CS4624
Multimedia, Hypertext, and Information Access

December 13, 2022

KidDataViz

Virginia Tech, Blacksburg, VA 24061

Instructor: Dr. Edward Fox
Client: Dr. Sally Hamouda
Team: Sasha Holt, Andrew Ahn, Joshua Yang, Heechan Lim, Timothy Kelley, Jack Homer

Table of Contents

Table of Tables

Table of Figures

1.0 Abstract

Data helps us to understand the world around us. Not only is interpreting data important, but understanding how to communicate data is an essential skill in the modern world. Teaching children how to make and understand data visualizations lays a solid foundation for their critical thinking and understanding of large-scale problems. This project aims to teach elementary school students how to visualize data engagingly and effectively.

Building on this goal, our project was to develop an accessible website with the ability to host seven or more different game implementations designed around data visualization concepts. These games target three groups of school levels: 1st- and 2nd-grade, 3rd- and 4th-grade, and 5th-grade and up. The goal of the games is to break down complex data visualization concepts for various levels of understanding. Consequently, each game is designed to be fun and replayable, so children engage with the website for longer periods and learn more.

The website has been fully implemented and is accessible to the public. This implementation allows new developers to add games easily, assuming they are familiar with web development. Additionally, we have implemented seven games in either JavaScript or Unity, each of which is playable from the website. We have conducted testing for the application, via a digital feedback form provided to testers. The feedback given on these forms is used to improve the website and the games that are hosted on it.

The website features a mobile dropdown menu, an introductory page, a page for feedback, and seven games that teach core concepts of data visualization. The website can be used in classroom settings as an easy way for teachers to introduce data visualization to students.

2.0 Introduction

2.1 Objective

The primary objective is to make a tool to teach elementary school students about data visualization. To achieve this objective, we decided to use games to grab kids' attention and then integrate lessons on data visualization by displaying data from the games. To accomplish this, we decided to create a website that hosts several games, targeting a variety of elementary school grades.

2.2 Deliverables

The core deliverable for this project is a web tool oriented around quick and appealing lessons for elementary school students about data visualization. We aim to do this by presenting the following deliverables:

- 6-8 unique games at a variety of levels of difficulty based on age ranges,
- A website that hosts all the games and provides text-based information about data visualization.

2.3 Client

The client for the project is Dr. Sally Hamouda. Dr. Hamouda is a Virginia Tech Collegiate Associate Professor in the Department of Computer Science. Her research focus is digital education. Dr. Hamouda is our source of guidance throughout the project, providing us with ideas, information, and resources. This project will be delivered to Dr. Edward Fox, a Virginia Tech Professor in the Department of Computer Science, by the end of the semester as he is the professor for CS 4624.

2.4 Team

Our team includes Sasha Holt, Andrew Ahn, Joshua Yang, Heechan Lim, Timothy Kelley, and Jack Homer. All team members are seniors majoring in computer science at Virginia Tech.

3.0 Requirements

The general project requirements derived from the objectives were to create multiple games with the intent they would improve children's data visualization abilities, and to create a website to host these games that is easily accessible for children.

3.1 The Games

In her specifications for the project, Dr. Hamouda mentioned that we could either focus on creating a general-use data visualization tool or on developing games tied to lessons. She showed us a website from her previous work, made with the intent of helping people with high school understanding or above to create data visualizations. The tool on the website created graphs and charts based on user-uploaded or pre-made datasets. This website is called AnyChart [1].

Our group decided to focus on game development, as we believed it would appeal more to younger children. To determine what we would require of our games, we needed to do contextual research, including finding current games and online tools focusing on data visualization for kids. We determined that engaging games involve children in the process of visualizing data, i.e. collecting data, sorting data into categories, making the visualization itself, or some combination of these methods.

Other requirements, derived from our objective, were that the rules of the game are easily understood, the tasks required of children could be performed by them, and the games were engaging visually. We also needed to tailor our games to specific age groups of children, ensuring that younger children did not attempt games that were too difficult for them and vice versa.

3.2 The Website

The core requirement of the website was for it to host games that were easily accessible by children. This requirement implies a few other UX requirements: the website should be easy to navigate, web pages should be easy to read, the information displayed should be suited to children (i.e., small amounts of text, large letters for navigation), and it should be clear which games are intended for which grades.

3.3 Timeline

The timeline for our project was determined along with the requirements and is reflected in Table 1.

Phase 1	September	• Research data visualization tools • Brainstorm simple

		game ideas ● Design website/game.
Phase 2	October	● Begin website and game development ● Implement data visualization games
Phase 3	November	● Testing and surveys conducted ● Feedback received
Phase 4	December	● Revise and re-implement design ● Finalize website

Table 1: Group Timeline

4.0 Design

Our team's design process focused primarily on using the requirements as a baseline and employing research and brainstorming to flesh the rest out. As previously described, our primary requirement was to create an accessible website with a variety of games to help children learn more about data visualization. Throughout the process, we also deliberately designed around our central audience, elementary schoolers. These two aspects of our design philosophy guided our major design choices. Our design prioritizes simple games, a simple and child-friendly interface, pleasing aesthetic and visual themes for children, ease of navigation, and games that present data suited to children.

4.1 Website Design

Our main method for designing the website was wireframing. With the help of Dr. Hamouda, we agreed on a variant and appealing color palette that would draw children's attention. From there, we decided to implement large rounded buttons into our design that would immediately draw attention and leave little room for confusion. We also designed a static toolbar, so that children would not get lost in the website and unable to navigate back to the main pages. Page wireframes were created in Figma [3], a wireframing website made for web design, to easily integrate the theme with the CSS formatting of the website on implementation. An example of the home page design is shown in Figure 1.

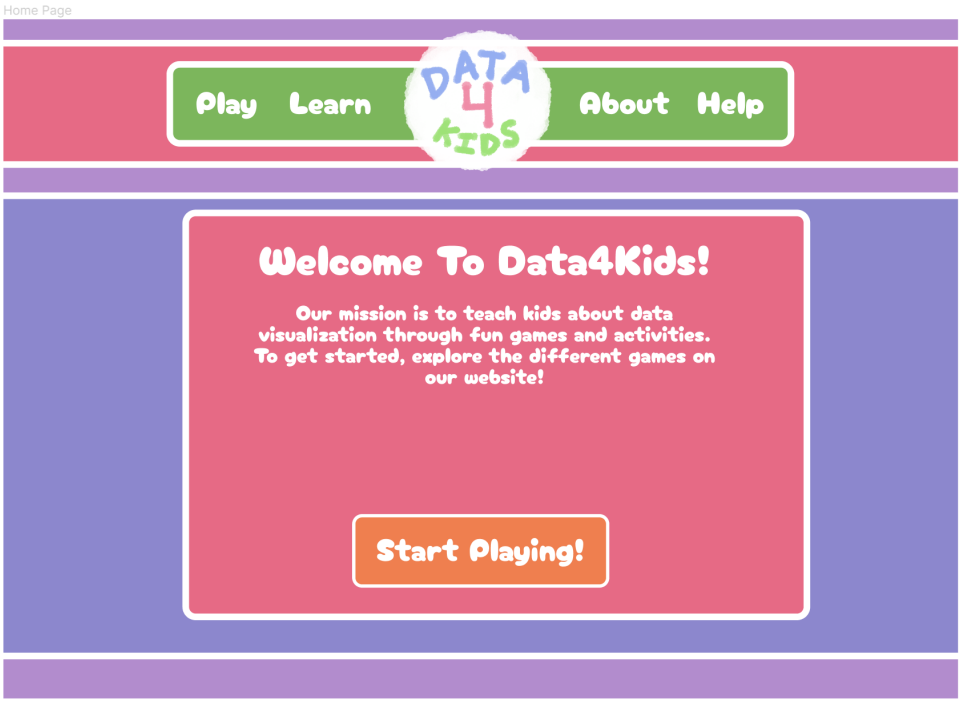

Figure 1: Figma Wireframe of Website Home Page

4.2 Games Design

As noted in the requirements section, we need games that involve children in making data visualizations. These games should be easy to use and targeted towards a specific group of elementary school levels (1st- and 2nd-grade, 3rd- and 4th-grade, and 5th-grade and older).

4.2.1 Catch Game

The Catch game is targeted towards 3rd- and 4th-grade students, thus being more challenging for younger students. The main goal of this game is to provide a fun way for children to understand and analyze bar charts. The data visualization component for this game is a bar chart that displays the number of falling blocks caught per color. The total number of blocks caught will be displayed along with an in-depth analysis of the blocks per color caught after the game is concluded.

