLab documentation, n.d.). This service was chosen over others because members of the team have had experience hosting databases for other projects using mLab. Additionally, the service was straightforward, easy to use, and the data was viewable through the mLab website. This ability to view data through mLab's API facilitated the process of testing and debugging by eliminating the need to query the database to perform trivial tasks such as checking if the data was actually persisted to the database.

In order to facilitate the use of MongoDB the team utilized Mongoose, an Object Document Modeling library (Munro, 2017). Mongoose is able to translate MongoDB's documents into objects that can be instantiated within the code. Unlike native MongoDB, Mongoose uses well-defined schemas to model data, which facilitated data validation and made it much easier to read and maintain data due to its consistent structure. Admittedly, using Mongoose in conjunction with MongoDB can come at the cost of performance (Shan, 2015). For the developer's use cases, however, this decrease in performance is negligible as it is only noticeable when there are hundreds of users concurrently accessing the database. This makes the utilization of Mongoose even more beneficial considering the increase in development speed when using Mongoose. The abstraction layer that Mongoose provides removes the overhead of creating a connection to the database, closing it, optimizing it, etc. This makes it much easier for developers to write the code responsible for reading and writing to the database because it is so much more intuitive. The abstraction layer also facilitates making changes to data handling code without halting other processes.

Besides benefits to the development process, Mongoose's data modeling features also made its utilization appealing to the *Unseeable* team. One of these features is the data validation that Mongoose provides. Mongoose's schemas make data validation much simpler to implement than in native MongoDB (MongooseJS Documentation, n.d.). This feature was important to the *Unseeable* developers because many of the documents in the database have fields that require a value in order to be persisted to the database (e.g. a user ID must be present in every document that is persistent). Mongoose also makes it simpler to declare that

fields should be unique within a collection (i.e. there can not be duplicate values in a collection for fields marked as unique). This feature was extremely useful when applied to the user ID field of document because it ensured that the same user could not persist multiple instances of the same type of document to the database. With these benefits, it was clear to the team that utilizing Mongoose was the correct choice during development.

5.2.6. Heroku

Heroku is a cloud platform supporting multiple languages for web application deployment. It features “Git-based, GitHub, and API deployment strategies, a large number of services offered as add-ons, and a full API” (About Heroku, 2011). Acquired by Salesforce in 2013, Heroku is backed by a funding amount of 13 million USD and multiple investors such as Ignition Partners and Redpoint (Crunchbase, n.d.). Heroku’s flexible language support and wide array of pricing options makes it a top choice for multiple developers looking to have their applications hosted on the web.

The *Unseeable* project team decided to utilize Heroku as their hosting platform because its features fit their use cases and needs. Heroku supports Node.js applications, meaning it integrated well with the game since its server-side code was also developed in Node.js. When the team encountered issues with Heroku, it was relatively easy to troubleshoot and find solutions due to the fact that the platform is so widely supported and used. Not only that, but with Heroku’s sandbox plan, there were no expenses involved in deploying the *Unseeable* application to the web. Additionally, the *Unseeable* developers all had prior experience deploying Node.js applications to the web through Heroku, making the process of deploying this application easy due to their familiarity with the platform.

5.3. Colorblind Filter

One of the most important aspects of our game is the Colorblind Filter. This technical feature, written as JavaScript functions that edit the HTML markup of the page, is the part of the game that simulates color blindness on the screen. It does this by adding an SVG element containing a filter within it that is overlaid on the screen. The filter takes the colors that are on the screen and shifts their RGB values so that they are changed to the same hues that a person with red/green color blindness would see. The values for shifting were determined by looking at various programs that had their own filter to simulate color blindness for images and browser tabs. In addition, the colorblind member of our group was able to identify the values that correctly simulated his color blindness. This was possible due to the recently created EnChroma glasses that are able to mostly correct the vision of a colorblind person. By looking at the filtered screen through both the EnChroma glasses and as someone with color blindness, he was able to change the values until the filtered screen looked identical with the glasses on or off.

The reason that this part of the game is so important is that, without it, we would not have been able to easily simulate the experience of being colorblind. If the filter were not implemented, each art asset would have to have been carefully colored so that they did not

contain any hues that a red/green colorblind person could not see. This would have added on a substantial amount of time to the development of art assets. However, thanks to the filter, we were able to design our levels without needing to account for this since the filter handled this aspect of the development process once it was complete.

Figure 5.3: No Color-blind Filter

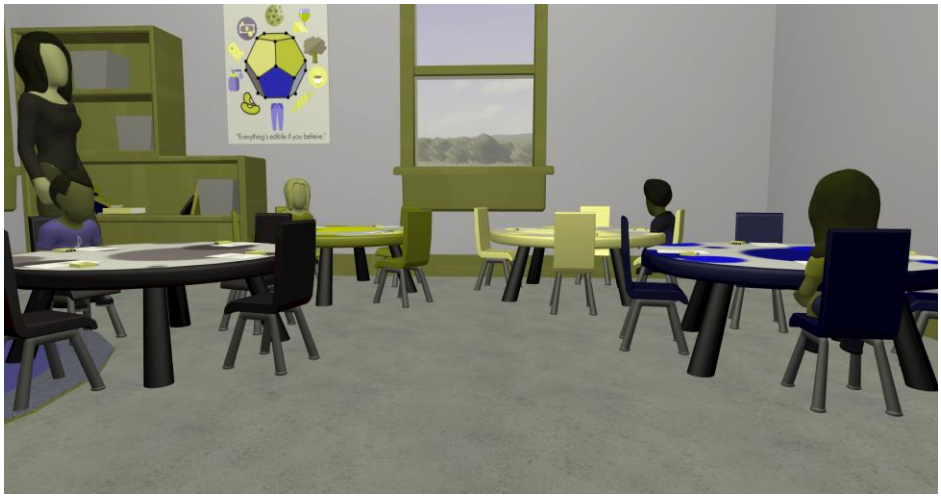


Figure 5.4: Color-blind Filter Applied

5.4. Game Engine Design

Most game engines consist of the main game program, rendering engine, and audio engine in order to run games. Below we will discuss how these were all implemented.

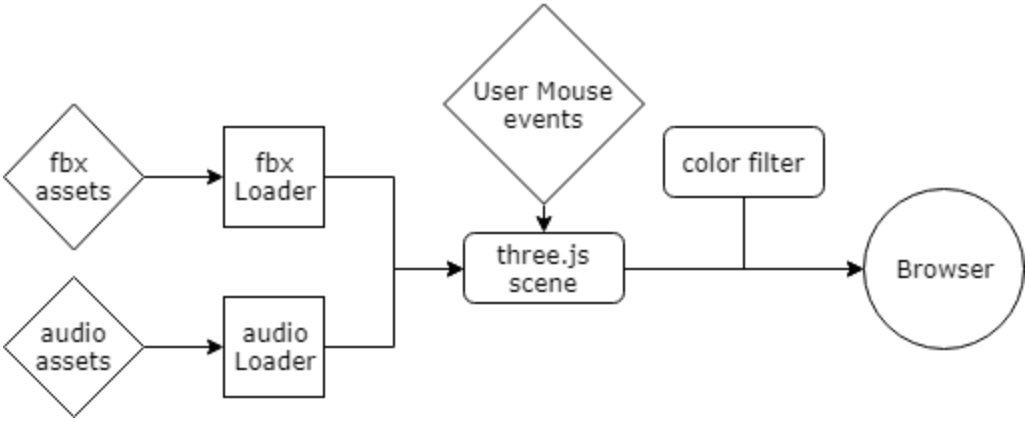


Figure 5.5: Game Engine Design

5.4.1. Main Game Program

The main game program is what controls the game logic. The logic for the game engine can be split into two distinct parts: level loading and event listeners.

The level loading happens soon as the script for the level script is loaded. The first step that happens during this is the setting of all global variables. After this comes the creation of a Three.js scene. Inside of the scene, a camera and lights are placed in their predetermined positions for the level. A renderer is created to render the camera' view of the scene. All external 3D FBX files and audio files are then loaded into the scene. The implementation of loading assets is defined in section 5.2.5. Event listeners are initialized here.

Javascript event listeners were utilized to detect cursor actions. The two event listeners that were used to control interaction were the **mousedown** and **mousemove** events. These used **raycasting** extensively to determine where in 3D space the cursor was located. Raycasting was done by using the cursor position on the screen to cast a ray outward from the camera. This ray is used to detect if the cursor is hovering over 3D object in the scene. If the ray intersects an object, it means that the cursor is hovering over it. The event listeners would then call the appropriate functions.

5.4.2. Rendering Engine

The rendering engine in a game generated and renders the 3D graphics that are seen in the game. Three.js handled all of the rendering for the engine, so it was not necessary to implement a new rendering engine which would have been a tremendous amount of work. Only two components had to be set differently in order to provide the best experience. One of these was to disable shadows, since they drastically reduced performance speed on less powerful computers and did not look believable. The other component that was changed was the use of

anti aliasing, which is a technique in computer graphics to smooth out lines in order to achieve a smoother look.

5.4.3. Audio Engine

The audio engine is responsible for playback of audio and any other audio control within the game. Audio files were imported into the Three.js during level loading. A global **AudioLoader** was initialized that would be responsible for providing the ability to load audio files. Using the audio file name, the AudioLoader would create an **AudioBuffer** that would be passed into an instance of Three.js' **Audio** object. Different parameters of the Audio object (e.g. volume, looping, callback functions, etc.) would be set based on developer input. Once loaded, the sounds could be later played by calling a custom function that would play a sound based on its audio file name.

5.4.4. User Interfaces

These were created by using html and layering it over the play canvas. Three.js does not natively support menus, so we used the html layered over it to control interactions whenever the player was at any of the menus. Using html/css/javascript was relatively painless, where we had the most difficulty having css behave how we wanted.

5.4.5. Asset Pipeline

The asset pipeline is how the 3D assets were imported into the Three.js scene. A file format that could be exported from Maya and imported into Three.js was necessary. Some file formats contain only the 3D geometry and need to load the textures externally. Other file formats need to be static models, while others can include animations.

The format that was decided to use was the FBX (Filmbox) file format. This was chosen because it can include textures, animations, and multiple objects in a single file. The 3D assets in each level were split up into multiple of FBX files: one file containing all of the static objects which were not animated, files for each of the people, and separate files for each of their animations.

Even after deciding on using the FBX file format, there were various issues importing it correctly in the engine. The files had to be exported from Maya with the correct export settings so that the FBX file was created in a way that the Three.js FBX loader would be able to understand. Trying to load assets with the incorrect settings would lead the program crashing or the imported assets missing geometry or textures. An early issue was that if an object had multiple textures, the loader would only apply the first texture, and ignore any others. This would lead to objects having holes where the secondary textures should be. This was temporarily worked around this by splitting objects apart wherever there were secondary textures, leading to a very large increase in the number of object in levels. An update to the FBX loader that happened later during development resolved this issue.

The largest problem that was overcome was importing animations into the Three.js scene. There was a lot of trial and error involved in finding the correct export settings when exporting the animations as FBX files. There were unsuccessful attempts to use different file

formats to export animations in attempts to load them into the scene. Eventually, compatible FBX files were successfully exported by using Mixamo. This became the final solution for getting animations into the Three.js scene.

Later on, models were able to be loaded in T-Pose and their animations were in separate files. This allowed models that each had multiple animations. This greatly decreased workload, because otherwise multiple models with a single animation each would have to be loaded and take a lot of resources, and changing animations would involve swapping out models entirely. The Three.js animation system is very good for handling objects with multiple animations. An example of this is that it allows us to fade between animations, making it look very natural when transitioning from one animation to another.

Besides importing models and animations with FBX files, there were some problems that had to be addressed when importing FBX files. One of these problems was that FBX files that contained lights in them could not be imported. This meant that any lights that were placed in Maya had to be manually placed within the Three.js scene for each level. The splines that the camera moves along did not import as import as splines, even though they were created as splines in Maya. They had to be converted into splines at load time.

5.4.6. Task Implementation

Within the game's levels are multiple tasks and puzzles that the player is required to complete in order to reach the end of a level. The implementation of these aspects of the game are listed and described in this section under their respective names.

5.4.6.1. Table Selection

The first puzzle of the first level was to select the red table out of four possible table options. In developing this puzzle, two important features were implemented: table highlighting and table selection. In order implement these features, the developers took advantage of the Three.js technology stack in order to manipulate the camera and objects themselves.

Table highlighting occured when the user moused over one of the tables in the scene. The purpose of highlighting the table was to inform the user that the table they were mousing over was a selectable option in the puzzle. In initial iterations of this feature, the color of the tabletop was changed to white in order to provide this feedback. In the most recent iteration of this feature a white outline appeared around the table, which allowed the user to continually view the color of the table they were mousing over it as well as know that they can select that table as an answer to the table selection puzzle. An example of a table being highlighted is shown in Figure 5.6.

Figure 5.6: Table highlighting shown with the second table from the left

Creating a white outline around the tabletops was not a simple task. Since all of the objects were in 3D, the outline also had to be implemented using the 3D tools provided by Three.js. These outlines were created while loading in the classroom model into the Three.js scene. In order to create outlines specifically for the tabletops, the corresponding objects were given specific names so that they could be identified when loading in all of the classroom objects. Every object in the classroom has a name field containing a string that can be set before loading it into the game. The object was then passed into the method createOutline, a helper function containing the logic that created the white outline around the tabletops. The createOutline method first creates a **MeshBasicMaterial** instance named outlineMaterial. The color of outlineMaterial was set to white. The "side" field was set to THREE.Backside, which caused only the back face of the object to render. The "transparent" property was set to "true" so when the opacity was changed the outline would actually appear transparent. The opacity was set to 0.9 to look more aesthetically pleasing. Then a new **Mesh** instance was created using the **geometry** of the original object, which was passed in a parameter to the createOutline method, and outlineMaterial. The result is a clone of the original tabletop that appeared white. Since only the back face of tabletop clone rendered, the original table's front face appeared on top the the clone. By scaling the size up slightly and recentering the white table, it appeared as if an a white outline was drawn around the original table when, in reality, the original table was actually just rendered on top of the back face of the white clone. Once the necessary parameters were changed to make the clone appear as if it were a two-dimensional outline, its visibility field was set to "false", making it invisible. This was done because the highlight should only appear when the user moused over a table, meaning that at other other time the highlight should not be present. This clone was then added to the scene as well as to the original table as a reference named highlight.

The highlights were made visible wherever the user moused over a table during the table selection task. Three.js' raycaster class was used in order to determine which table was being moused over. If the object intersected by the ray was a table, which was indicated through the inclusion of the word "table" in an object's name, then the intersected object would be saved as a global reference to be referred to later. The previously created reference to the table's larger white clone would then be used to set that clone's visibility to "true" making it visible in the scene again and the mouse cursor's style would be set to "pointer". If the cursor was not hovering over a table, the cursor would be set to its default look and any highlights that were made visible would have their visibility field set to "false" using the previously defined global reference. The global reference would then be set to false, an indication that there is no table being hovered over by the mouse.

Selecting the table through mouse clicks was another important feature implemented in the game. Whenever the user chose a table, their character walked from their current spot to a spot near the table they chose. In order to implement this walk, ten paths were created in the classroom model for every route the user could take between each table and the initial position at the beginning of the task. These paths were loaded into the engine and converted into splines. To convert the paths into splines, an array of vectors had to be created for each path. Each index in that array represented a point on the spline. That array was then passed as a parameter to the constructor of Three.js' CatmullRomCurve3 class (defined in Appendix F), which creates a smooth three-dimensional spline given a series of points.

To actually move the camera along the path to simulate walking, the spline, the direction the camera should move in along of the spline, and the duration it should take are all passed into a helper function, moveAlongSpline. This function sets the selected spline as active in a separate module responsible for spline movement. When that spline was set to active, a function in the renderer, which only executes code if there was an active spline, became responsible for moving the camera along it. Using Three.js' **Clock** class, a record of how long it had been since the spline was set to active was kept. If the elapsed time was less than the intended duration of spline movement, a new variable was created that was set to the appropriate position on the path. This point was calculated by dividing the elapsed time by the intended duration of spline movement, which equals the percentage of the spline the camera should have moved along. The point was retrieved based on that percentage using the CatmullRomCurve3 class' "getPointAt" function, which retrieves a point on a curve based on a parameter between 0 and 1. The camera's x position was set to this point and this process repeated until the elapsed time equaled the intended duration of spline movement, where the camera would be located at the end of the spline. During this movement the camera's y position was also constantly changing to go up or down based on a calculated small sine wave. This vertical motion made it seem more like the character was walking rather than gliding along the floor.