The basic idea of the game is to catch one falling block after another by moving a paddle/bucket left and right. Once a block is missed, the game is finished. To keep the user's interest, they will be able to visualize which blocks of each color they caught after each game and start a new game after they are finished. This game is suitable for an older age range such as 3rd and 4th graders since attempting to catch a falling block with the arrow keys requires quick reaction and anticipation due to the randomization of the block positioning upon falling.

4.2.2 Cake Game

The basic idea of this game was to involve the player in creating a pie chart for them to analyze and answer questions about. More specifically, the player will create a cake with three different flavors. The flavors are listed on the side, are clickable, and can be placed onto the baking tray in the middle of the screen. Each flavor will have a number underneath it with the text “Pieces left:” to represent how many people want that flavor. After the player has filled each spot in the baking tray, they will be given a pie chart representation of the cake they made and will answer basic questions using it.

The target audience for this game is first and second-graders, which will be a big focus in implementation. A consideration for this group is to make sure the data set is straightforward; the baking tray would have a total of ten pieces and thus the sum of each flavor would add up to ten. Another necessary aspect of this design is that the visual theme should be aesthetically pleasing to the player; we want players to be more invested in the game so that they are then more invested in the learning aspects that come with the game.

4.2.3 Tower Game

The tower game is targeted toward 5th-grade and older students. The game is a bit more challenging than the ones made for lower grades. The purpose of this game is to help the target audience analyze bar charts. Therefore, the data visualization is a bar chart followed by a set of questions that involve counting and math. The interface is also simple and matches the colors and fonts of the overall website.

The basic idea of this game is to stack the tower as high as possible. There is a block that moves left and right and the player can drop the block by clicking the screen. If the block lands on the previous block, the game continues. Otherwise, the game ends. There are three rounds with increasing difficulty where the speed of the block increases respectively. At the end of the game, the game will have created a bar chart and a set of questions that can be answered.

4.2.4 Math Game

The math game is currently targeted at 1st- and 2nd-grade students with math problems consisting of addition and subtraction. The purpose of this game is to help teachers assess their student’s addition and subtraction knowledge. It can also serve the purpose of helping students visualize correct and incorrect answers through a bar chart using the dataset they create from playing the game.

The basic idea of this game is to answer a set of randomized addition and subtraction questions. Given the answer that the student types and submits, a screen will appear that notifies the student whether the answer is correct or incorrect. As the game is played out until the last question, a dataset is created based on how many questions the student answered correctly or incorrectly. At

the end of the game, a simple bar chart is produced and displays the dataset and the correlation between correct and incorrect answers.

4.2.5 Ocean Game

The ocean game is targeted towards 1st- and 2nd-grade students and therefore has a more simplified interface and concept than some of the other game designs. The purpose of this game is primarily to help children understand how graphs relate to visible, tangible data. Thus, the game was designed to be an interactive playground for children to produce an image of an ocean. The user can press on any part of the "ocean" and a fish is created. If the user presses right above the bottom of the ocean, kelp or coral is made instead. The user can press again on the fish to change colors or to change it back to open ocean. Upon completion of the ocean scene, the user can press a button in the bottom left corner to view the data in the form of a bar chart of fish, so the user can see how many of each type of fish or object was used to create their scene.

The idea of this game is to give students a chance to make something creative. A lot of games on the website are linear in how you play them while this game is very free form. Kids can spend as much time as they want on their pictures of the ocean. Then they can learn about how their actions, when creating the picture, affect the data behind the picture they created.

4.2.6 Lego Game

Like the Cake Game, the Lego Game seeks to involve children in the process of data visualization by building the actual chart; in this case, it would involve building a bar chart. The player would start with a set of Legos that are of different shapes and colors but with similar sizes. The game would involve first having them sort the Legos by color. They would then need to stack the Legos on top of each other based on the category. For example, if it was by color, all green would stack on top of the green, all blue on blue, etc. After this is finished, they are shown a bar chart of their stacks right next to each other, and possibly asked questions about the charts.

As the game is aimed at first and second-graders, it seeks to be a pretty simple game. We think the combination of sorting a data set and then making the chart is a good fit for our target elementary level. From personal experience with our targeted audience, we think Legos will be a very appealing aesthetic to them and will get them more invested in the game.

5.0 Implementation

There are a couple of key components to the project implementation. The web application was implemented to allow for fluid navigation and interfacing with the games. Additionally, each game was implemented by an individual member of the team in either Unity or JavaScript, which will be described in more detail in this section. The website and games were implemented separately and then later integrated during the final stages of the project.

5.1 Website Implementation

The website implementation is a React application based in JavaScript, HTML, and CSS. The application is hosted on Vercel through a link with our GitHub Repository.

5.1.1 Application Development

The first stage of website implementation was the development of the React application. To start, an empty React application was created and basic infrastructure was implemented. Among this infrastructure, website routing was implemented using React Router, and JSX files were created for each of the necessary pages on the website. A toolbar was then made to be fixed at the top of every page, allowing for easy navigation. Once the pages and navigation were in place, the project switched gears to filling out page content and recreating the theme and formatting of the Figma wireframes. After some design review, it was decided that the website should be more accessible from mobile devices and tablets, which resulted in a redesign of the main toolbar to be more minimalistic and collapsible for mobile devices. Once the front-end website was implemented, the rest of the work consisted of game integration, the process of creating pages for each game, and integrating the game scripts into JSX components.

5.1.2 Hosting

Once the application was usable, the next stage of website implementation was hosting. To host the website, an application was created on Vercel and subsequently linked to the GitHub repository with our React application. Since the React application was directly linked, the web application was able to be modified so that every commit to GitHub would automatically create a new build, meaning that the website would be kept up to date consistently. Once the root folder was set to the React project containing the website, nothing had to be modified after connecting the GitHub to Vercel.

5.2 Game Implementations

5.2.1 Catch Game

The Catch game was implemented with JavaScript, HTML, and CSS. The use of JavaScript follows a functional programming paradigm and was used to create functions to construct a fluid, working game. There are JavaScript functions to move a paddle left and right, have blocks fall down the canvas and register a catch, get data for visualization, and reset the game. The HTML and CSS portions are for formatting and styling the game. An example of the start screen for the Catch game is shown in Figure 2.

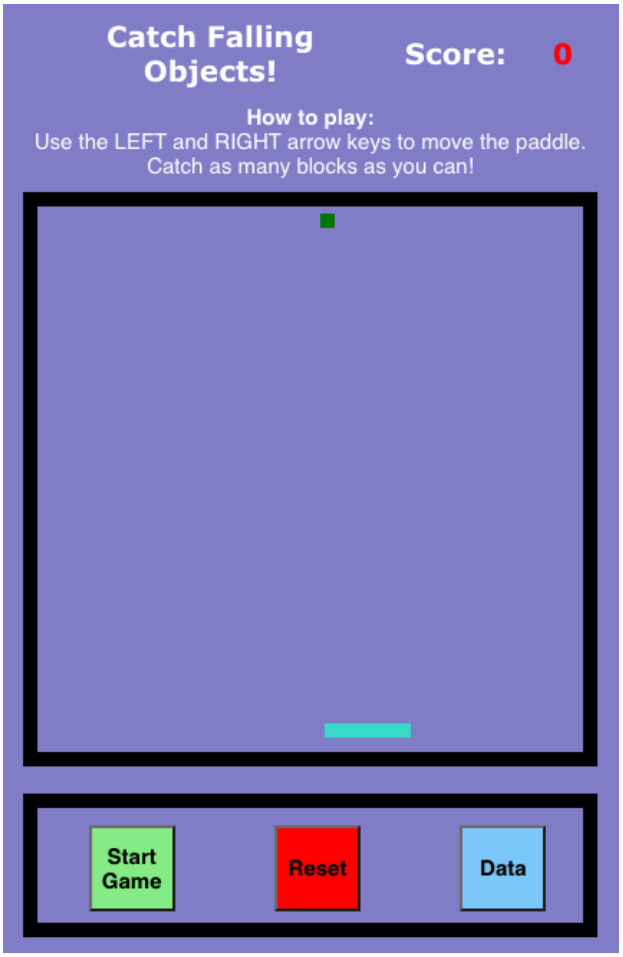

Figure 2: Catch Game Load Screen

The bottom section of the game page contains buttons that are selected with the click of a mouse. These buttons are used to start the game, reset the game, and grab the data visualization component after the user has lost. Once a game has been started, the user must control the paddle with the left and right arrow keys to catch the blocks. As the user catches blocks, their total score is updated in the top section of the game. The Catch game end screen is shown in Figure 3.