This walking implementation would only be executed when the player clicked on a table, where the appropriate path was selected based on the point they were currently at and the table they chose. Raycasting was utilized to determine which table was selected. If the objects being

intersected by the ray cast from the mouse were tables, the walking logic would be executed if a click was detected. Otherwise, nothing would happen. The developers added a fourth parameter to the moveAlongSpline function that contained a callback function to be executed after walking along the entire spline. This was used to play the appropriate audio based on the player's table choice and move forward in the level if the aforementioned choice was actually correct.

5.4.6.2. Color by Number

The implementation of the color by number task in level 1 was very involved. There were multiple aspects, while simple to the user, that required ample amounts of work from the developers. These aspects include: the crayon selection logic, the crayon movement logic, the paper highlighting implementation, and the paper coloring implementation.

The first part of the crayon selection implementation was providing feedback to the user about which crayon they were selecting. When the player hovered over a crayon with their mouse, that crayon would move slightly out of the box it was in, shown in Figure 5.7. This feedback would only occur if there was no crayon currently selected by the user (e.g. there was not a crayon in the player's hand). In order to implement this feature, raycasting was used to detect when the mouse hovered over an object. To filter this to specifically crayons, the team's artist gave crayons specific names that would differentiate them from the other objects (e.g. Main_CrayonGreen). When the mouse intersected a crayon, that crayon's x and z position were offset in order to move it slightly out of the box. The mouse cursor was also changed to a "pointer" style from its default look, which is normal feedback for indicating that an item is selectable in a user interface. This cursor style is shown in Figure 5.8. Mousing off a crayon would return it to its original position and return the cursor to the "default" style.

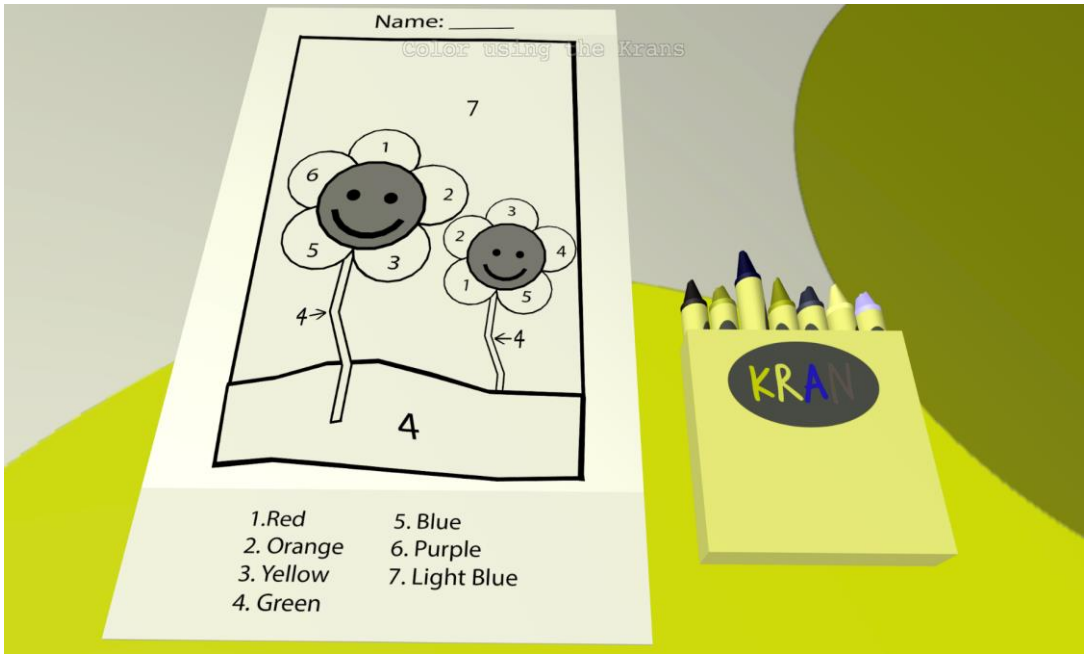


Figure 5.7: Example of a crayon moving slightly out of its box when hovered over

(a) **(b)**

Figure 5.8: Mouse cursors styles default (a) and pointer (b)

Triggering a mouseodown (defined in Appendix F) event while hovered over a crayon made it the currently selected crayon. The crayon object itself was set to a global variable "currentObject" that is used in the paper coloring and highlighting implementation, to be discussed later. The rotation of the crayon was changed in order to orient it as if it was being held in someone's hand. The position of the crayon was also offset in order to align the crayon's tip with the mouse cursor, allowing the user to align the crayon tip with paper section to color it similar to a real-life scenario. Additionally, the mouse cursor's style was set to "none" (i.e. the mouse cursor is not visible) when a crayon was selected. In initial implementations of this feature, the crayon would actually phase through the paper because it was place at the exact point of intersection between the raycaster ray and paper. To remedy this, the crayon was placed 90% along that ray, where it would float slightly above the paper, but appear as it was touching it due to perspective.

The implementation for moving the crayon was similar to selecting it. The only difference was that instead of moving the crayon on a mousedown event, the crayon's position was changed on a mousemove event (defined in Appendix F) as long as there was a crayon currently selected. The position of the mouse was set to 90%, where 100% was the point of intersection and 0% was where the camera was, along the ray intersecting with the scene so that it would not phase through any objects. The offsets to keep the crayon's tip at the point of the cursor were kept the same.

The user was not able to select another crayon if they there was already a crayon selected. To select a new crayon, the user had to place the crayon back in the crayon box first. The user was able to place the crayon back by clicking on the box when they had a crayon selected. In order to better inform the user that clicking on the box would return the crayon to its original location, a ghost of that crayon would appear there when the player hovered over the box. To implement the appearance of these ghost crayons, a clone of each of the crayons were created when loading in the classroom. Any **materials** of the original crayon were also cloned and those clones were set as the materials of the crayon clones. Clones of the material had to

be created because when a material is changed in Three.js, any object with the same material experiences the same change. By cloning the original crayon material and setting it as the clone's material, any change to the clone's material are independent of other objects. The opacity of clones's materials were set to 0.3 and their transparent properties were set to true. This made the them appear transparent. Their positions and rotation were set to that of their corresponding original crayon. The last property that was changed was their visible fields, which were set to false so that they ghosts did not appear while there were already crayons in those same positions. These ghosts were added to the scene and a reference to a crayon's corresponding ghost was added as a field to the original crayon object. When the ray emanating from the mouse intersects with the crayon box and there is a crayon currently selected, the visible property of the ghost crayon will be set to "true" using the reference stored in the currently selected crayon. If the user mouses off the of the crayon box, that property will be set to "false" again. Clicking on the box will also set the visible property of the ghost to "false", return the currently selected crayon to its original position, and set the global variable for the currently selected crayon to "null". An example of this feature is shown in Figure 5.9.

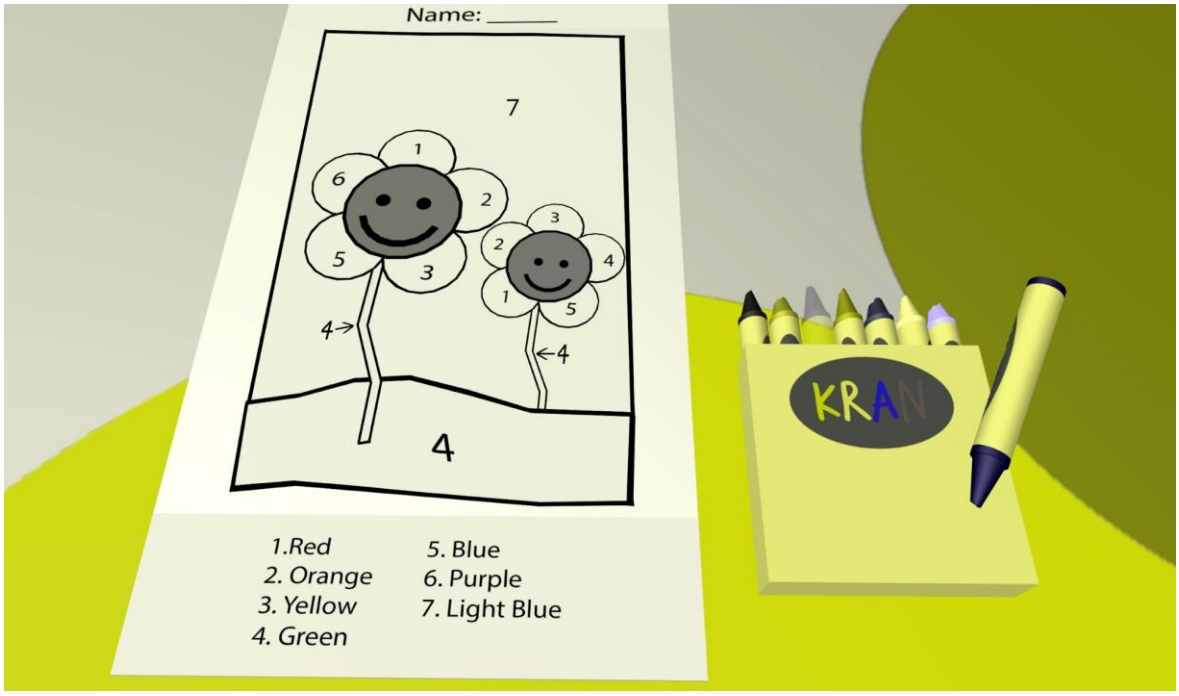
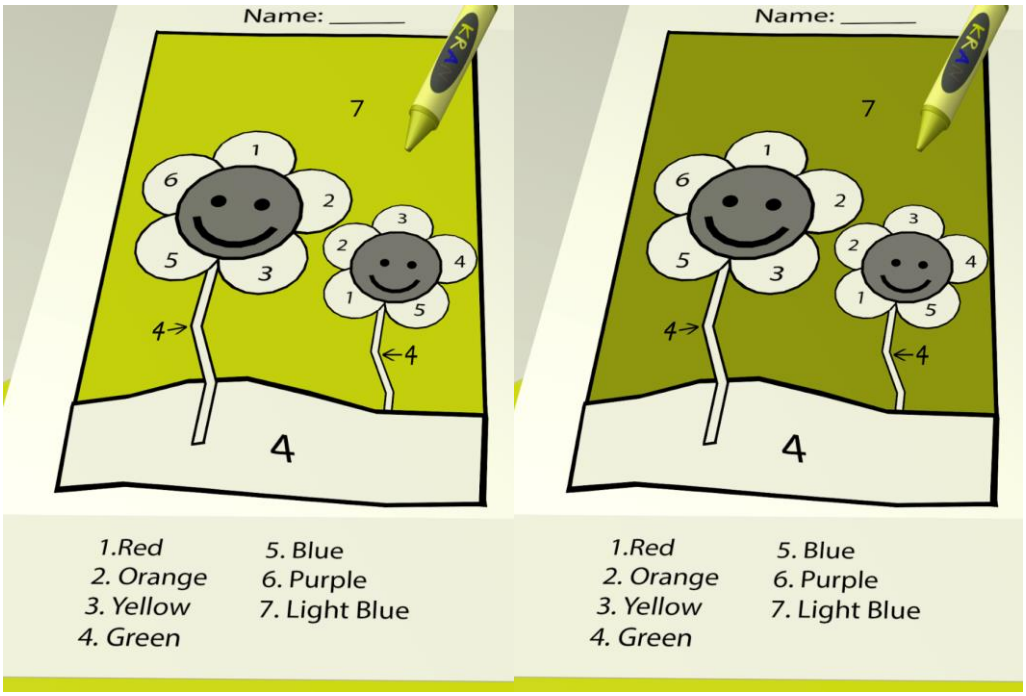


Figure 5.9: Ghost crayon appearing when hovering over box with a crayon selected

In order to provide additional feedback on which section the user was mousing over with the crayon paper highlighting was implemented. Paper highlighting occured when the user moused over a colorable section of paper. That section of paper would change to a lighter version of the color of the currently selected crayon. If no crayon was selected, the paper would not change color. In order to implement this feature, all of the colorable sections of the paper were stored in an array. Whenever the mouse hovers over the paper, the name of the sections of paper being hovered over will be searched for in the array. If that name is found in the array, the paper highlighting code will execute.

To change the color a section of the paper, the material of the paper was cloned first. The color of the currently selected crayon was then set as the new color of the paper's material clone. The rgb values of the color were then increased, which made the color of the material clone lighter than the color of the currently selected crayon. Finally, the material of the paper was set to the newly colored material. The original material of the paper was also saved to a global variable which was used to set the paper section back to its original color when the user moused off of it. This feature can be seen in Figure 5.10.



(a) (b)

Figure 5.10: Paper highlighting (a) vs. paper coloring (b)

If the user clicked on the section of paper, however, it did not change back to its original color when moused off it. Clicking it removed that section of paper from the array of colorable sections,which removed the ability to color or highlight that specific piece again. The user was made aware of this through visual feedback. When the user clicked on a colorable section of paper, the color of that section's material was set to the color of the crayon. Since the highlight color was a lighter version of the actual crayon color, the user was able to view a change between the two as an indicator that they had successfully colored in a section of paper, which is shown in Figure 5.6. After every colorable section of paper was filled in the array became empty, which then triggered the game to start the next cutscene and move on to the paper posting task.

Coloring a section of paper also calls a paper scoring function, which calculated the user's current score and recorded what color they used to color that specific section. The function compared the user's choice to the actual answer, which was stored in a JSON (defined

in Appendix F) file that was converted into an array when loading the level. This array was accessible anywhere within the first level's code. If the user's answer matched the actual answer a global integer, colorPaperScore, was incremented by 1. At the end of the color by number task, the score and recorded answers were packed as a JSON file and sent to the database.

5.4.6.3. Posting the Paper

After finishing the paper coloring activity, the player is asked to place the paper on the whiteboard amongst the other students' papers. Since all the colorable sections of paper were different objects, they had to be grouped in order to be moved together. To do this, the colorable sections of paper were placed in an array when the level was loaded. When the game detected that the player had colored all the sections of the paper, all of the objects in that array were added to an instance of Three.js' **Group** class, making the paper a single object. The paper was rotated to be parallel to the whiteboard in the classroom. This logic executes during a fade out in the cutscene before the user is asked to place the paper on the whiteboard so they do not see the paper rotating.

The camera's position was moved to directly in front of the whiteboard, zoomed in enough to not be able to view the edges but still see some of the other papers. An example of this view can be seen in Figure 3.3. The paper was placed where the player's cursor was before the fade in occured. After the fade in, the player is given the ability to move and place the paper on the whiteboard using the mouse. To move the paper, the point where the cursor intersected the whiteboard was found using raycasting. The paper was moved to this position with some offsets that made it so the paper's center was where the mouse cursor was. When the player clicks on the whiteboard, the functionality to move the paper stops. The camera then zooms out of the view of the whiteboard to view it in its entirety and an audio track plays based on how well the user colored the paper. To implement the zoom out, the camera was moved from its previous position to a position whose z value was slightly farther away from the whiteboard. The transition between the positions was created using Three.js **Tween** class (defined in Appendix F). The conditions that triggered the different audio tracks were if the user colored the entire paper correctly, nine to thirteen sections correctly, or less than nine sections correctly. The audio that played reflected how well the user performed that paper coloring task.

5.2.6.4. Stacking the Blockos

All of the setup happens when the level is loaded. Once the bedroom FBX file is loaded, there is a function that loops through all of the objects in the bedroom, which then processes them based on their names. The names for all of the objects were set by Isaiah when he created them in Maya. Like in the previous level, the lines were converted into splines. The objects that contained 'Blockos' in their name were the blocks that the player would be picking up and interacting with. Their origins are set to the centers of their geometries in order to make their placement and rotation easier. They are assigned a variable which is used to check whether or not it had been placed, so that the player will not be able to pick it up once placed. Each of these objects and their materials are cloned in order to create the ghosts for the blocks to be placed their original positions. The final house that the player has to build is already built

and placed in the level. References to these block are added to their corresponding pieces on the carpet and they are made invisible.