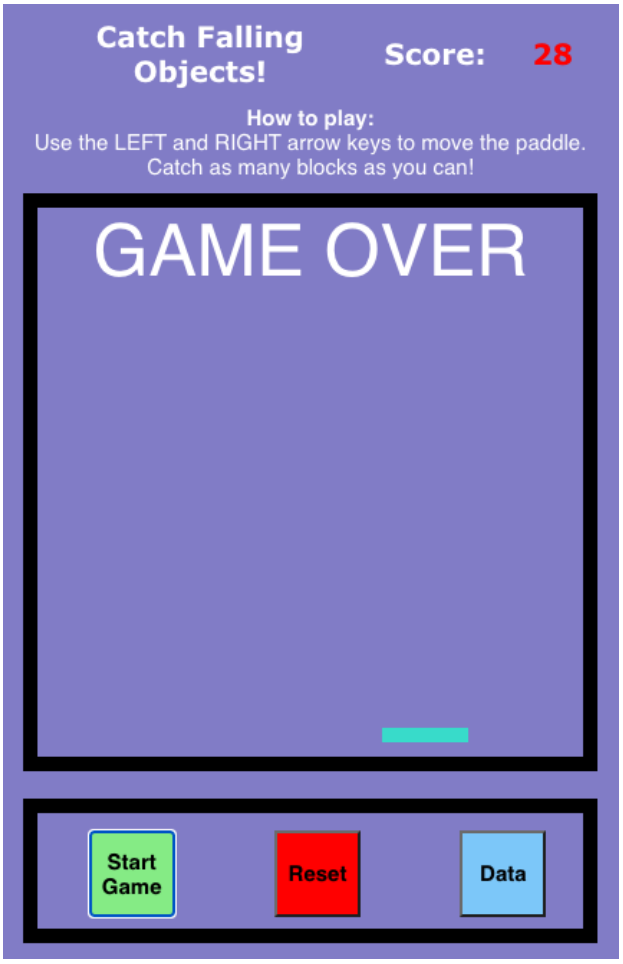

Figure 3: Catch Game End Screen

Once the user has missed a block, the game shows a “GAME OVER” screen. The user can now click on the “Data” button and visualize the number of block colors that they caught. After analyzing their data, the user may start a new game with the appropriate button selection. The data visualization screen of the Catch game is shown in Figure 4.

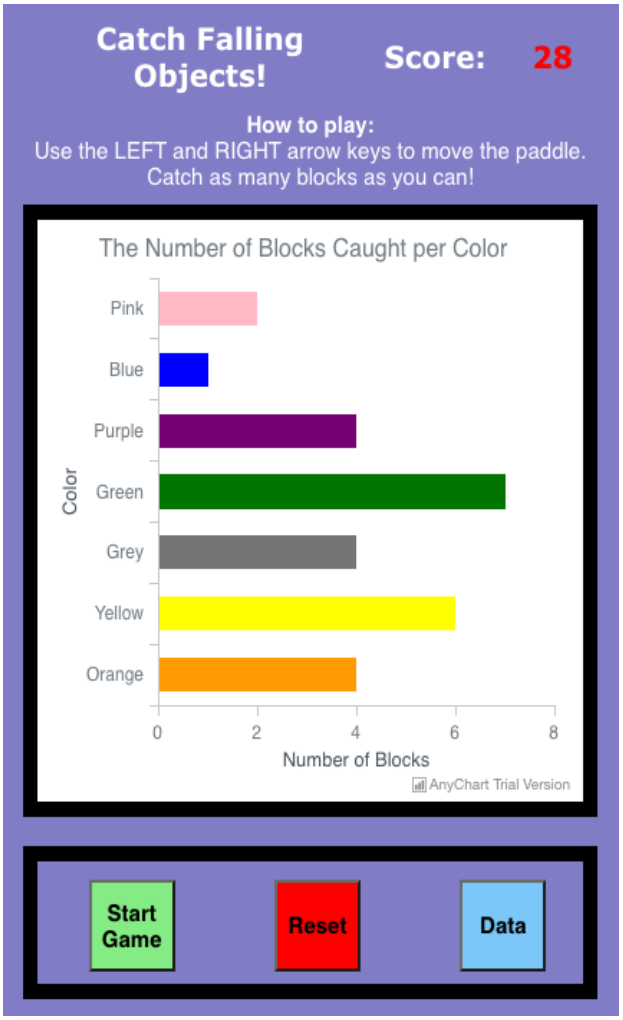

Figure 4: Catch Game Data Visualization Screen

5.2.2 Cake Game

The game was implemented in various chunks that do not reflect the order in which the content of the game is displayed to the player; the explanation of the implementation for this game will start from the player's perspective.

Even though these games are quite simple, the rules need to be explained well; as the target for this game is first and second graders, we need to make sure there is not too much text or they are likely to skip over it. This was a concern expressed by our client, Dr. Hamouda. So, the start screen in Figure 5 keeps short, simple sentences. The instructions go up to the end of the actual game part and do not include the questions at the end. When the player has finished reading the instructions, they will click "PLAY." Hovering the mouse over the button will make it brighter and pressing it will go to the next game scene.

Figure 5: Cake Game Start Screen

This next scene is the game itself as shown in Figure 6. The basic instructions are kept at the top of the screen in case the player is not sure what to do next; it will alternate between "Choose a flavor!" and "Place the cake!" The flavor table on the right is the only thing interactable at first and will create a black border around the piece they are hovering over; this is to get the player to appropriately click on one of the flavors. After the piece is clicked, the flavor maintains a border so the player knows which flavor they have selected. The tray is also now interactable; hovering over pieces of the tray makes the piece a lighter color and clicking on the piece will make it the color of the flavor chosen. This will continue to happen until all pieces are placed on the tray.

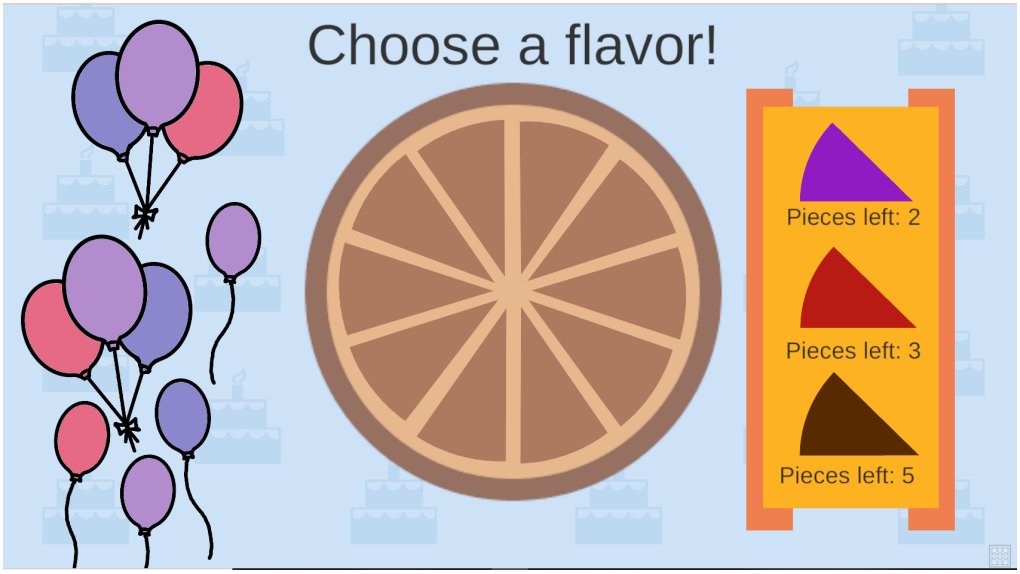

Figure 6: Cake Game Game Screen

At this point an animation will play with a pair of oven mittens taking the cake to be baked; this was added to provide a better sense of purpose to the game and in keeping with baking a cake. A new game scene will be loaded with a section of questions for the player, as shown in Figure 7. As this game is for 1st- and 2nd-graders, we do not use percentages but simple fractions instead. The pie chart should be similar to what the player made during the game.

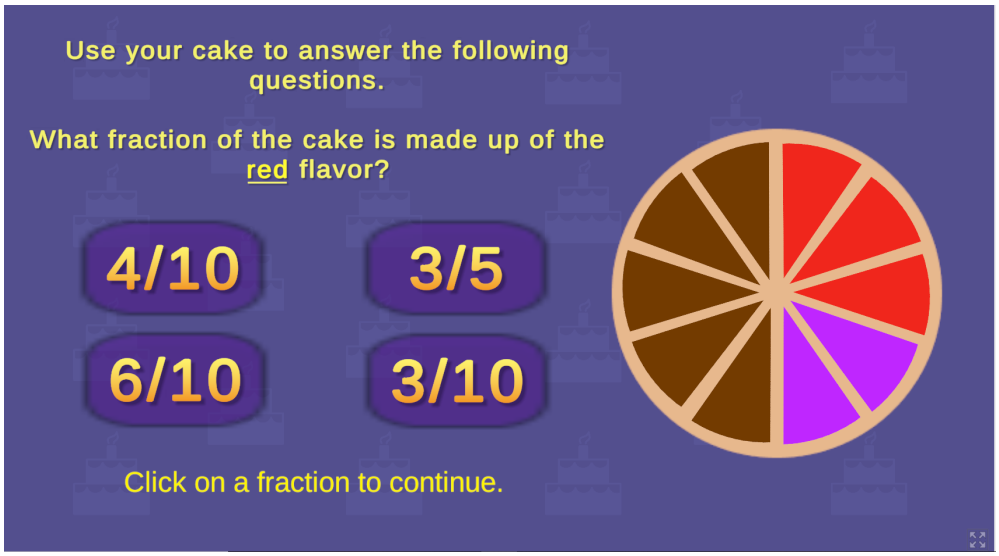

Figure 7: Cake Game Questions Screen

5.2.3 Tower Game

The tower game was implemented in vanilla JavaScript, HTML, and CSS and follows a functional programming paradigm. There are functions to start the game, restart the game, process a game over, create a new block, create a graph, and create questions. The game itself is within an HTML Canvas element. Overall, the game was implemented using DOM manipulation and event listeners.

Figure 8: Tower Game Start Screen

In Figure 8, the initial start screen for the game is shown. It is self-explanatory and the player simply needs to click the screen to start the game.

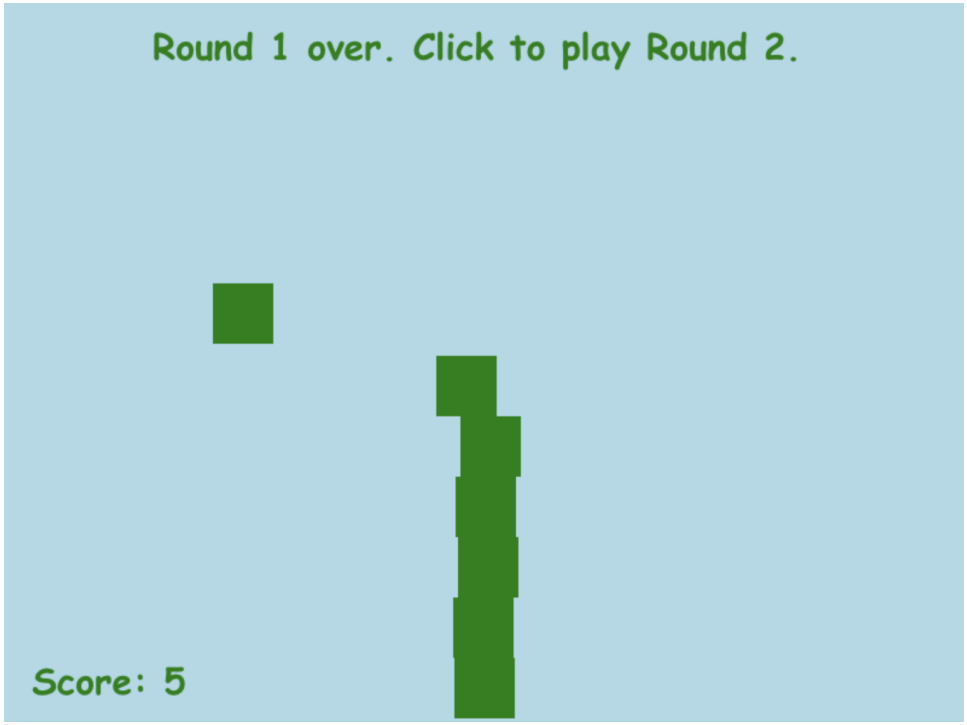

Figure 9: Tower Game End of Round 1

In Figure 9, the end of round 1 screen is shown because the player has missed the last block. A score is shown on the bottom left.

Figure 10: Tower Game Questions Screen

In Figure 10, a bar graph has been generated that has questions for the student to answer about the graph.

5.2.4 Math Game

The math game is a self-explanatory game where the player inputs a number in the blank provided to make the equation to be true. The equations for this game consist of only addition and subtraction problems.

The game itself was implemented using JavaScript, HTML, and CSS. It also follows a functional programming paradigm. There are functions to randomize the number of questions, randomize the input blank, and randomize the questions. The game itself is handled using event listeners and buttons with specific styles from HTML and CSS.

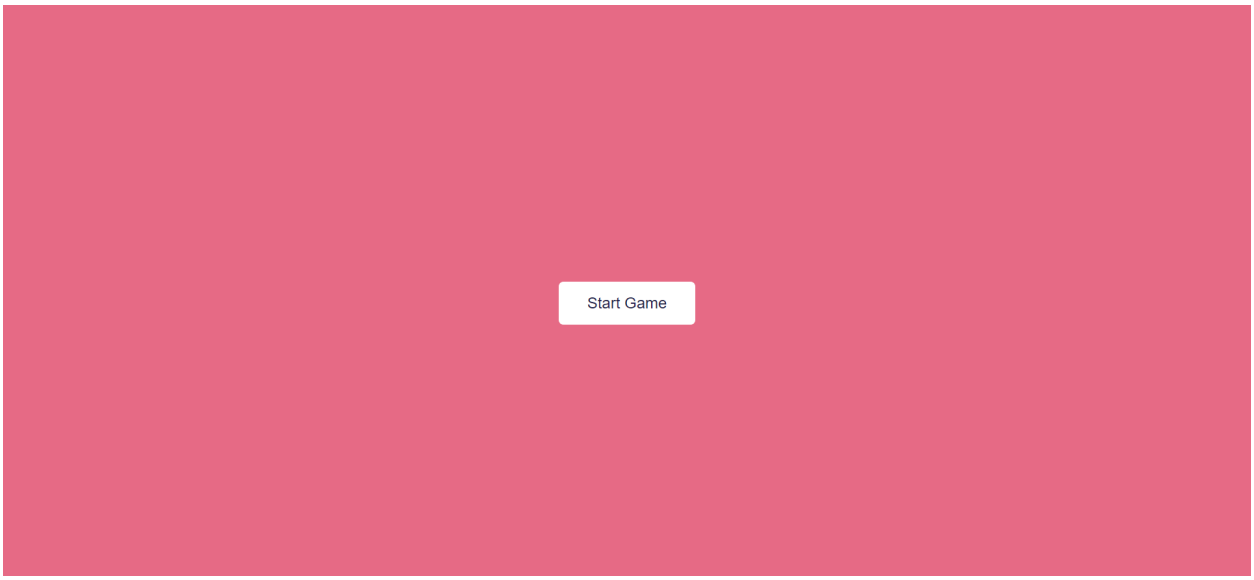

Figure 11: Math Game Start Screen

In Figure 11, the screen lays out the start screen for the game with a simple button with the text “Start Game.” This screen is self-explanatory and the student can start the game by pressing the button.

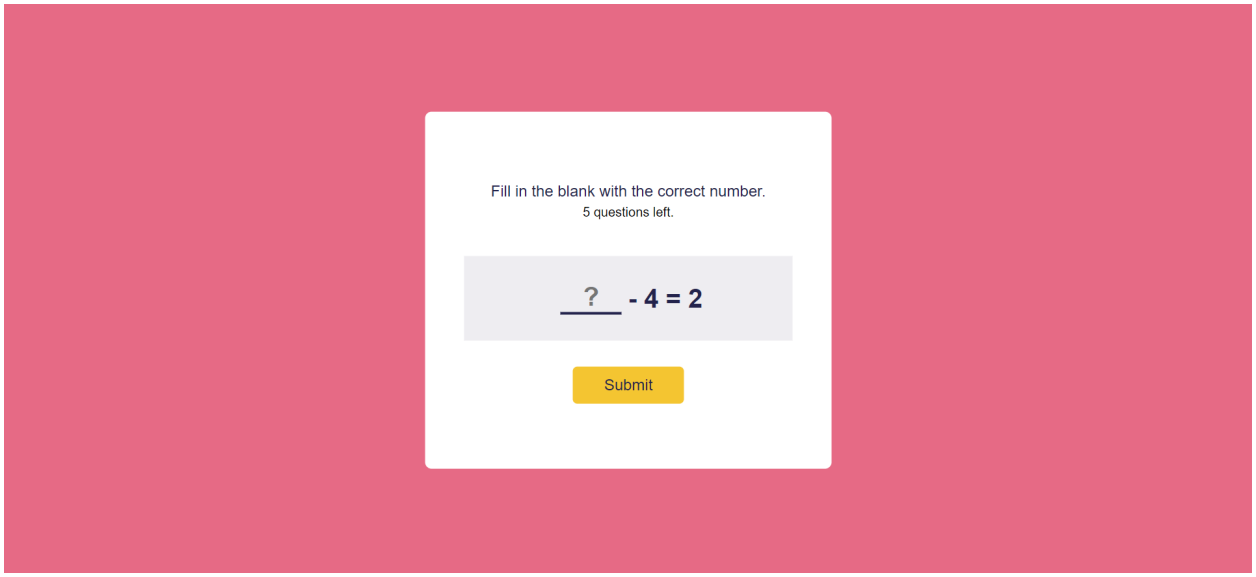

Figure 12: Math Game Question Screen

In Figure 12, the screen lays out the question screen for the game. This screen is self-explanatory and it displays to the students that they should fill in the blank, how many questions are left, the question itself, and a submit button.