This task is controlled entirely with the mouse, and as a result raycasting is used heavily in this level to determine what the mouse hovers over. If the mouse hovers over an unplaced block, it raises slightly, similar to the crayons in the first level. Clicking on the block picks it up, and the offset of the point of the click and the center of the block is used so that the block is moved around by the point that it was picked up from. With this offset, the block is placed along the ray cast so that the point where the block was picked up from is always under the mouse. The material of the block is copied to the material of the block in the final position so that the color of the ghost matches the color of the picked up block. There is a second ray that is cast from the camera to the center of the picked up block which is used to determine how close it is to one of its ghosts. If close enough to one of the ghosts the, the ghost becomes more opaque and clicking the mouse will attempt to place it down in that position. If attempting to place the block in its final position, first the shape is checked, then its color, and if both of these are fine, then the block is placed smoothly using a tween. If either of those tests fail, then the appropriate voiceline is played for the friend's irritation. If attempting to place the block in its original position, there is no checking and a tween is used to return the block to its original position.

The instructions that are laid out on the carpet have some similar initial setup as the blocks. They also need to their origins set to the centers of their geometries for proper placement and rotation. When they are clicked on, the top page is the only one that is moved. The final page position in front of the camera is calculated by taking the camera's position, applying the camera's quaternion for the rotation, and adding a small offset to distance it from the camera. A tween is used to move the page from its initial position to this newly calculated position. If the instructions are open, any click will start a tween to move them back to their original position. Whenever a block is placed correctly, a tween is started to pull the top instruction page off screen and any future interactions with the instructions will open the new top page.

There are three global variables that keep track of the player's progress through the level. There is one for the building step that they are currently no, which will trigger the good ending once they complete the tenth step. The other two variables keep track of the player's incorrect block and incorrect color attempts. These dictate which audio line to play when the player attempts to place incorrect blocks and will end the level if either of their limits is reached.

5.5. Logger

The Logger was created in to record game metrics in the background of user gameplay. It was implemented as a class that could be instantiated for different levels of the game, containing a variety of methods that either recorded data or persisted it to the database. To create an instance of the Logger, the constructor had to be passed in a player ID and a level ID. The constructor takes these two parameters and stored them in the instance's log attribute, a JSON containing level-related data: the player ID, the level ID, the date the player started the

level, the time the player started, and the total time it took the player to complete the level. An example of the format of this JSON and the definitions of its fields can be seen in Figure 5.11.

{
userId: UUID representing the player,
levelId: the level number (e.g. 1, 2, 3, etc.),
date: the date the player started the level,
startTime: the time the player started the level,
levelDuration: the total time it took to complete the level
}

Figure 5.11: Log JSON with definitions

The player ID field was generated using an **npm package** (defined in Appendix F) that created **UUIDs** (defined in Appendix F). This ID was generated when the user pressed the "Play" button on the title screen and was passed between pages using the **window**'s "**sessionStorage**" variable (defined in Appendix F). The date and time fields were generated using javascript Date class, excluding the need to pass input into the constructor for these items. The levelDuration field was generated when the user completed the level by taking the time (in milliseconds) they finished the level and subtracting that by the time they started the level. To send this log, the endLog function of the Logger class was called at the end of a level, which took the log attribute and sent it to the server through a **POST request** (defined in Appendix F) to the "logger/createLog" address. It was then converted into a Mongoose object and sent to the MongoDB database.

The Logger had an additional attribute, taskStartTime, that was constantly updated to record when a user started a task or puzzle within a level. To record task data, the name of task, the grade the player earned, and any additional information (if any) had to be passed into the logTask function of the Logger class. This function stored relevant task data in a JSON, which is shown with field definitions in Figure 5.12. The userId field was set by accessing the stored value in the window's session storage. The levelId and name fields were manually set by the developer (e.g. they type "1" for the level and "Color by Number" for the task name). The duration was calculated by taking the current time (in milliseconds) and subtracting that by the Logger class' recordStartTime variable, which was updated to reflect the time when the task began. The grade calculations were performed independently from the Logger class and passed in as a parameter to the task logging function. Additional information that the developers wanted to record were passed into the function as an array. This data was sent to the server via a POST request to the "logger/createTask" address. The server then took the JSON data and uploaded it to the MongoDB database.

```
{
    userId: UUID representing the player,
    levelId: the level number (e.g. 1, 2, 3, etc.),
    name: the name of the task being logged,
    duration: the total time it took to complete the level,
    grade: the score the user earned for the task as a fraction (e.g. 0.44),
    additional: any additional information to be recorded, stored in an array
}
```

Figure 5.12: Task JSON with definitions

6. Art Production Pipeline

All of the art assets were created with many constraints and factors influencing the design, including the fact that the game was created in a custom engine, runs in a browser, and must fit aesthetically with the chosen art style. The objects and characters followed different production pipelines, and included various softwares in order to be completed.

6.1. Level Objects

The two levels of Unseeable were created in Maya, and initially only contained a few placeholder objects. The scenes, one of which can be seen in figure 6.1, were then exported in the .fbx file format.

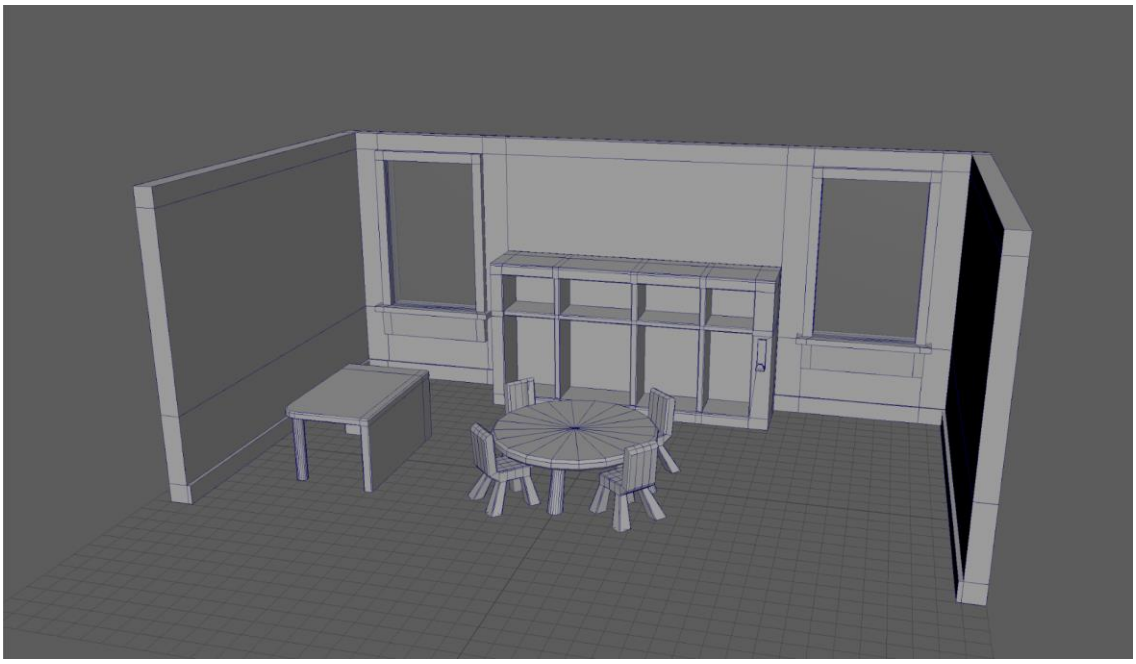


Figure 6.1: Initial Object Creation

The .fbx format was chosen due to the fact that it allows objects to retain the information associated with them in their initial 3D package; which was, in this case, Maya. This means each object will retain its scaling information, smoothing information, textures, etc. once exported. The simple scenes were then sent to the technical team which allowed them to test the .fbx loader and make sure all of the objects loaded into the scene in the correct places. Once confirmation was received that the objects could load correctly, the scenes were then populated with more objects.

All of the level assets, such as tables and shelves, were created using as little extra geometry as possible while still allowing them to fit into the cartoon-like theme discussed in

Section 4.4. Having less geometry per-object allows for faster rendering during runtime, which is a concern when it comes to a game running on a web browser on various computers with different processing capabilities. Figure 6.2 shows the polygons making up each of the objects in the first level, and it can be seen that each object does not have much density.

Figure 6.2: Level 1 Wireframe

In addition to simplistic geometry were simplistic materials, which were created in Photoshop. Many textures were hand painted, such as the table tops and the rug texture; the others are photo textures. Photo textures started with photorealistic detail, then had various filters applied in Photoshop to achieve the desired cartoon look, as seen in figure 6.3.

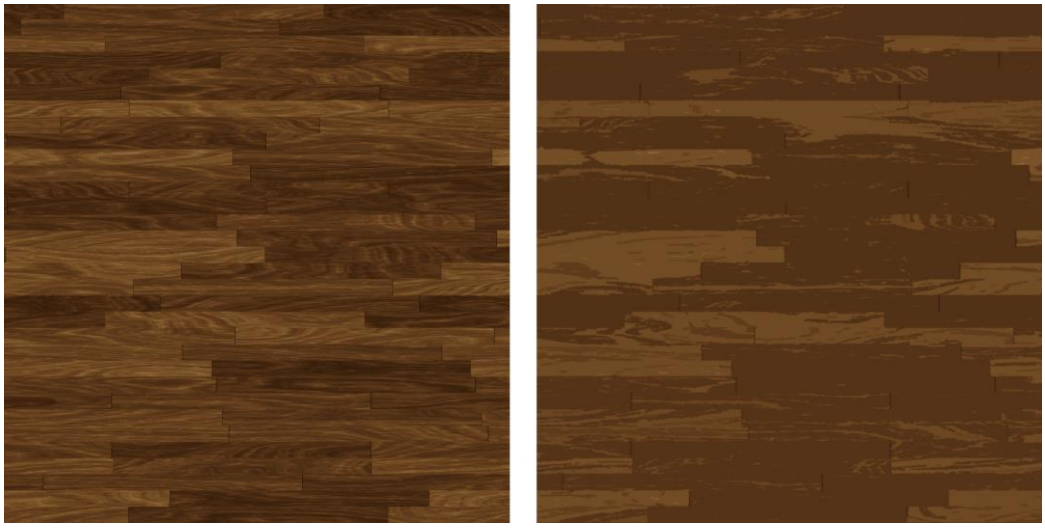


Figure 6.3: Original photo texture (left) and texture with filters applied (right)

Most of the materials in *Unseeable* only contain diffuse textures, making them appear flat. This was due to the fact that any materials with specular parameters appeared *very* shiny within our game engine. Any specular parameters that were changed within a material's settings in Maya did not translate to 3.js, and appeared distractingly shiny in-game. It was for this reason that very few materials have specularity values. Plastic objects, such as the chairs and the sides of the tables in the first level, and the blockos in the second level, all have specular values because the items they represent are normally shiny.

6.2. Characters

The characters, seen above in section 4.4.1 were created in ZBrush using various sphere chains then sculpted musculature. The clothing and hairstyles were also created in ZBrush. The hair and clothing were created using a modular approach, meaning different hairstyles could be paired up with different clothing, effectively creating entirely new characters. The characters below in figure 6.3 are wearing the same outfit, which are just textured differently.

Figure 6.4: Character Clothing

After all character assets, i.e. hair, clothes, and body, were modeled, they were retopologized in order to minimize the amount of geometry used by each. Reducing the amount of polygons in each character was very important, as they were much denser than the other objects in each scene. Leaving the characters at their original polygon count would make the game run very slowly, especially when each character is animated. Once the characters were

retopologized, they were individually poly-painted in ZBrush, which creates the diffuse textures for each.

The characters were then individually exported to 3ds max, where the character assets were condensed into one object, and scaled to the correct dimensions in order to accurately fit into the level. They were then uploaded to Mixamo, which is a service that auto-rigs biped characters and provides various animations which can be applied to them. Due to the fact that many characters had to be created, each with different animations, Mixamo was utilized. From Mixamo, each character was downloaded in a T-pose and sent to the technical team who applied the downloaded animations to each T-posed character.

7. Audio

The audio elements of the game were created through a combination of free, online sources and through recording the voices of various people for the vocal audio in the game. All of the sounds that were used were edited in the free audio editor software Audacity. It is in this program that the quality of the audio recorded and found was able to be improved by removing noise and by changing various aspects for different parts of it to suite the needs of the game.

7.1. Vocals

The choice to have characters speak within the game was done in order for the player to receive instructions more naturally, rather than simply displaying the instructions on the screen. This helped to add more realism to the game so that the player could further immerse themself within the game. In addition, this decision was made in order to deliver larger emotional feedback to the player, something that would be much harder to do by just displaying text within the game.

7.1.1. Level 1 Vocals

For the first level of the game it was decided that the teacher would be, other than a couple student voice lines, the only character that would speak in the level. This was done as a teacher would, in the situation of a classroom, be the most appropriate person to be giving instructions to the player's character, a student. A script was written for the teacher that would allow the teacher to give the player instructions while still sounding natural in order to keep the feeling of realism within the game. When the script was complete, a person was found who could voice the teacher. The lines were recorded and edited to remove any background noise before placing them in the game. The voice was recorded to make the teacher sound kind and caring, but also to imply that she did not understand that the player's character has colorblindness.

In addition to the teacher giving instructions, there are three different voice lines that can play at the end of the level. These were played at the end of the level to give the player feedback about how they did on the coloring task, as well as to deliver a larger emotional impact on the player. The first voiceline that can occur is the most common as it plays when the player gets between one and nine colors correct. In this case, a child in the class, who is offscreen, will declare that the player does not know how to color and the entire class will laugh at the player. This was the main voiceline that was created to evoke more emotion out of the player by making them feel embarrassed and humiliated, in order to have them understand that people can be cruel to those who are colorblind. The second voiceline that can play occurs when the player gets between 10 and 13 colors correct. In this case, a child in the class will point out that the player's coloring sheet seemed a bit off which was done to tell the player that, despite being close to correct, it was still noticeable that they did not get all the colors correct during the task. The final voiceline that can play occurs when the player gets all the colors correct. In this case, the teacher states that the kids did a good job and tells them it's time for a story as the level ends. This was done to show the player that, even though they struggled greatly to complete the

task, it was something that all other students were able to easily do and they would get no special reward or extra praise for completing the task.

7.1.2. Level 2 Vocals

For the second level, it was decided that the player character's friend would be the most appropriate character to be voiced in this section of the game. This was done to, as with the first level, provide instructions to the player while keeping a sense of realism within the game. However, unlike the first level the vocals also served to provide the player with immediate feedback on their choices as they were happening as well as gradually evoking more emotion out of the player through the feedback. This was done by writing a script in which, if the player tried to place the wrong colored pieces when building with the blockos, the player character's friend would respond differently depending on the number of times the player had tried to place an incorrect piece. At the first, the friend will gently say that the player is wrong and points out which block should be used, serving as the only hint the player will get in the level. After the player makes another mistake, the friend will start to become noticeably annoyed with the player. The annoyance will begin turning to frustration and anger at the player until enough mistakes are made where the friend yells at the player and says they no longer want to play with them. This was done to make the player feel bad for getting the pieces wrong. In addition, it serves to show the player that those who do not understand colorblindness can become easily annoyed and angry at those who are colorblind despite the fact that the colorblind person cannot do anything about it.

When the player continually gets pieces correct, the friend will respond with some encouraging voice lines to indicate that the player is doing well. If the player is able to complete the task without angering the friend to the point where he doesn't want to play anymore, the player will be able to hear one of two different voice lines. The first occurs if the player got between zero and two errors when building. In this voiceline, the friend will say that the player did well and the game will come to an end. This was done in contrast to Level 1, reward the player with a small amount of praise due to the increased difficulty of the task. The second voice line occurs when the player gets three or four mistakes before completing the task. In this case, the friend will be happy that the player finally finished but will remark that they didn't think the task should've taken so long as it wasn't that difficult. This was done to show that, even if a colorblind person is able to complete a task that is difficult for them and not others, they will still be judged based on the standards of those who are not colorblind. The task also shows that a colorblind person can also be expected to complete the task in the same time it takes a person with normal color vision to complete it.