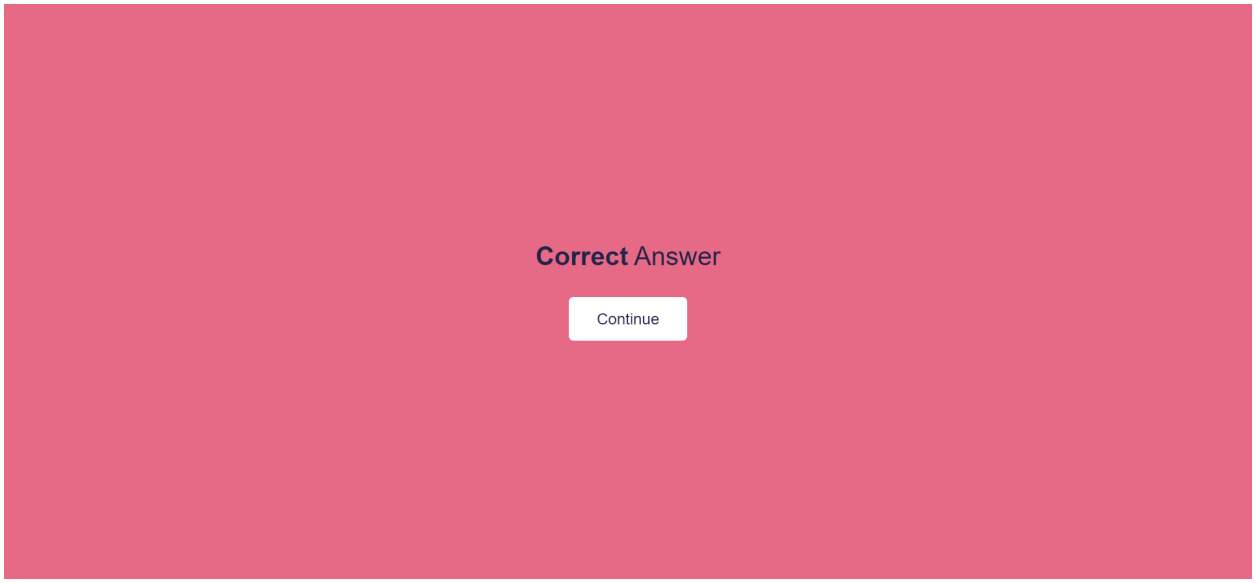

Figure 13: Math Game Correct Screen

In Figure 13, the screen lays out the correct screen for when the student answers a question correctly. It displays “correct answer” and includes a button to continue the game until the last question.

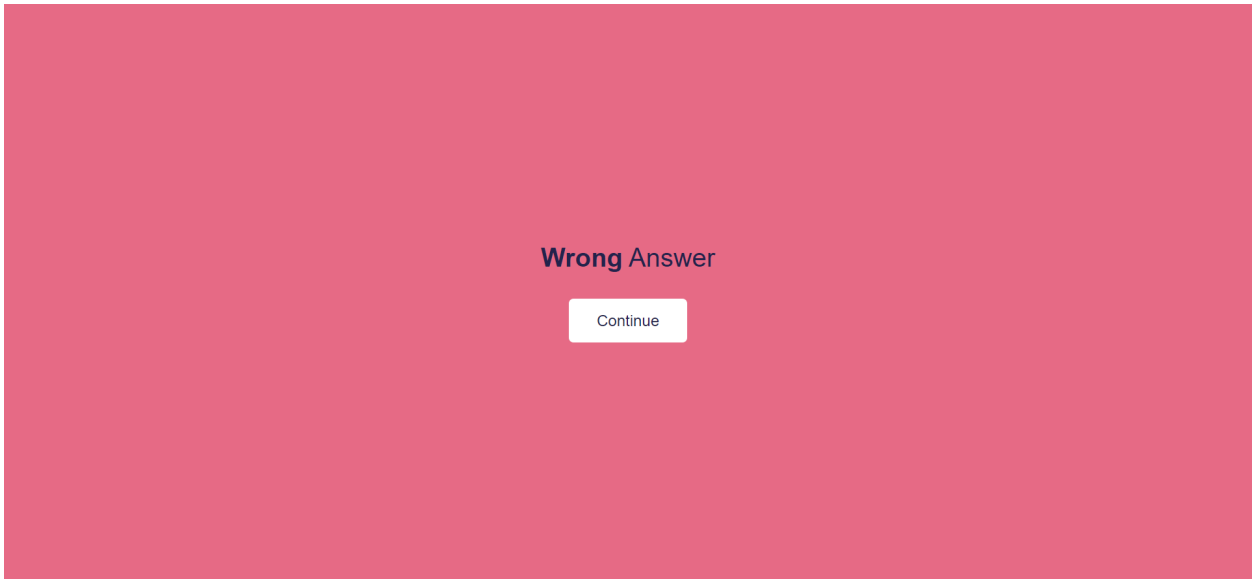

Figure 14: Math Game Incorrect Screen

In Figure 14, the screen lays out the incorrect screen for when the student answers a question incorrectly. It displays the text “incorrect answer” and includes a button to continue the game until the last question.

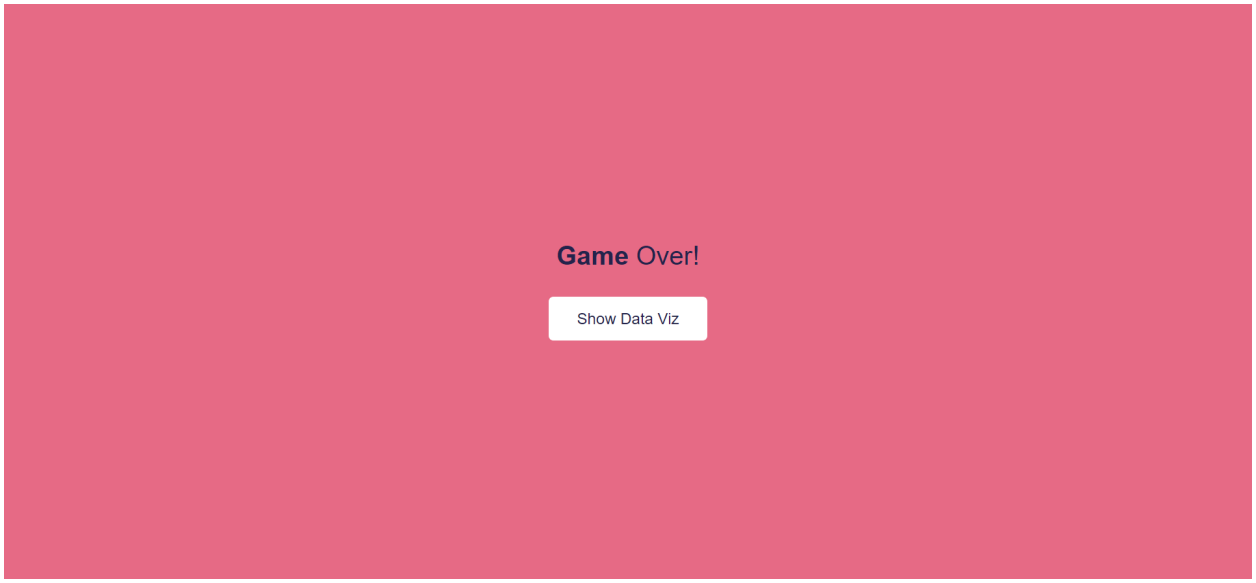

Figure 15: Math Game Over Screen

In Figure 15, the screen lays out the “Game Over” screen for when the game has concluded. This screen only appears after each of the questions has been answered. It displays “Game Over!” and includes a button to show the data visualization part.