As with the first level, a suitable person was found to voice the friend. The person was chosen due to their ability to weave emotions into their voice, allowing for the lines to more easily evoke emotion within the player. Once the lines were recorded, they were then edited in Audacity to remove any background noise as well as to change the pitch and speed of the lines to make the voice sound like that of a child's as well as to make the pacing more natural.

7.1.3. Vocals Subtitles

With the introduction of vocals to the game, it was decided that it would be best to add subtitles to go with the voice lines within the game. The reason for this was to make sure that players would not mishear the voice lines and so that those who cannot hear the voices for any reason can still get instructions as well as feedback on their progress in the game. While the subtitles do not deliver the same emotional impact as the voice lines themselves, they help to ensure that the game is accessible to those who are unable to hear the audio within the game.

7.2. Additional Audio

While the majority of the audio in the game is comprised of the voice lines delivered to the player, there are some audio pieces that were inserted into the game to make it feel more realistic and to fill in the void where no voice lines are taking place. This was done by first adding ambient noise to each level, with the first level having the sound of kids talking and playing in the background. This seemed to be the most logical choice as the level takes place in a kindergarten classroom: a place that would be filled with the sounds of small children talking and playing. The second level had the ambient sounds of a forest outside as well as the low hum of a computer. This was done because the level takes place within a child's room, with the outdoor sounds being the most logical choice as well as the hum of the child's computer being something that is present in many rooms in current times.

Other than the ambient sounds, various sounds to indicate that actions have occurred were placed in the game to add a small sense of realism to it. This includes the sounds of picking up a crayon, placing the crayon back in the box, coloring with the crayon, placing the coloring sheet on the board, picking up a blocko, placing a blocko down, and the sound of paper moving when selecting the instruction sheets in the second level. These simple sounds were included to not only add to the realism, but to provide audio feedback to the player that an action had been performed in order to reinforce that something had occurred.

8. Research Methodology

As stated previously, the main goal of the game was to allow non-colorblind individuals to see from the perspective of a colorblind person and understand some of the struggles they endure in their everyday lives. This means that the participant should undergo some amount of stress while playing the game, indicating that they themselves are experiencing struggle while experiencing vision impairment. In order to determine whether or not the game elicited any stress from the user, subjective surveys were given to the user to complete. These surveys indicated their disposition towards empathic concern and changes in mood.

8.1. Data Collection Methods

Several different methods were used order to collect use data. The type of methods consisted of mostly surveys as well as a Logger that collected game data as the user played through different levels.

8.1.1. Informed Consent

The informed consent document is required by the Institutional Review Board (IRB) in order to collect data from users. The document must be sent to and approved by the IRB before collecting any data using it. The document describes general information, procedures, risks, benefits, compensation, and contact information related to the study. Additionally, the informed consent explicitly states that any and all of the participant's information will be kept confidential and that no publicly identifiable information will be disclosed in the study. The participant is asked to sign the consent form indicating that they have read and understand the information provided in the document and that they are ready to participate in the experiment (i.e. take further surveys and play the game). For ease of use, the informed consent was digitally adapted so that the user could provide an electronic signature and continue the study through the same browser window. The full informed consent document can be located in Appendix A.

8.1.2. Pre-Survey

The pre-survey was meant to gain some general information from the user about factors that could influence their perception of the game (e.g. if they are already colorblind or have close ones that are colorblind). By collecting this information it is possible to separate, or at least identify, outliers that may exist within our data. The full-pre-survey in its digital form is located in Appendix B.

8.1.3. Post-Survey

The post-survey was meant to collect feedback after playing the game. More specifically, the post-survey asks questions pertaining to the general difficulty of the puzzles in the game, whether the user feels as if they have learned more about the colorblind experience, and if they have any feedback related to the game or experiment they would like to share. This information was used to see whether or not the perceived difficulty of the game had any effect on the

participant's mood, whether or not the participant felt they learned about colorblindness, and provide a method to receive direct feedback from the participant. The full post-survey in its digital form can be located in Appendix C.

8.1.4. PANAS

The Positive Affect and Negative Affect Scale (PANAS) was meant to measure the participant's current positive mood and negative mood. The scale was given before and after the user played through the first level. The scores of each instance of the scale were compared in order to see how the participant's mood changed based on this activity. The change in positive mood is quantified by calculating the difference between the positive affect of the two scale instances. The same process is used to quantify the change in mood using negative affect. This change in positive and negative mood was how the team was able to examine whether or not users felt stressed out about the game at all without having to take direct physiological measurements. The full PANAS is located under Appendix D.

8.1.5. IRI

The Interpersonal Reactivity Index (IRI) is meant to measure empathy as a trait of an individual based on answers to a subjective survey. Beyond that, the IRI is also capable of measuring a subject's disposition to perspective-taking, placing themselves in fictitious scenarios (i.e. fantasy), and personal distress. Collecting this information allowed the team to see if people who were generally more disposed to empathic concern had a more pronounced difference in mood after playing the first level based on the PANAS. The full IRI is located under Appendix E.

8.1.6. Game metrics

The description of the game statistics and how they were recorded is located in Section 6.3. Dates and times that the user played the game were kept in order to more confidently match logs with user IDs. The time spent playing levels and completing puzzles were recorded in order to examine any patterns in the amount of time a participant took and whether time affected their change and mood significantly. The user's score for each task was also recorded in order to examine whether it had an effect on the participants mood. Additional data was recorded only if the team thought if there were other metrics that could derive interesting patterns (e.g. seeing if there was a pattern in colors used during the 'Color by Number' activity).

8.2. Method Sequence

Figure 8.1: Data collection method sequence

Due to the nature of some of the forms and surveys there was a specific sequence in which users had to view them. The informed consent was the first form in this sequence. As required by the IRB, the user had to sign the informed consent before any data was collected

from the user. After signing the document the user was then asked to complete the first PANAS. The PANAS had to be taken before any other survey or playing the game so that it reflected the user's state of mind before any part of the study could affect their mood. The IRI immediately followed the completion of the PANAS, which was then followed by the pre-survey. The order of the IRI and pre-survey had no real significance besides having to be completed before the user played the game. After the pre-survey, the user entered actual gameplay. The gameplay for different levels is described in Section 3. Game metrics were recorded in the background as the user played through the levels. What metrics were being recorded and how this recording was implemented is discussed in Section 5.3. After completing the game, the user was asked to fill out another PANAS in order to measure their mood immediately after experiencing the colorblind perspective. Finally, the user was asked to complete the post-survey. The post-survey was introduced last so that the user did not have time to think or relax after a potential stressor was introduced (i.e. the game) before measuring their mood with a second PANAS. Additionally, the post-survey was the section of the study in which users could express any feedback they had, making it the most sensible to introduce the survey after the user completed all the other surveys and played the game.

8.3. Data Analysis

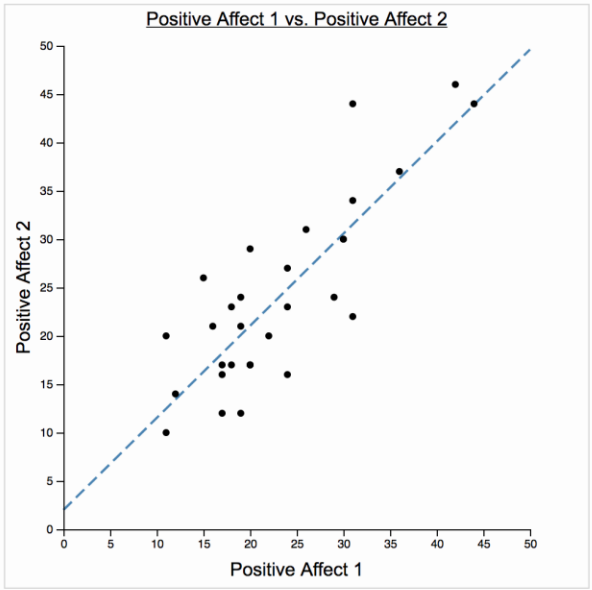
There was some necessary filtering that had to be done with the data before it could be used to create any visualizations. Depending on the information being plotted, data points had to be removed from the dataset if there was not a consistent user ID present among the necessary sections. A user ID not being present indicates that the user began the study, but did not complete it within a single session, where a session is defined as the period of time that the window the game was being played on was open. For example, some PANAS data could not be used because because users would complete the first PANAS, play the game, but then close the window before continuing on to the second PANAS. In graphs where is was necessary to plot the change in positive or negative affect using PANAS scores as a measure, having the user fill out both instances of the scale was absolutely necessary. Any datasets where a similar issue occured was filtered in a similar fashion.

Once the data was filtered, a series of visualizations were created in order to more easily understand the the results of the study. Scatter plots were the only type of visualization used to represent the data. The choice was deemed the most appropriate considering that the results of data relied on viewing the correlation between two variables (e.g. change positive affect, change in negative affect, empathy score vs. change in positive or negative affect). Using a scatter plot allowed for all the data points to be viewed to discern if there was a specific pattern that was being followed based on the shape the graph. A linear regression line was generated for each scatter plot in order to discover the overall trend of the datasets being examined, if any.

9. Results and Analysis

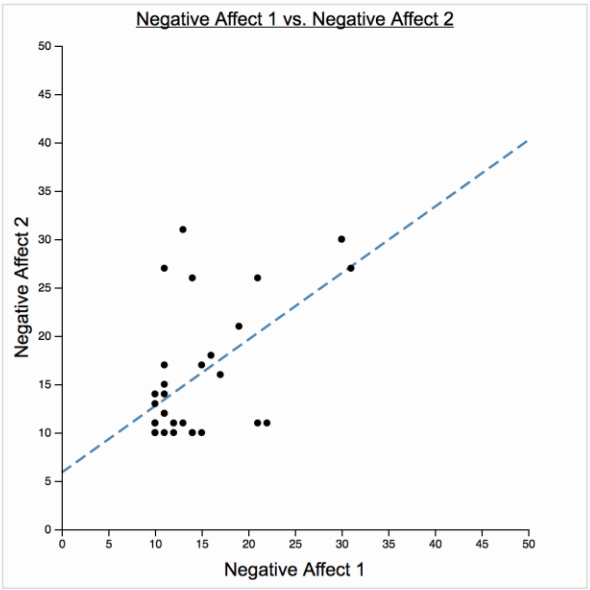
The goal of the game was to have non-colorblind individuals empathize with the daily struggles of the colorblind population and the stress they endure. In order to determine if our game accomplished this, the stress of the players had to be measured in some fashion. The PANAS was utilized in order to discern if the player experienced stress. By having them take the PANAS before and after playing the game, the team was able to analyze the data and determine if the game caused stress, which would have been indicative through a decrease in positive affect and an increase in negative affect.

This section not only discusses the changes in player mood after playing the game, but also reveals correlations between those changes in mood and other measured values. All of the visualizations in this section are scatter plots with linear regression lines. The scatter plots allowed us to examine individual data points to look for patterns or outliers. The linear regression line on each graph informs us of the overall trend of the data without taking away from the examinable pattern formed by the distinct scatter plot points. The slope-intercept equation of each line can be seen below each graph for more detailed analyses.



y = 0.9522661523625844x + 2.0011043021121724

(a)



y = 0.6872946640987878x + 5.861957629998868

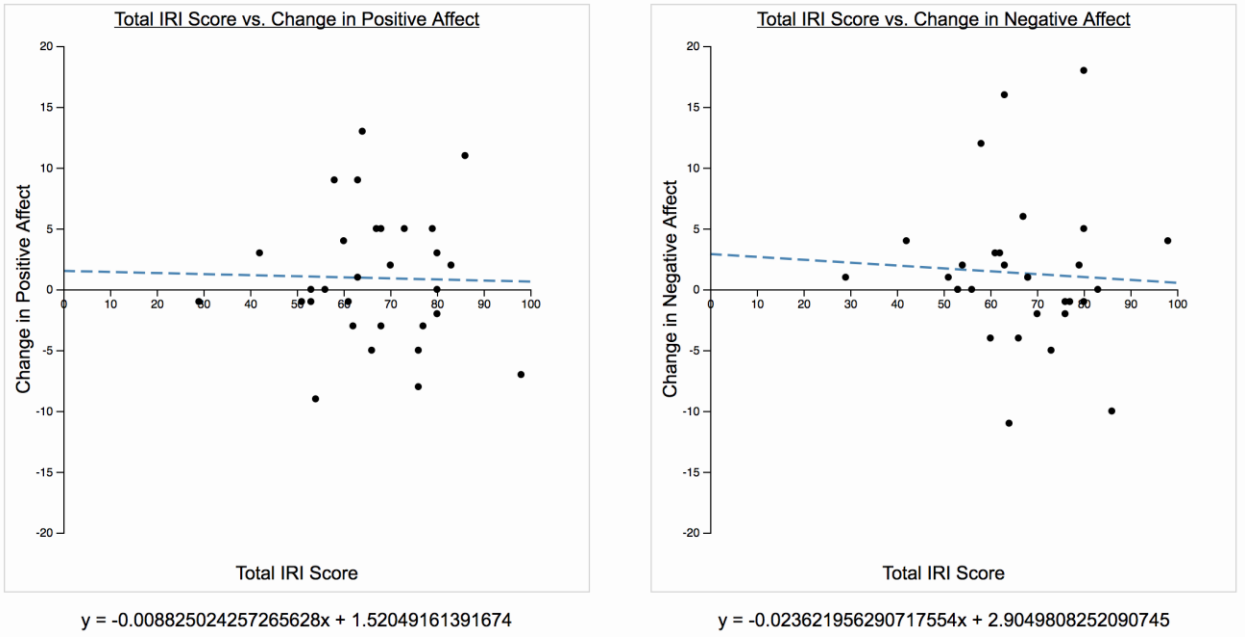
(b)

Figure 9.1: Positive affect before playing game vs. after playing game (a) and negative affect before playing game vs. after playing game (b)

To determine whether or not the game caused stress, the PANAS scores before and after the game were compared. More specifically, the change in positive affect and change in negative affect were examined. This data can be seen through the scatterplots in Figure 9.1. The scores from the first PANAS are plotted on the x-axis, while the scores from the second PANAS are plotted on the y-axis.

Figure 9.1.a displays the comparison of positive affect scores. The slope of the trend line is approximately 0.952. This indicates that on average, there was a slight decrease in positive affect after playing the game. The distinct points on the scatter plot closely follow the shape of the trend line, confirming that this correlation is reliable.

The negative affect of users also decreased. Figure 9.1.b displays the comparison between negative affect scores before and after playing the game, the slope of the trend line is approximately 0.697. This indicates that there was in even lower decrease in negative affect than there was for positive affect. The data, however, is very scattered and does not show a particular pattern, making the trend of the linear regression line less accurate.



(a) (b)

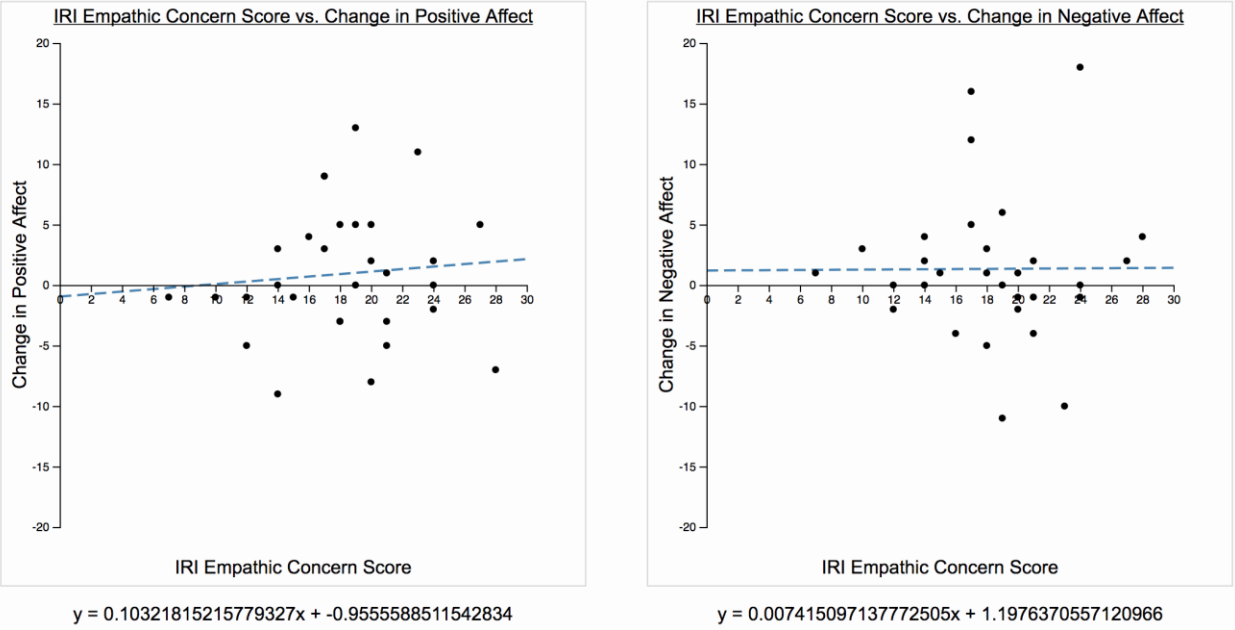
Figure 9.2: Total IRI score vs. change in positive affect (a) and total IRI score vs. change in negative affect (b)

The players' dispositional empathy was measured via the IRI. This data was recorded in order to compare it to changes in mood. The purpose of this comparison was to examine if those more disposed to empathy or its sub-categories would be more likely to experience a greater change in mood after playing the game (e.g. a greater increase in negative affect). Comparisons between the total IRI score and change and mood can be seen in figure 9.2. When viewing the graphs in figure 9.2 and any similar visualizations involving changes in PANAS scores, it is important to note that a positive value indicates an increase in affect while a negative number indicates a decrease in affect.

The relationship between the players' total IRI score and change in positive affect can be seen in Figure 9.1.a. The slope of the trend line is approximately -0.009, indicating that users with higher IRI score showed a miniscule decrease in positive affect. The slope is so minute,

however, that any correlations that derive from is are negligible. The data itself is also very scattered, which reduces the accuracy of the trend line and any derived from it.

The trend line in Figure 9.2.b suggests that players with higher IRI scores saw a decrease in negative affect with an approximate slope of -0.024. Similar to the Figure 9.1.a, however, the data set is very scattered, which reduces any generalizations made by the linear regression.



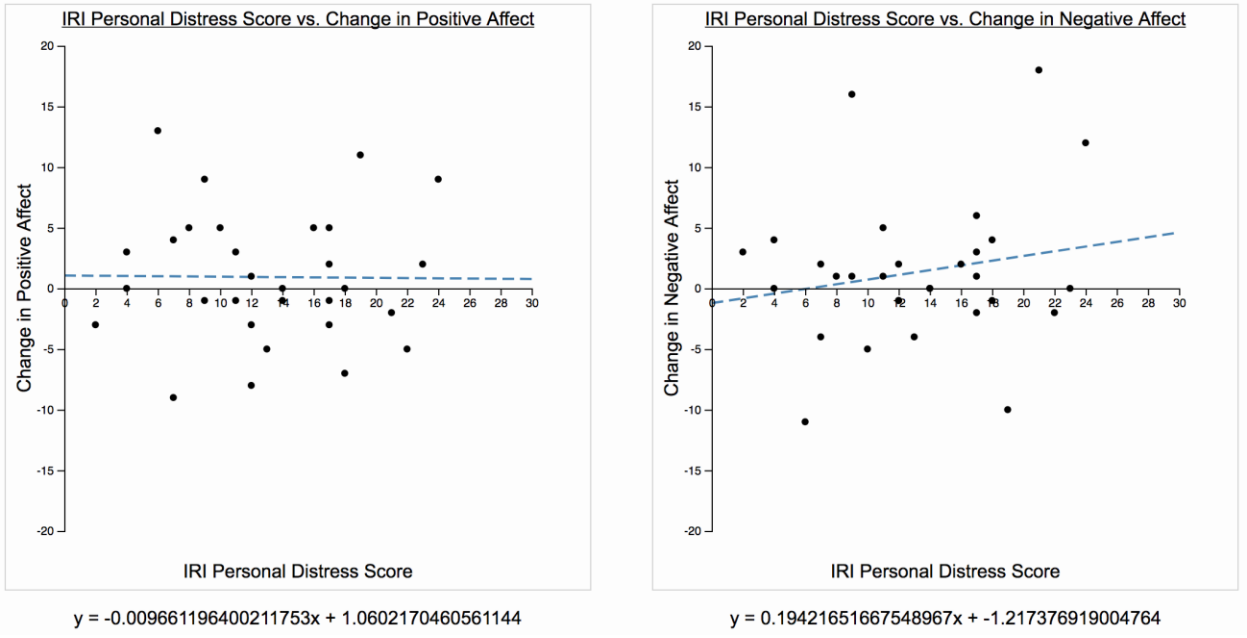
(a) **(b)**

Figure 9.3: IRI empathic concern score vs. change in positive affect (a) and IRI empathic concern score vs. change in negative affect (b)

After examining the overall IRI data, the changes in mood were compared to IRI scores in its different categories: empathic concern, personal distress, perspective-taking, and fantasy. This data was compared in order to discern whether or not more specific types of empathy affected the stress induced by the game.

Figure 9.3.a displays the relationship between player IRI empathic concern scores and changes in positive affect. The positive slope of the trend line (m = 0.103) indicates that users who scored higher in this subcategory of the IRI were more likely to experience an increase in positive mood after playing the game.

The data visualized in Figure 9.3.b indicates that after playing the game users experience a slight increase in negative affect if their empathic concern score is higher. The slope of the linear regression line is so miniscule, however, that this correlation is negligible. Additionally, the data's scattered nature makes any deductions from it less valid due to its inaccuracy.

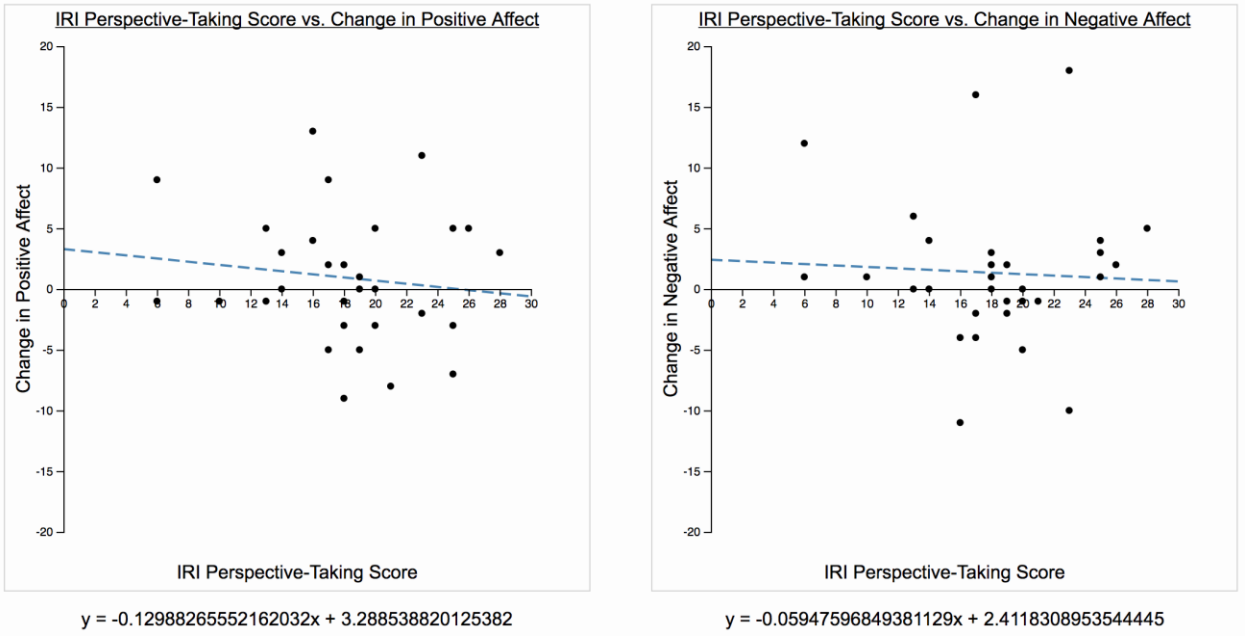


(a) **(b)**

Figure 9.4: IRI personal distress score vs. change in positive affect (a) and IRI personal distress score vs. change in negative affect (b)

The relationship between users' IRI personal distress scores and their change in positive affect is visualized in Figure 9.4.a. The trend line (m=-0.010) indicates that there is a small decrease in positive affect, but the data set is too scatter to make this derivation reliable.

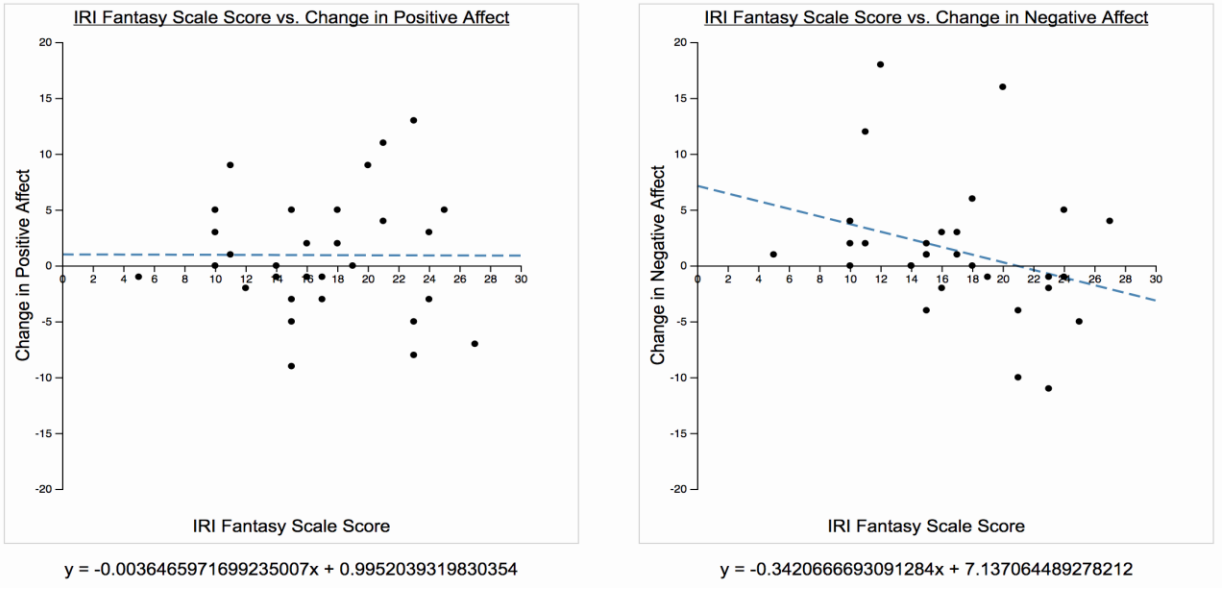
Figure 9.4.b displays how users' personal distress scores affect their change in positive mood. The positive slope (m=0.194) of the linear regression line implies that users with higher personal distress scores are more likely to experience an increase in negative affect. While the data seems to contain multiple outliers, the pattern of the data points somewhat follow this trend.



(a) **(b)**

Figure 9.5: IRI perspective-taking score vs. change in positive affect (a) and IRI perspective-taking score vs. change in negative affect (b)

Figure 9.5.a displays the relationship between users' IRI perspective-taking score and their change in positive affect after playing the game. Figure 9.5.b compares the perspective-taking score against their change in negative affect. The linear regression lines of both graphs suggest that users who have a higher perspective taking score are more likely to see a decrease in both negative affect and positive affect. The data in both graphs, however, is neither accurate nor precise, making any assumptions that derive from it unreliable.

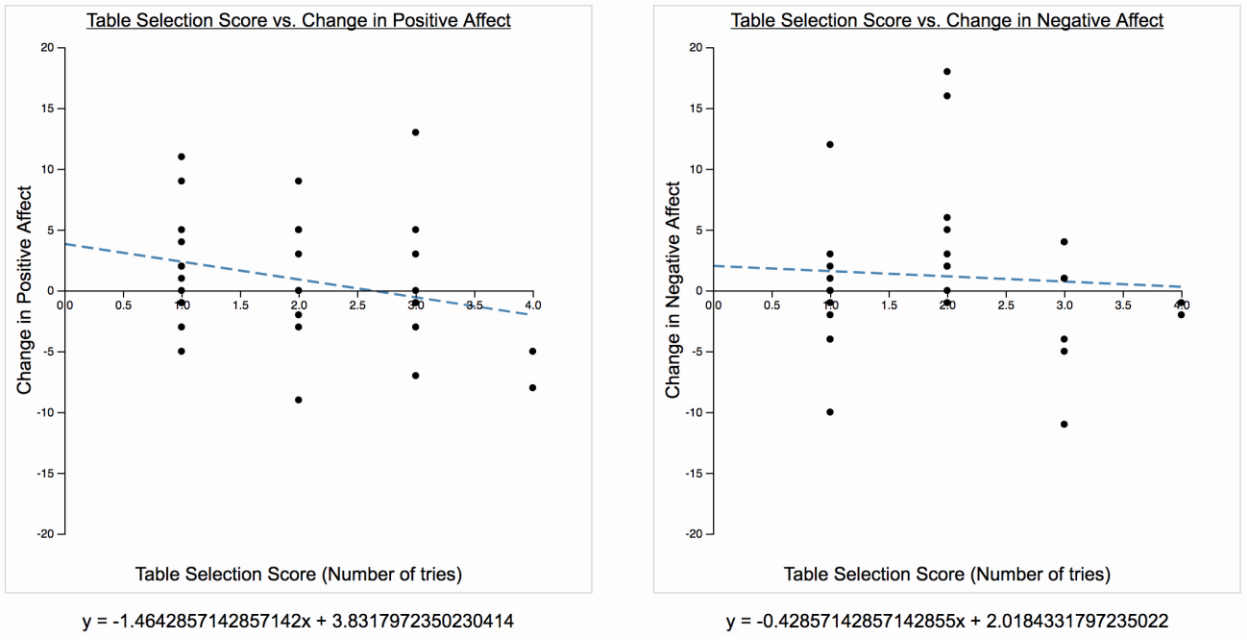


(a) **(b)**

Figure 9.6: IRI fantasy scale score vs. change in positive affect (a) and IRI fantasy scale score vs. change in negative affect (b)

The relationship between users' IRI fantasy scale scores and their change in positive affect is shown in Figure 9.6.a. The negative slope (m=-0.004) of the trend line suggests that players who score higher in their fantasy scale score are more likely to experience a decrease in negative affect. This slope is so small, however, that this result is negligible. Additionally, the data set itself is widely scattered, making the data less valid.

Figure 9.6.b displays the comparison of the users' fantasy score against their change in negative affect. The linear regression line's slope of -0.342 suggests that users with higher fantasy scale scores are more likely to experience a decrease in negative affect.