Figure 16: Math Game Data Visualization Screen

Figure 16 shows the data visualization screen for when the game has concluded. This screen only appears after each of the questions has been answered and the show data viz button has been clicked. It displays a bar chart that scores the number of correct and incorrect answers. In this example, we can see that out of 10 questions, the student answered 5 questions correctly and 5 questions incorrectly.

5.2.5 Ocean Game

The ocean game is essentially not a game but more of an art tool with a data visualization providing information about what was created by the user.

The ocean game was made purely in React. There are two modes which are changed by the button in the bottom left corner. The two modes are the create mode and data mode. The mode that is loaded in when the game is selected is the create mode, shown in Figure 17.

Figure 17: Ocean Game Initial Create Mode

The create mode consists of 429 25x25 pixel images of empty pictures. When clicked on, the image changes to blue fish, then red fish, then back to an empty picture. The bottom row has coral and kelp instead of blue and red fish, respectively. The game counts every time the picture is switched between a different object for the data visualization. A fully designed game in create mode is shown in Figure 18.

Figure 18: Ocean Game Create Mode with a Design

Upon pressing the button in the bottom left, the data visualization component appears, shown in Figure 19. The creation is then deleted and a grid of images appear on screen showing all the objects that were in the drawing together and sorted by image. The counter is then displayed too, so the user can see exactly how many objects they added. When the button in the bottom left is pressed again during data mode, the drawing is reset and the game looks the same as it did in the

initial creative mode. The counters are also reset to 0.

Figure 19: Ocean Game Data Visualization

5.2.6 Lego Game

The Lego Game is set up very similarly to the Cake Game; first we start with the screen that details what the rules are for the game. The instructions go up to the end of the first part of the game which the developer refers to as the "sorting part." It does not include the questions at the end. The instructions can be seen in Figure 20. When the player has finished reading the instructions, they will click "PLAY." Hovering the mouse over the button will make the button background dark and pressing it will go to the next game scene.

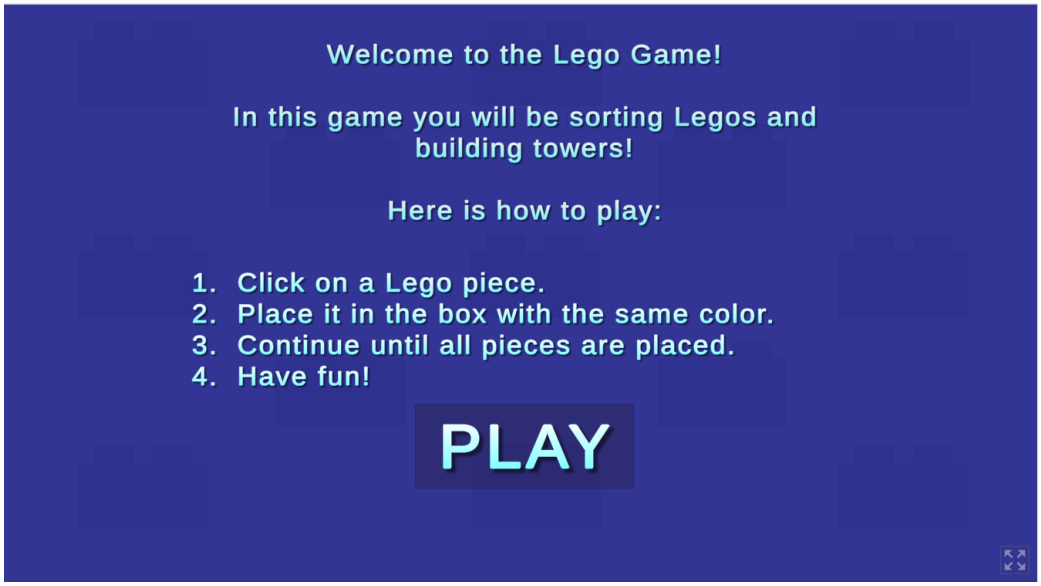

Figure 20: Lego Game Start Screen

This next scene is the game itself as shown in Figure 21. The basic instructions are kept at the top of the screen in case the player is not sure what to do next; it will alternate between "Choose a Lego piece." and "Place the piece with the correct color." The squares represent bins; these are

where the player will place their pieces which will momentarily disappear after placing them. The player will continue to place the unsorted Legos in bins until there are no more, at which point the next game scene will load.

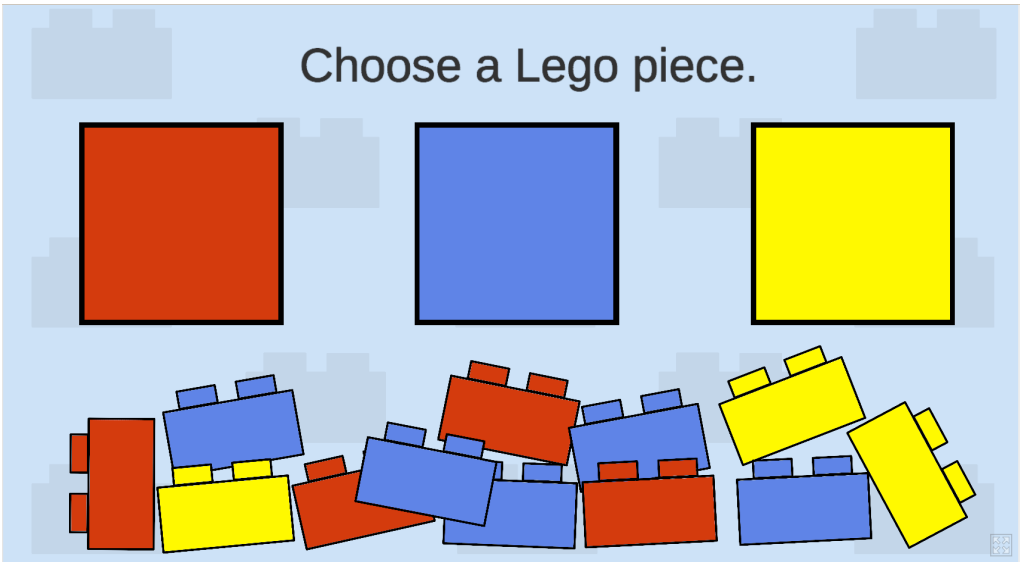

Figure 21: Lego Game Game Scene

The player will then see the screen in Figure 22. Here they are tasked with stacking the same colored bricks on top of each other. The translucent piece is there to show where the player should place each brick; as they place one, the translucent piece will move up the equivalent of one brick and its original position will be filled with the correct-colored brick. The translucent piece will disappear completely when the player has stacked all pieces of that color.

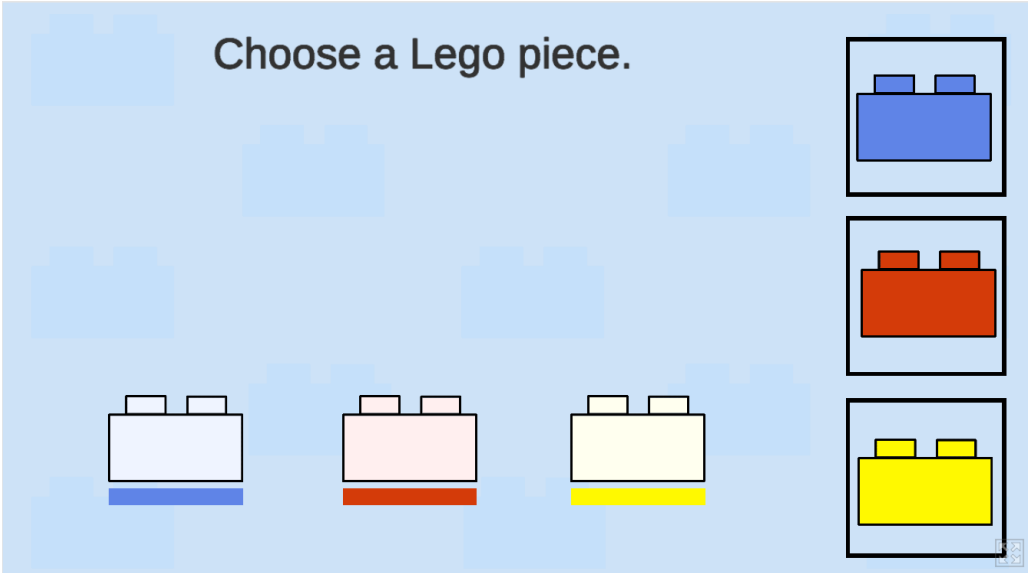

Figure 22: Lego Game Game Scene 2

Next, the player will get two scenes with different questions on basic fractions; one such screen can be seen in Figure 23. At this point, they will hopefully be well acquainted with the data set and can easily answer these questions and refer to the bar chart as necessary. The bar chart will be the same as the one the player made during the game.