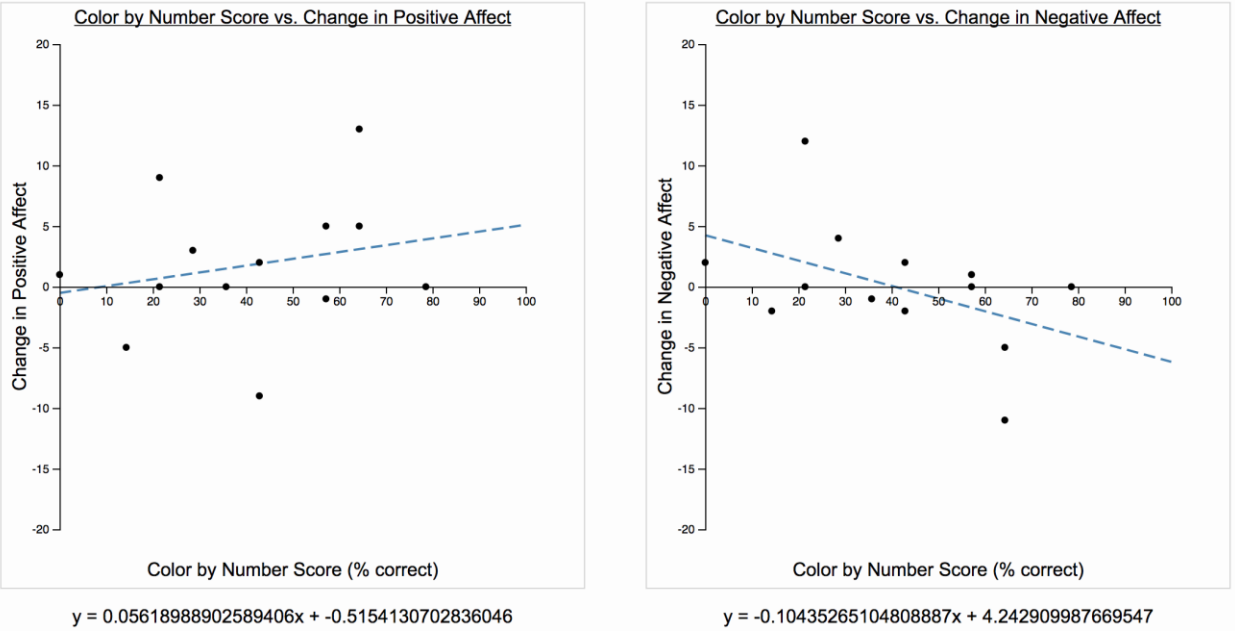
(a) **(b)**

Figure 9.7: Table selection score vs. change in positive affect (a) and table selection score vs. change in negative affect (b)

The specific scores that users' received in the different puzzles in the game were plotted against their change in positive affect and negative affect. These values were compared in order to discern whether or not the users' performance directly affected their mood. The Figures in 9.7 correspond to the table selection activity in level 1 of the game. Since the recorded grade of the user for this activity was the number of tries it took to select the correct table, a higher score corresponds to lower performance.

Figure 9.7.a shows the relationship between the users' table selection score and their change in positive affect. The linear regression line (m=-1.464) indicates that users who performed poorly in the table selection task were more likely to see a decrease in positive affect.

The relationship between the users' table selection score and their change in negative affect is shown in Figure 9.7.b. The slope of this scatter plot's corresponding trend line is -0.429, which suggests that users who performed more poorly in the table selection task had a greater chance of experiencing a decrease in negative affect. The data points are not entirely consistent with the trend line, however, making this result less valid.



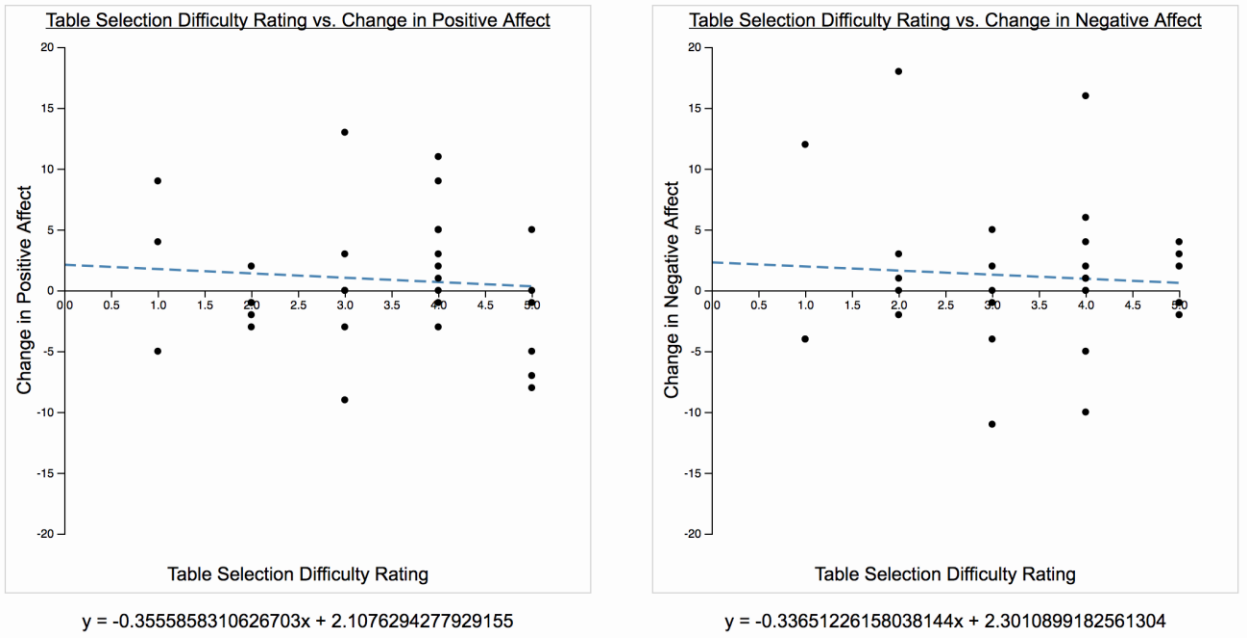
(a) **(b)**

Figure 9.8: Color by number score vs. change in positive affect (a) and color by number score vs. change in negative affect (b)

The scatter plots in Figure 9.8 display the relationship between the users' score in the color by number task and their change in mood. The scores for this task were recorded as percentages, therefore, higher numbers in the color by number score equate to better performance.

The scatter plot shown in Figure 9.8.a displays the relationship between the users' color by number task score and their change in positive affect. The trend line's positive slope (m=0.056) indicates that users who performed better in this task were likely to experience a greater increase positive affect.

Figure 9.8.b plots the relationship between the users' color by number task scores and their change in negative affect. Users who performed poorly in this task were more likely to experience a greater increase in negative affect, as suggested by the corresponding trend line's negative slope (m=-0.104).

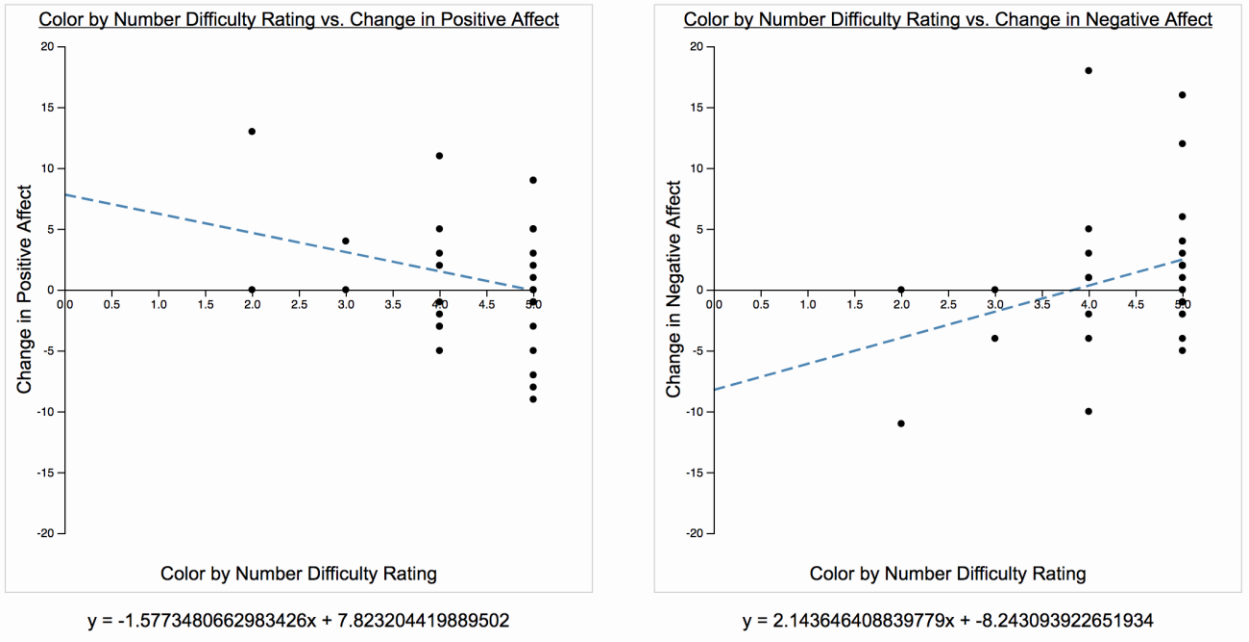


(a) **(b)**

Figure 9.9: Table selection difficulty rating vs. change in positive affect (a) and table selection difficulty rating vs. change in negative affect (b)

The users' subjective rating of the difficulty of each task was compared against their change in mood. After playing the game, users were asked to rate how hard they thought a specific task was on a scale from one to five. A value of one corresponded with "easy", while a value of 5 corresponded with "hard". The graphs in Figure 9.9 show the comparison of the users' difficulty rating of the table selection task and their change in mood.

Figure 9.9.a displays the relationship between the users' difficulty rating for the table selection task and their change in positive affect. The negative slope of the line of linear regression (m=-0.356) suggests that users who thought the task was harder were more likely to experience a decrease positive affect. This same result can be discerned from Figure 9.9.b, which shows the relationship between the users' difficulty rating for the table selection activity and their change in negative affect. The data for both graphs, however, do not consistently fit their respective trend lines. Therefore, the data displayed in these scatter plots are less valid due to lesser accuracy and precision.

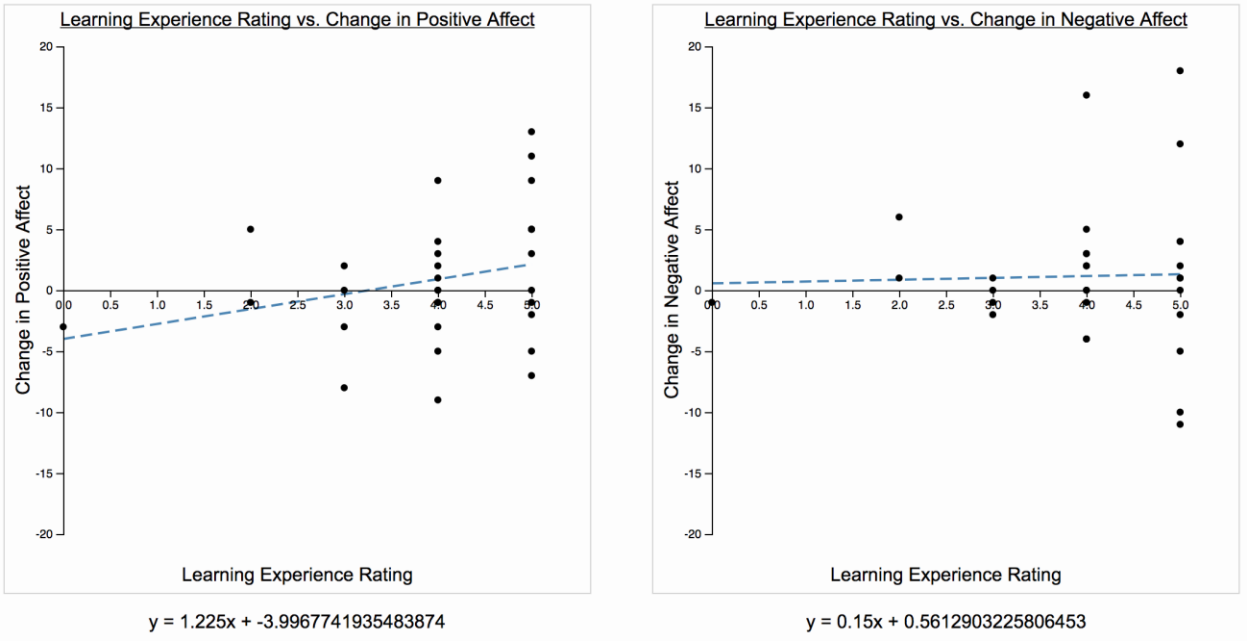


(a) (b)

Figure 9.10: Color by number difficulty rating vs. change in positive affect (a) and color by number difficulty rating vs. change in negative affect (b)

The scatter plot shown in Figure 9.10.a plots the relationship between the user's difficulty rating for the color by number task and their change in positive affect. The linear regression line's slope value of -1.577 implies that users who found the task to be more difficult experienced a greater decrease in positive affect.

Figure 9.10.b shows the relationship between the user's difficulty rating for the color by number task and their change in negative affect. The slope of the trend line (m=2.144) indicates that players who found the task to be harder saw a greater increase in negative affect.



(a) **(b)**

Figure 9.11: Learning experience rating vs. change in positive affect (a) and Learning experience rating vs. change in negative affect (b)

After completing the first level of the game, users were asked if they felt they learned more about the colorblind experience. They answered this question by rating how much they learned on a scale from one to five. One corresponded with not learning anything, while five corresponded with learning a lot. These subjective ratings were plotted against the users' change in mood in order to derive any possible correlations between the two variables.

Figure 9.11.a displays the relationship between the users' rating of their learning experience and their change in positive affect. The linear regression line corresponding to this graph has a slope of 1.225. This indicates that users who felt they learned more about the colorblind experience after playing the game felt a higher increase in positive affect.

Figure 9.11.b show the relationship between the users' rating of their learning experience and their change in negative affect. The slope of the corresponding trend line (m=0.15) suggests that users who felt they learned more about the colorblind experience felt a greater increase in negative affect.

10. Discussion

Our original hypothesis was that positive affect would decrease and negative affect would increase after playing the game due to the stress. This stress would have been the result of experiencing simulated color blindness while making color-based decisions and receiving negative feedback for incorrect actions. The initial data we examined contradicted our hypothesis, but further analysis showed results that supported our original claim. There were also multiple relationships examined that led to no legitimate conclusion. Many of these contradictions and dead-ends could be explained by a multitude of potential limitations pertaining to the project study.

10.1. Explanation of results

When comparing the PANAS scores of users before and after they played the game the results contradicted our original hypothesis. We stated that positive affect would decrease and negative affect would increase due to stress. The data showed, however, that while positive affect did slightly decrease among players, negative affect decreased significantly more after playing the game. While positive and negative affect are independent of each other, a potential distressor was still able to lessen negative mood, an atypical result. Our team discerned that this result was a due to players being interested in the game or its concept. This interest coupled with the stress induced by the game would explain our unique results. The decrease in positive affect was lessened due to stress being reduced by interest, while interest overwhelmingly influenced the decrease in negative affect. Further analyses were performed after this first comparison in order to attempt to understand these contradictions and discover correlations that matched our original hypothesis.

The original purpose of recording IRI scores was to discern if a person's dispositional empathy had any effect on their change in positive or negative affect after playing the game. Since all four subcategories of the IRI scoring system represented greater empathy of a specific type (e.g. personal distress) for users who scored higher in those areas, we took the summation of those scores and compared it to the change in PANAS scores first. Then, the scores for the four separate subcategories of empathy were compared to the PANAS scores. While some correlations made sense (e.g. players who had a higher IRI personal distress score had a greater increase in negative affect), a majority of them were unreliable due to invalid data. As a result, no significant conclusions could be drawn from comparing change in affect to dispositional empathy.

The next step in the analysis process was to determine whether or not the user's results directly affected their change in mood. When plotting the table selection scores against changes in positive affect, the result matched our hypothesis: people who performed poorly saw a decrease in positive affect, most likely due to stress. Negative affect, however, decreased, contradicting our hypothesis. This could be explained by the type of feedback that user received when picking the incorrect table. While they were clearly informed that their choice was incorrect, the audio feedback received was the teacher gently informing them that their choice

was wrong, not a reprimand. This type of gentle statement could actually make the user feel marginally better and decrease their negative mood, while the stress of having to choose the correct table could still decrease their positive affect.

Comparing the users' color by number scores to their change in mood showed promising results. The data showed that users who scored higher in the color by number task experienced a decrease in positive mood and increase negative mood. This result matches our hypothesis and is most likely due to two factors. First, this task is the hardest task in level one, so it should have definitely caused a greater amount of stress than the table selection task. The second factor is the type of feedback that the user received. The color by number task was designed to give three types of feedback depending on the score the user received: positive feedback, slightly embarrassing feedback, and extremely embarrassing feedback. Players with lower scores received the extremely embarrassing feedback, while players with higher scores receive the positive feedback. This feedback is the last scene the player encounters before the level ends and they are asked to complete the next PANAS. With this information, it would make sense that users experienced a decrease in positive affect and an increase in negative affect for performing poorly. The reaction they experienced for being incorrect is purposely negative and meant to cause stress. This reaction occurring at the end of the level also leaves the stress induced from it freshly imprinted in their mind as they complete the second PANAS.

We also made an effort to determine if perceived difficulty of a task affected the users' change in mood. Similar to the table selection score analysis, a decrease in both positive affect and negative affect was viewed when comparing subjectively rated difficulty for the table selection task and changes in affect. We believe that this result is also due to the feedback received by teacher. While the user is stressed about making incorrect choices, the teacher's gentle feedback is not severe enough to cause an increase in negative feedback. In fact, the perceived kindness of the feedback could leave users to believe that the task is not as hard as it actually is, further decreasing the stress induced from the activity.

The relationship between perceived difficulty of the color by number task and the users' change in mood was the same as comparing the tasks' scores to changes in affect. Players who thought that the puzzle was more difficult were more likely to experience a decrease in positive affect and an increase in negative affect. This is most likely due to the same factor that caused this pattern when comparing scores and mood: the more severe feedback. Not only that, but the score the user receives for this activity is displayed at the end of the first level, reminding users of how well, or poorly, they performed the task. All of this feedback can make the task seem much harder, which is extreme considering the color by number task is the hardest task of the first level.

Players were also asked how much they felt they learned about the colorblind experience in the Post-Survey. We decided to compare this metric to the changes in positive and negative affect in order to see if the induced stress from the game would be incorporated into how much the user felt they learned. Users who said they learned more showed an increase in both positive and negative affect. This is probably due to the fact that they enjoyed

the concept of experiencing the colorblind perspective, increasing positive affect, but also became stressed through the experience, increasing negative affect.

10.2. Potential limitations

There were a multitude of potential limitations that were considered before collecting data, and others that were considered after viewing the initial data sets. These limitations were taken into account when constructing the study in order to minimize their effects on the data. It was important, however, that these limitations were recognized and considered when viewing the data and any outliers within it.

It was entirely possible that the game, which should have decreased the users' positive mood as a potential stressor, actually caused the positive affect to increase. This was seen in the data as well, where the positive affect of some users actually increased after playing the game. This could have been due to the users' interest in the game itself and the concept behind it. With descriptors such as "interested" and "excited", a user who thought that the game or its message was enjoyable could experience higher scores with these words and other positive emotions while completing the PANAS. Additionally, if a player achieved greater success in the puzzles in the game, it could result in an increase in their positive emotions and also cause their positive affect to increase through that medium.

Besides the game, the surveys were also considered as part of the potential limitations of the study. If a user became bored or thought the surveys were lengthy or unnecessary, their positive mood could have decreased and negative mood may have increased. The same effect could be achieved when considering the user was unaware of how many surveys would be asked to take before playing the game. While this change in mood was the hypothesized result, the surveys were not the predicted causation. When users became so opposed to taking the surveys that they quit the study before completing it, it made a number of data points useless that required comparison between two survey results (e.g. 8 users quit the study before completing the second PANAS).

The environment that the user experienced the study in could have also had a large effect on the data. Since the application was created as a web-based game, the link to a working version of it was made available via Heroku. This web-based feature, while convenient, also meant that a user could play in any environment they desired so long as they had a computer with a connection to the internet. This variation in environmental factors could have resulted in less precise or accurate data.

The computer itself is another part of the environmental factors that could negative affect the data from the study. At times, the game required a sizable amount of resources from the CPU. This would cause the machine the game is running on to slow down if the there were too many CPU-heavy operations that had to completed. This could have decreased the positive affect as compared to the first PANAS because the user became impatient with the study.

11. Project Promotion

Once the game was far enough along that the first level was playable, it was decided that it was time to start promoting the game so that others could play it and provide feedback on what had been made so far. The first place that the project was promoted was at WPI's Alpha Fest in which the alpha versions of games were set up in a small area where people could come and freely try the games there while the developers observed or played games themselves. Here, a version of the game that contained a mostly complete first level was set up on a laptop for people to try with it being advertised as a game that "simulated what it's like to be colorblind."

The second way that the project was promoted was through sending out a link to the friends and families of all group members that would let them play the game on their own computer in the Chrome web browser. This version of the game contained a more complete version of the first level as well as surveys that those who played would be asked to take before and after playing. This way was done in order for a large number of people to play the game as well as provide feedback on it without any of the group members needing to be present for it.

The third way that the project was promoted was by having it tested by an IMGD class that one of the project advisors was teaching at the time. This was done in order to get feedback from those who understood how games work and would be able to provide accurate information on any bugs that were found. The version of the game that was presented to them was the same version that was sent out in the link mentioned above.

The final way that the project was promoted was through having it displayed at WPI's booth at PAX East, a large gaming convention where consumers come from around the world to play the games on display there. This was done to get the concept and name of the game out to the general public as thousands of people would be present at the convention. This allowed for the game to be seen by a large number of people in a very short period of time, serving as a great way to promote it. The version of the game that was provided for this promotion was a fully complete version of the first level *without* any surveys before or after it.

12. Conclusion

The goal of *Unseeable* was to allow non-colorblind individuals to empathize with colorblind persons by seeing through their perspective and understanding some of the struggles they endure in their everyday lives. We determined that the game should induce an ample amount of stress in order to achieve this goal because color-blind persons experience distress when coping with their impairment. In order to discern whether or not the game induced stress, we recorded the users' mood as a state before and after the game in order to examine their change in mood through positive affect and negative affect. A number of other metrics were recorded in order to view the relationship between them and any changes in mood that could relate to our experience goal.

When comparing the changes in positive affect and negative affect after playing the game, the results contradicted our hypothesis. Positive affect did decrease slightly for players, but negative affect decreased by a larger amount. Upon further analyses, however, we found that plotting the change in affect against the users' color by number task score and their perceived difficulty of the task yielded the results we desired. Players with lower scores that thought the game was more difficult experienced a decrease in positive affect and an increase in negative affect, matching our original hypothesis.

These two correlations were the most important in supporting our hypothesis because this task was the main puzzle of the first level. The color by number task was meant to be the hardest color-based puzzle in level one. The feedback the user received from the game was meant to be of major importance as well, being the only puzzle in the level where the score was reported back to the user. The audio feedback the player receives is also the harshest in content, making it the task that had the highest potential to induce the most stress in a player. It is for these reasons that the correlations between the color by number task's score and its perceived difficulty with the change in positive affect and negative affect held the most value to our team. Other correlations were drawn from analyses of the data, but the data for a large majority of these correlations were invalid, making results stemming from them unreliable.

Based on the correlations in the data found in comparing the users' color by number score and their difficulty rating of the task with change in positive affect and negative affect, we determined that the game had accomplished its experience goal. Through the main section of the game experience (i.e. the color by number task), players experienced induced stress that indicated to our team that they were struggling with color identification, similar to the colorblind population. By feeling those same emotions, the non-colorblind participants were able to empathize with colorblind people, which was our original intent. As a result, we consider the game and study to be a success.

References

Capan, T. (). Why the hell would I use node.js? A case-by-case tutorial. Retrieved from https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Crunchbase.Heroku. Retrieved from https://www.crunchbase.com/organization/heroku

Davis, M. H. (1983). Measuring individual differences in empathy: Evidence for a multidimensional approach. *Journal of Personality and Social Psychology, 44*(1), 113-126. 10.1037/0022-3514.44.1.113 Retrieved from https://search.proquest.com/docview/1295920010

Holowaychuk, T. J.Express.js. Retrieved from https://expressjs.com/

Jenny, B. (2017). Color oracle. Retrieved from http://colororacle.org/

LAWRENCE, E. J., SHAW, P., BAKER, D., BARON-COHEN, S., & DAVID, A. S. (2004). Measuring empathy: Reliability and validity of the empathy quotient. *Psychological Medicine, 34*(5), 911-920. 10.1017/S0033291703001624 Retrieved from http://journals.cambridge.org/abstract_S0033291703001624

Mark H. A. Davis. (1995). *A multidimensional approach to individual differences in empathy* Retrieved from http://data.theeuropeanlibrary.org/BibliographicResource/2000051616803

Medlock, J. (2017, March 30,). An overview of MongoDB & mongoose. Retrieved from https://medium.com/chingu/an-overview-of-mongodb-mongoose-b980858a8994

Miller, A. (2014, August 20,). 5 things people who are colorblind (and their doctors) want you to know. Retrieved from https://health.usnews.com/health-news/health-wellness/articles/2014/08/20/5-things-people-who-are-colorblind-and-their-doctors-want-you-to-know

MongoDB. MongoDB and MySQL compared. Retrieved from https://www.mongodb.com/compare/mongodb-mysql

Munro, J. (2017). An introduction to mongoose for MongoDB and node.js. Retrieved from https://code.tutsplus.com/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527

N.a. (a). About heroku. Retrieved from https://stackoverflow.com/tags/heroku/info

N.a. (b). BSON specification. Retrieved from http://bsonspec.org/

N.a. (c). mLab documentation. Retrieved from https://docs.mlab.com/

N.a. (d). MongooseJS documentation. Retrieved from http://mongoosejs.com/docs/guide.html

N.a. (e). Node.js introduction. Retrieved from https://www.w3schools.com/nodejs/nodejs_intro.asp

N.a. (2015). Facts about color blindness. Retrieved from https://nei.nih.gov/health/color_blindness/facts_about

N.a. (2016). Colour blind awareness. Retrieved from http://www.colourblindawareness.org/

Rachowicz, J. (2017, February 23,). When, how and why use node.js as your backend. Retrieved from https://www.netguru.co/blog/use-node-js-backend

Shan, P. (2015, June 7,). Mongoose vs mongodb native driver – what to prefer? Retrieved from http://voidcanvas.com/mongoose-vs-mongodb-native/

Simoneau, L. (2010, July 10,). Node.js is the new black. Retrieved from https://www.sitepoint.com/node-js-is-the-new-black/

Spreng, R. N., McKinnon, M. C., Mar, R. A., & Levine, B. (2009). The toronto empathy questionnaire: Scale development and initial validation of a factor-analytic solution to

multiple empathy measures. *Journal of Personality Assessment, 91*(1), 62-71. 10.1080/00223890802484381 Retrieved from http://www.tandfonline.com/doi/abs/10.1080/00223890802484381

Stueber, K. (2013). Measuring empathy. Retrieved from https://plato.stanford.edu/entries/empathy/measuring.html

Sukin, I. (2013). *Game development with three.js*. Livery Place, 35 Livery Street, Birmingham, UK: Packt Publishing Ltd.

Watson, D., Clark, L. A., & Tellegen, A. (1988). Development and validation of brief measures of positive and negative affect. *Journal of Personality and Social Psychology, 54*(6), 1063-1070. 10.1037/0022-3514.54.6.1063 Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/3397865

Xplenty. (2017, September 28,). The SQL vs NoSQL difference: MySQL vs MongoDB. Retrieved from https://medium.com/xplenty-blog/the-sql-vs-nosql-difference-mysql-vs-mongodb-32c9980e67b2

Appendices

Appendix A: Informed Consent

Informed Consent Agreement for Participation in a WPI Research Study

Investigator: Brian Moriarty, IMGD Professor of Practice

Contact Information:

Brian Moriarty, bmoriarty@wpi.edu, 508 831-5638
Isaiah Cochran, ilcochran@wpi.edu, 860 995-4606
Alex Horton, aihorton@wpi.edu, 617 640-3204
Drew Tisdelle, ddtisdelle@WPI.EDU, 508 965-0927
Tommy Trieu, ttrieu@wpi.edu, 860 716-8498

Title of Research Study: *Unseeable*

Sponsor: WPI

Introduction: You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study: The purpose of this study is to obtain playtest feedback in order to locate/address operational bugs, to identify opportunities for design improvement, and to gather data to conduct statistical analyses on to measure games effectiveness towards the experience goal.

Procedures to be followed: You will be asked to play a brief game lasting less than ten minutes. Instrumentation in the game software will anonymously record your activity during play. Before and after completing the game, you will be asked to complete brief, anonymous surveys describing your subjective experience, a positive and negative affect scale, and an interpersonal reactivity index.

Risks to study participants: There are no foreseeable risks associated with this research study.

Benefits to research participants and others: You will have an opportunity to enjoy and comment on a new game under active development. Your feedback will help improve the game experience for future players.

Record keeping and confidentiality: The only instance of your personal information being recorded will be whether or not you are colorblind. Records of your participation in

this study will be held confidential so far as permitted by law. However, the study investigators and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to this confidential data. Any publication or presentation of the data will not identify you.

Compensation or treatment in the event of injury: There is no foreseeable risk of injury associated with this research study. Nevertheless, you do not give up any of your legal rights by signing this statement.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact the Investigator listed at the top of this form. You may also contact the IRB Chair (Professor Kent Rissmiller, Tel. 508-831-5019, Email: kjr@wpi.edu) and the University Compliance Officer (Jon Bartelson, Tel. 508-831-5725, Email: jonb@wpi.edu).

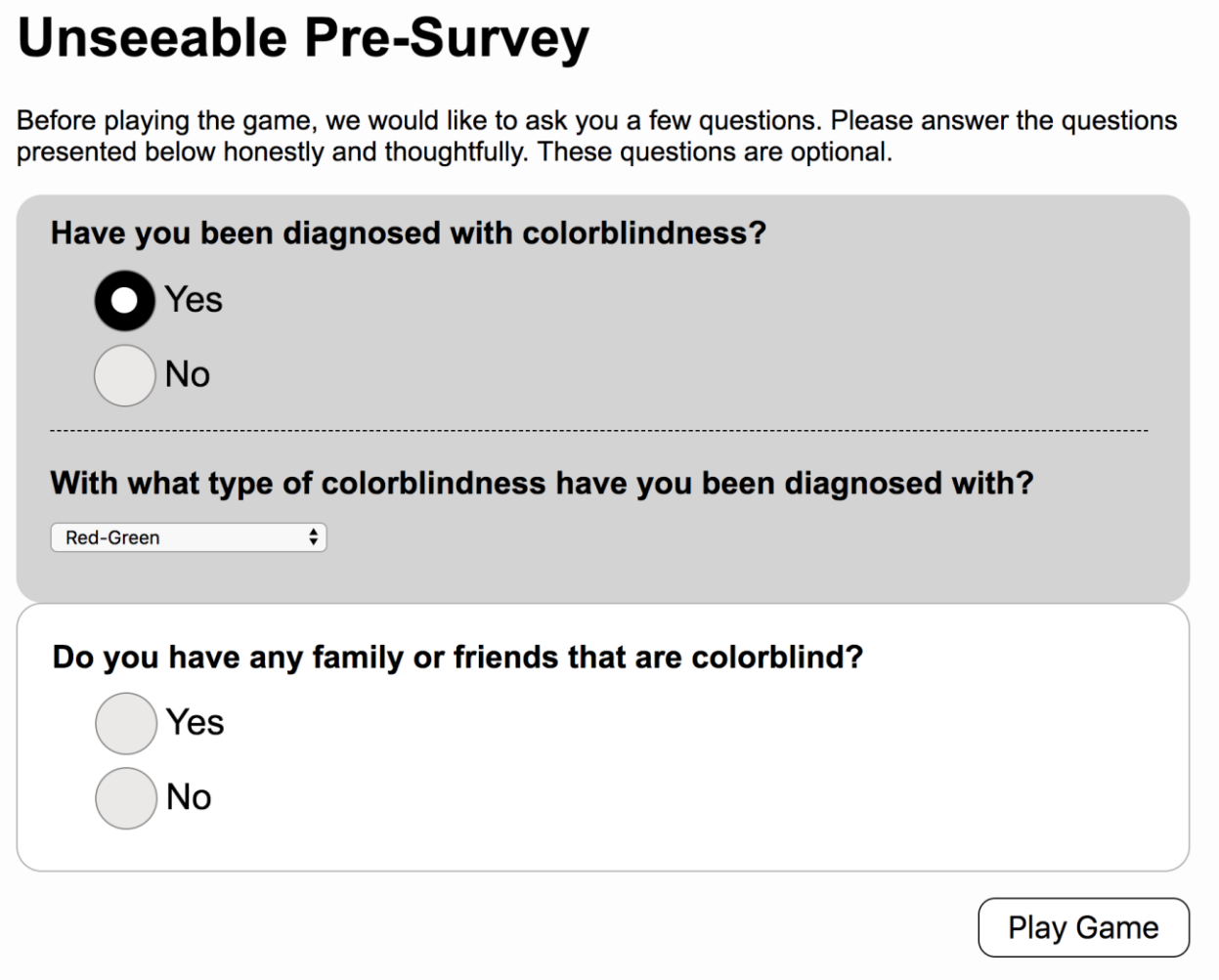
Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

_____ Date: _____
Study Participant Signature

_____
Study Participant Name (Please print)

Appendix B: Pre-Survey



Pre-survey with 'Yes' as the answer to question one.