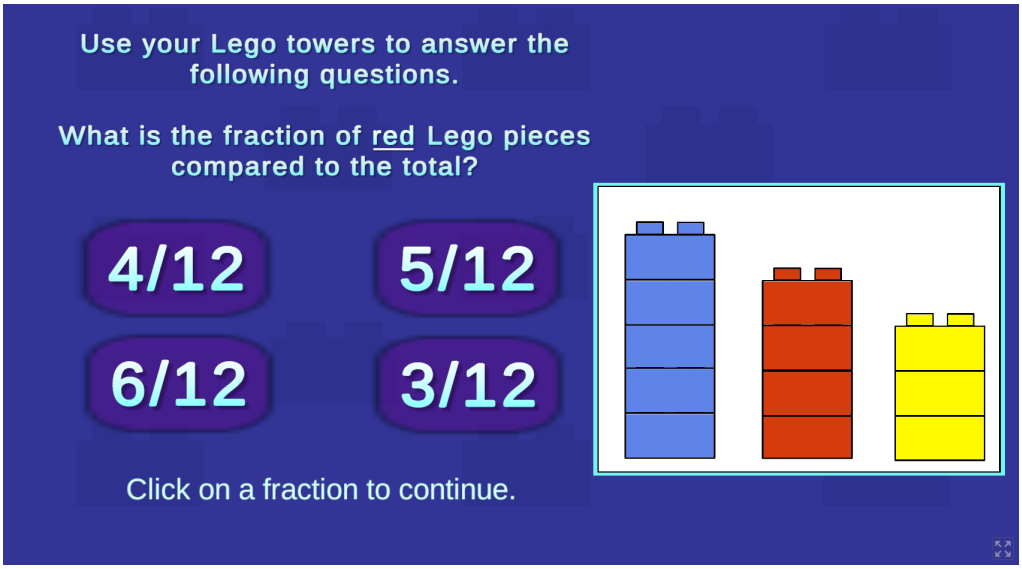

Figure 23: Lego Game Questions Screen

6.0 Testing

Once a presentable and complete version of the website was finished, with 6 of our games fully integrated and debugged, we began our testing process. To allow a broader range of testers to interact with our website we created an easy-to-share anonymous digital form with a link to our website followed by a prompt for feedback.

6.1 Form Implementation

The form starts with collecting demographic information from the tester, shown in Figure 24. Since age range is important to this project, we collect data on both the age of the tester and the grade of the tester (if they are in elementary school). Since the website will also be tested for general accessibility, some of the testers may be older than the elementary school grade range.

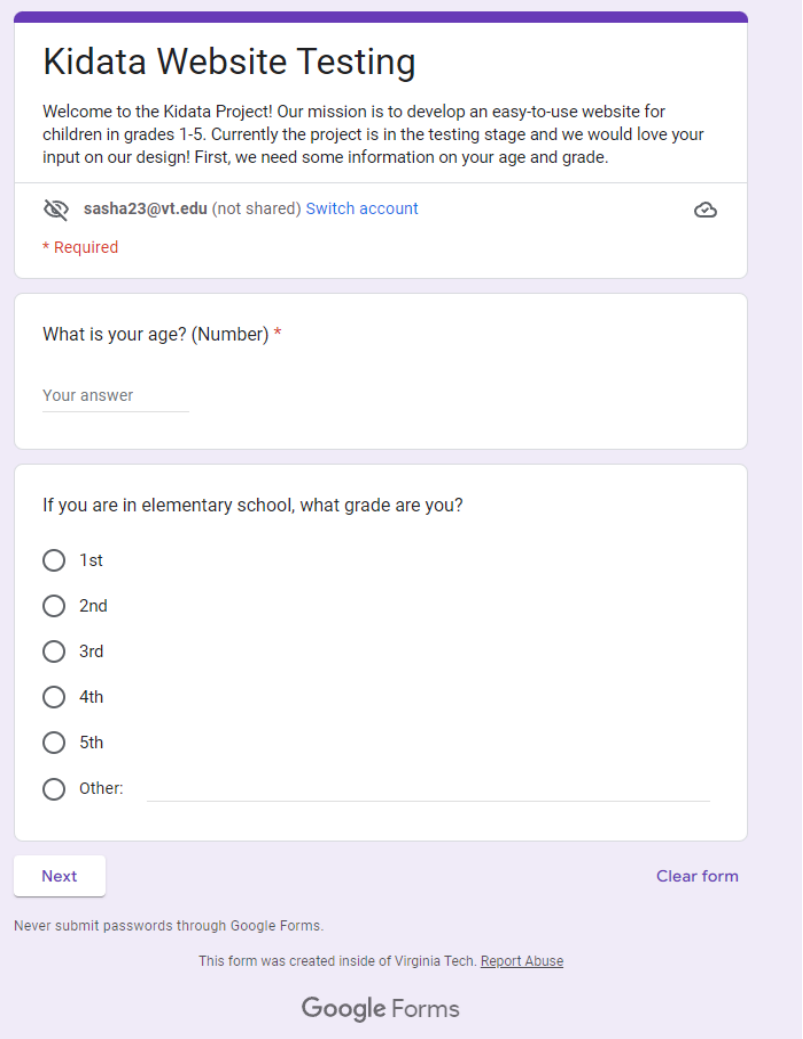

Figure 24: Section 1 of Testing Form

After the demographic information is taken, some questions are asked regarding the user's ability to open the website if the website cannot be accessed; then the form finishes. Selections for this question are seen in Figure 25.

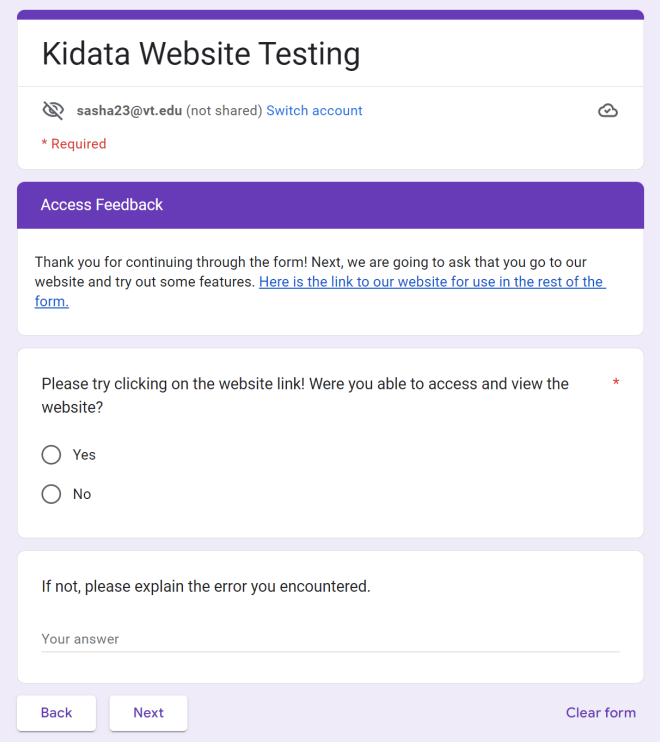

Figure 25: Section 2 of Testing Form

If the user can access the website, a substantial part of the testing begins, in which the user is prompted to do various tasks, such as finding a page or playing a game. Each prompt is followed by a section for the user to record their experience, thoughts, and advice. These questions can be seen in Figure 26.

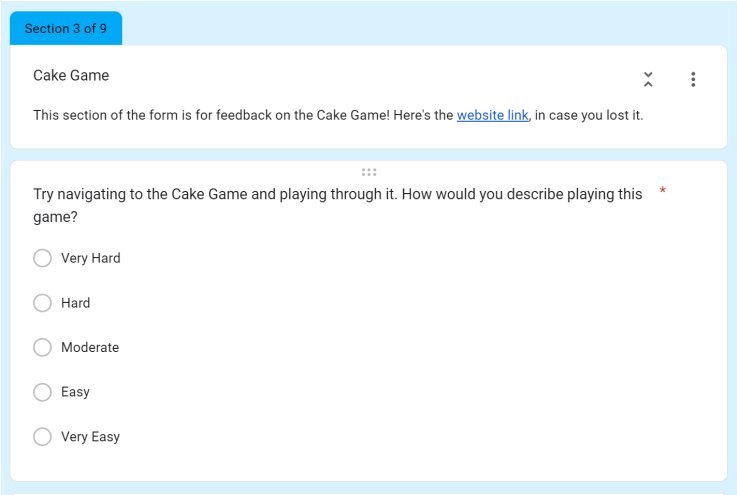

Figure 26: Sample Question from Section 3 of Testing Form

6.2 Changes

A week after the form was sent out, the submissions were compiled and evaluated. Any smaller changes like formatting, layout, and communication were implemented as soon as possible, while larger systematic changes are included in future work documentation.