Unseeable Pre-Survey

Before playing the game, we would like to ask you a few questions. Please answer the questions presented below honestly and thoughtfully. These questions are optional.

Have you been diagnosed with colorblindness?

- ○ Yes
- ◉ No

Have you ever thought about what it's like to be colorblind?

- ○ Yes
- ○ No

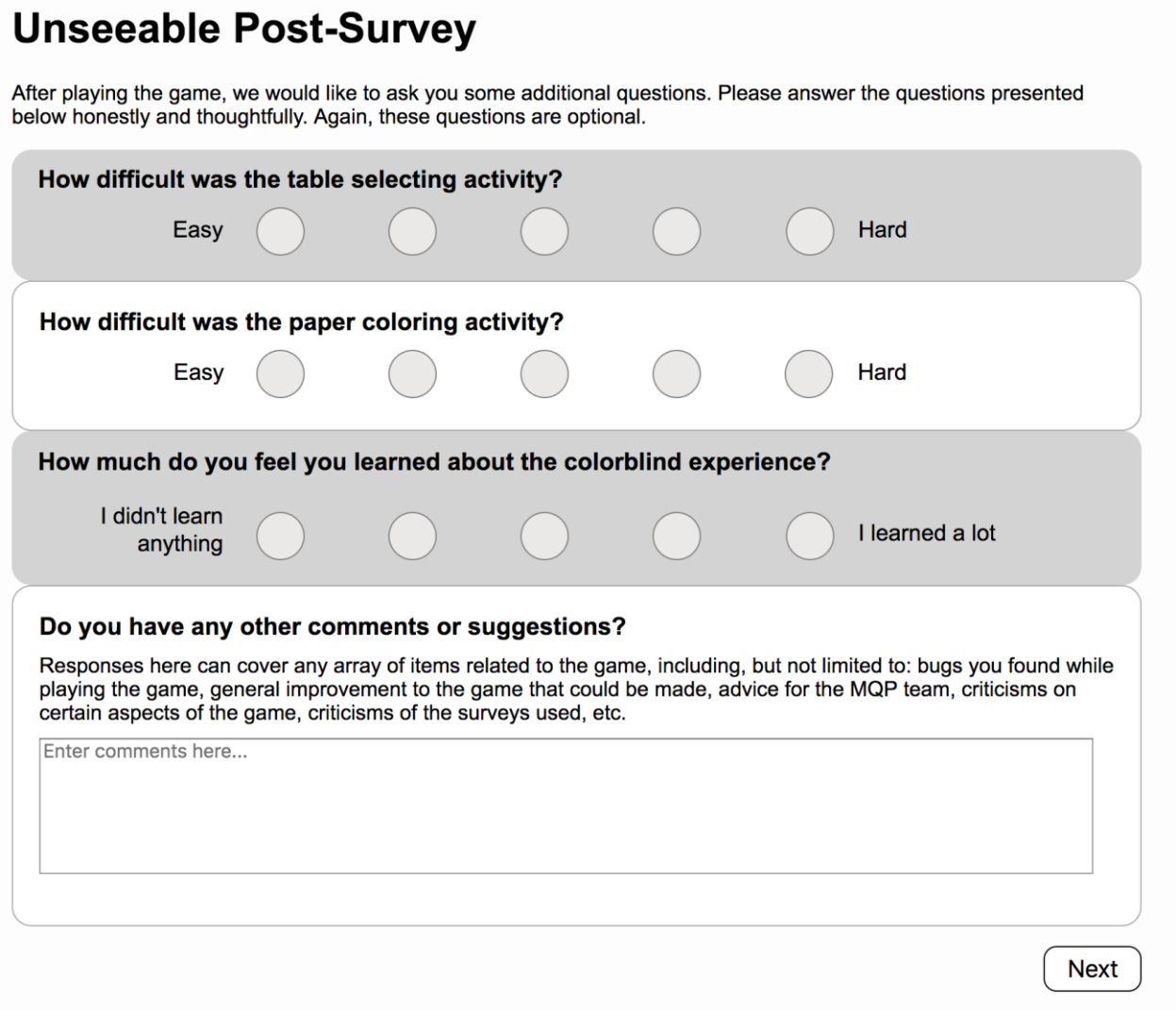
Do you have any family or friends that are colorblind?

- ○ Yes
- ○ No

Play Game

Pre-survey with 'No' as the answer to question one.

Appendix C: Post-Survey



Unseeable Post-Survey

After playing the game, we would like to ask you some additional questions. Please answer the questions presented below honestly and thoughtfully. Again, these questions are optional.

How difficult was the table selecting activity?

Easy ○ ○ ○ ○ ○ Hard

How difficult was the paper coloring activity?

Easy ○ ○ ○ ○ ○ Hard

How much do you feel you learned about the colorblind experience?

I didn't learn anything ○ ○ ○ ○ ○ I learned a lot

Do you have any other comments or suggestions?

Responses here can cover any array of items related to the game, including, but not limited to: bugs you found while playing the game, general improvement to the game that could be made, advice for the MQP team, criticisms on certain aspects of the game, criticisms of the surveys used, etc.

Enter comments here...

Next

Appendix D: PANAS

Worksheet 3.1 The Positive and Negative Affect Schedule (PANAS; Watson et al., 1988)

PANAS Questionnaire

This scale consists of a number of words that describe different feelings and emotions. Read each item and then list the number from the scale below next to each word. **Indicate to what extent you feel this way right now, that is, at the present moment *OR* indicate the extent you have felt this way over the past week (circle the instructions you followed when taking this measure)**

1	2	3	4	5
Very Slightly or Not at All	A Little	Moderately	Quite a Bit	Extremely

_____ 1. Interested	_____ 11. Irritable
_____ 2. Distressed	_____ 12. Alert
_____ 3. Excited	_____ 13. Ashamed
_____ 4. Upset	_____ 14. Inspired
_____ 5. Strong	_____ 15. Nervous
_____ 6. Guilty	_____ 16. Determined
_____ 7. Scared	_____ 17. Attentive
_____ 8. Hostile	_____ 18. Jittery
_____ 9. Enthusiastic	_____ 19. Active
_____ 10. Proud	_____ 20. Afraid

Scoring Instructions:

Positive Affect Score: Add the scores on items 1, 3, 5, 9, 10, 12, 14, 16, 17, and 19. Scores can range from 10 – 50, with higher scores representing higher levels of positive affect. Mean Scores: Momentary = 29.7 (*SD* = 7.9); Weekly = 33.3 (*SD = 7.2)*

Negative Affect Score: Add the scores on items 2, 4, 6, 7, 8, 11, 13, 15, 18, and 20. Scores can range from 10 – 50, with lower scores representing lower levels of negative affect. Mean Score: Momentary = 14.8 (*SD* = 5.4); Weekly = 17.4 (*SD* = 6.2)

Appendix E: IRI

INTERPERSONAL REACTIVITY INDEX

The following statements inquire about your thoughts and feelings in a variety of situations. For each item, indicate how well it describes you by choosing the appropriate letter on the scale at the top of the page: A, B, C, D, or E. When you have decided on your answer, fill in the letter next to the item number. READ EACH ITEM CAREFULLY BEFORE RESPONDING. Answer as honestly as you can. Thank you.

ANSWER SCALE:

A	B	C	D	E
DOES NOT DESCRIBE ME ME WELL				DESCRIBES VERY WELL

1. I daydream and fantasize, with some regularity, about things that might happen to me. (FS)

2. I often have tender, concerned feelings for people less fortunate than me. (EC)

3. I sometimes find it difficult to see things from the "other guy's" point of view. (PT) (-)

4. Sometimes I don't feel very sorry for other people when they are having problems. (EC) (-)

5. I really get involved with the feelings of the characters in a novel. (FS)

6. In emergency situations, I feel apprehensive and ill-at-ease. (PD)

7. I am usually objective when I watch a movie or play, and I don't often get completely caught up in it. (FS) (-)

8. I try to look at everybody's side of a disagreement before I make a decision. (PT)

9. When I see someone being taken advantage of, I feel kind of protective towards them. (EC)

10. I sometimes feel helpless when I am in the middle of a very emotional situation. (PD)

11. I sometimes try to understand my friends better by imagining how things look from their perspective. (PT)

12. Becoming extremely involved in a good book or movie is somewhat rare for me. (FS) (-)

13. When I see someone get hurt, I tend to remain calm. (PD) (-)

14. Other people's misfortunes do not usually disturb me a great deal. (EC) (-)

15. If I'm sure I'm right about something, I don't waste much time listening to other people's arguments. (PT) (-)

16. After seeing a play or movie, I have felt as though I were one of the characters. (FS)

17. Being in a tense emotional situation scares me. (PD)

18. When I see someone being treated unfairly, I sometimes don't feel very much pity for them. (EC) (-)

19. I am usually pretty effective in dealing with emergencies. (PD) (-)

20. I am often quite touched by things that I see happen. (EC)

21. I believe that there are two sides to every question and try to look at them both. (PT)

22. I would describe myself as a pretty soft-hearted person. (EC)

23. When I watch a good movie, I can very easily put myself in the place of a leading character. (FS)

24. I tend to lose control during emergencies. (PD)

25. When I'm upset at someone, I usually try to "put myself in his shoes" for a while. (PT)

26. When I am reading an interesting story or novel, I imagine how I would feel if the events in the story were happening to me. (FS)

27. When I see someone who badly needs help in an emergency, I go to pieces. (PD)

28. Before criticizing somebody, I try to imagine how I would feel if I were in their place. (PT)

Appendix F: Technical Definitions Glossary

Audio - a Three.js global audio object

AudioBuffer - objects designed to hold snippets of audio

AudioListener - virtual listener of audio effect in the scene

AudioLoader - class for loading an AudioBuffer

BoundingBox - Smallest possible box that contains the entirety of a geometry

BSON - short for binary JSON, a binary-encoded serialization of a JSON like document

Clock - keeps track of time

FBX (Filmbox) - file format that allows import and export of files between 3d animation software

Geometry - vertices, faces, colors, etc of 3D objects

Group - group of objects in Three.js used to manipulate multiple object at once

JSON - JavaScript Object Notation, data format used very often in browser-server communication

Material - describe the appearance of an geometry, such as texture and reflectiveness

Mesh - the combination of a geometry and material

MeshBasicMaterial - simple material for drawing geometries in a flat way

Mousedown event - fired when a pointing device is pressed on an element

Mousemove event - fired when a pointing device is moved over an element

Npm package - folder containing a program described by a “package.json” file

POST request - HTTP request method that requests that a server accept the data enclosed in the message body

Raycasting - casting a ray outward from the camera, intersection of the ray with objects is used to determine what the mouse is over

SessionStorage - a property of the browser where values can be saved for as long as the browser window is open.

Scene - a Three.js object that allows users to set up what and where is to be rendered (e.g. objects, lights, and cameras).

Spline - smooth curve created from a series of points, used for camera movement

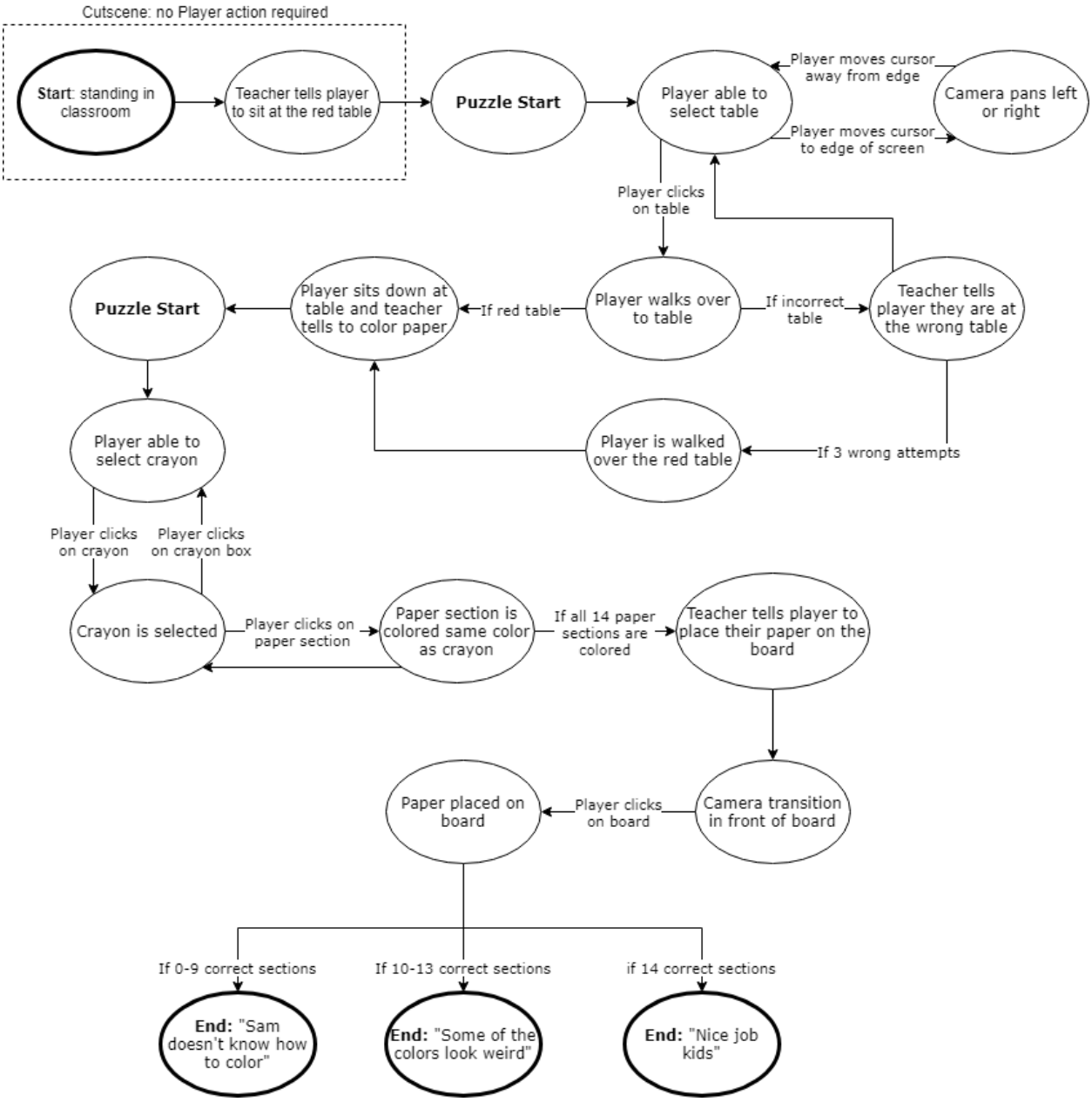
Tween - used in animation to generate intermediate values between two values to give the appearance of smooth motion

UUID/GUID - Universally/Globally Unique Identifier a 128-bit number used to uniquely identify objects in programs.

Window - the browser’s window.

Appendix G: Game Flow Diagrams

Level 1 Game Flow Diagram



Level 2 Game Flow Diagram