7.0 User's Manual

7.1 Website

The KidDataViz website (https://kid-data-viz-299f.vercel.app/) can easily be accessed from the URL assigned to the application by Vercel. To navigate, follow the toolbar at the top of the page, or on mobile, use the pop-up menu in the upper left corner. There is a Home, Play, About, and Help page in the application.

7.2 Games

The games are accessed on the website on the "Play" page of the website, where you will find the categories by elementary school level (1st- and 2nd-grade, 3rd- and 4th-grade, and 5th-grade and older). Each game has been given one of these categories in the Design and Implementation sections, and can be found in their respective category.

An explanation of how the game works and why it was implemented is given in the Implementation section; this section will go over the basic rules/flow of the game.

Note: If you need or would like a visual aid, it is recommended that you look at the Implementation section's figures.

7.2.1 Catch Game

Once the user has navigated to the game via the website, the user may click "Start Game" and a block will begin to fall. The user must control the paddle with the left and right arrow keys to catch the falling block. The user will only need to focus on catching one block at a time; however, the following block will begin movement immediately after catching the previous block. Lastly, the user must catch as many consecutive blocks as they can. If a block is missed, the game will end.

Once the game has ended, the user may click on the "Data" button to see a visual analysis of the color of the blocks that they caught. After understanding the data visualization component, the user may choose to start over and begin a new game.

7.2.2 Cake Game

Once you have navigated to the game via the website, simply read the rules and then click the "PLAY" button to begin. A new screen will appear with a baking tray in the middle with ten slots for placing the cake flavors. On the right will be the flavor table where you will be able to select one of three flavors of cake.

To start, click on a flavor and begin placing the pieces onto the baking tray. You should try to place all similar pieces next to each other. Once you have finished one flavor, click a new one and place all of those; do this for the last flavor as well.

The baking tray will be taken from you and a new screen will appear with simple fraction questions. Read and answer these questions to complete the game. A screen will display your results once all of the questions have been answered. Reload the page if you would like to play it again.

7.2.3 Tower Game

Once you have navigated to the game via the website, read the basic instructions and click the screen to begin. The first round will begin immediately with a starting block at the bottom, a new block at the top, and a score at the bottom left. The new block will continue to go right and left; click the screen to drop the block. If the new block lands on the previous block, the game will continue. Otherwise, the game will end. Continue to stack the tower as high as possible.

There are three rounds total, each with increasing difficulty. When a round is over, there is an intermediate screen to click to start the next round. Once all the rounds are over, a bar graph is generated from the scores.

7.2.4 Math Game

Once you have navigated to the game via the website and clicked the start button, the game will begin. The question screen will first appear and the user can play by inputting an answer into the given slot. The question screen for each consecutive question will remain the same with only the number of questions and the question itself changing. Each question will require the user to click the submit button when they are confident that their answer is correct. Once submitted, a screen will appear depicting whether or not the question has been answered correctly or incorrectly. From there, the user can continue playing the game by clicking the continue button. Once each question has been answered, a screen will appear stating that the game is over. From there, the user can display the dataset they have created from playing the game by clicking the show data viz button. The bar chart data visualization will then appear on the screen. The user can restart the game by refreshing the page and the start screen will appear again.

7.2.5 Ocean Game

Once the user has navigated to the game via the website, they click anywhere within the sea blue box to create a fish, coral, or kelp. Upon completion of their design, they can click on the blue button in the bottom left corner of the page. This will erase the user's design and give them data about what they made. They can then press the same button to reset the game and make a new design.

7.2.6 Lego Game

Once you have navigated to the game via the website, simply read the rules and then click the “PLAY” button to begin. A new screen will appear with twelve unsorted Legos on the bottom of the screen and three different colored bins on the top.

To start, click on a Lego piece and place it into the bin with the same color. Continue to do this until all of the Legos have been sorted into the bins. Once finished, a new screen will appear with Lego pieces on the right and three translucent pieces on the bottom. Stack each Lego piece onto its corresponding color; you will specifically place each piece onto a translucent piece of the correct color.

A new screen will load with basic fraction questions. Read and answer these questions to complete the game. A screen will display your results once all of the questions have been answered. Reload the page if you would like to play it again.

8.0 Developer's Manual

8.1 Website

Before working on the website, the developer should have access to both the GitHub for the React application and the Vercel application, both of which were given to Dr. Hamouda at the end of the project.

The first step to continue developing the website is to locally clone the GitHub repository that is currently linked to the Vercel application. This will give access to the code that is currently running on the server, and reading through this code will help any new developer understand the code structure of this project.

Once the repository is locally cloned, Node.js and npm must be installed on the local machine for local development. For more information on this, follow the npm docs at https://docs.npmjs.com/downloading-and-installing-node-js-and-npm. It is recommended you use a Node version manager like nvm (https://github.com/nvm-sh/nvm) to install Node.js and npm. Once Node.js is installed, run "npm install" in the React application directory to ensure that all dependencies are installed, including the various Material UI (MUI) libraries that are used.

It is recommended that you use a web development IDE like WebStorm to then modify the application code. To host the application locally and see any progress made on the website before re-deploying the website, simply run the command "npm start" in the React application directory and follow the link that is shown on the console after it compiles to view the application in any browser.

Once the modifications are ready for publishing, commit your changes to the main branch of the GitHub repository and check Vercel for the build.

8.2 Games

The games were created with either JavaScript on the website or as Unity games embedded into the website.

To work on the games implemented with JavaScript, start by following the instructions on how to work on the website in Section 8.1. The development of all our JavaScript games involves the use of the AnyChart JavaScript library. Implementation of the data visualization component can be found on the AnyChart bar chart tutorial at https://www.anychart.com/blog/2017/10/25/javascript-bar-chart-tutorial/. To edit the functionality of the game, work in the game's respective .js files in the public directory. To edit the game formatting and styling, work in the game's respective .css and .jsx files within the game's respective directory in the pages directory. The HTML component of the games was converted from an HTML file to a JavaScript JSX file so that the game can be implemented directly onto the React website. The HTML implementation needs to be in JSX form.

To work on games with Unity, download Unity from https://unity3d.com/get-unity/download. However, if you specifically want to work on the version of Unity that the student did for the Cake Game, go to https://unity3d.com/get-unity/download/archive and select version 2020.3.25f1.

8.2.1 Catch Game

The development of the catch game uses vanilla JavaScript along with HTML and CSS, and uses the AnyChart JavaScript library for the data visualization components of the game. There are functions to control the paddle using the left and right arrow keys, as well as a swiping functionality for mobile use. There is a function that starts the game when the "Start Game" button is clicked called "movedown". This function calls another function called "frame" that is responsible for updating each frame of the game where a colored block is falling down a path and a paddle is being controlled to catch the block. Once the game is started, a randomized colored brick at the top of the play area (.path) is generated and begins to start falling towards the bottom of the .path box. The paddle (.brick) is controlled with the left and right arrow keys. If the position of the falling block is within the horizontal range of the paddle's width while also at the vertical position of the paddle, the block will be considered "caught" and the score will be updated through the updateScore() function. The counts of each color block are kept by a dictionary where the key is the color of the block and the value is the frequency of the block. If the block does not meet the conditions to be considered "caught", then the "GAME OVER" screen will appear. There are two additional buttons that reset the game and show the data of the last game played. The reset() function clears the values of the dictionary and resets the score counter. The getData() function utilizes the AnyChart Javascript library to generate and display the bar chart.

8.2.2 Cake Game

The following section will go over the major development parts of the Cake Game including game scenes, game objects, art assets, and scripts. It will talk about these things as if the reader has access to the Unity project itself; it will detail where these things are and what should be worked on when editing the game with figures and text.

A scene in Unity is a space setup where there are multiple game objects; this helps with the flow of control as you can go from one scene to the next, e.g., the start screen to the game scene. Figure 27 has the four scenes we used for the game. "GameScene" has the actual game where the player selects flavors and puts them onto a cake. "LoadingScene" is used in between scenes, "Questions" is used for the sections with multiple-choice questions at the end of the game, and "StartMenu" is the menu the player sees at the start of the game. Were this to be edited, most of the work would be in "GameScene" as that is where most game objects and scripts are used.

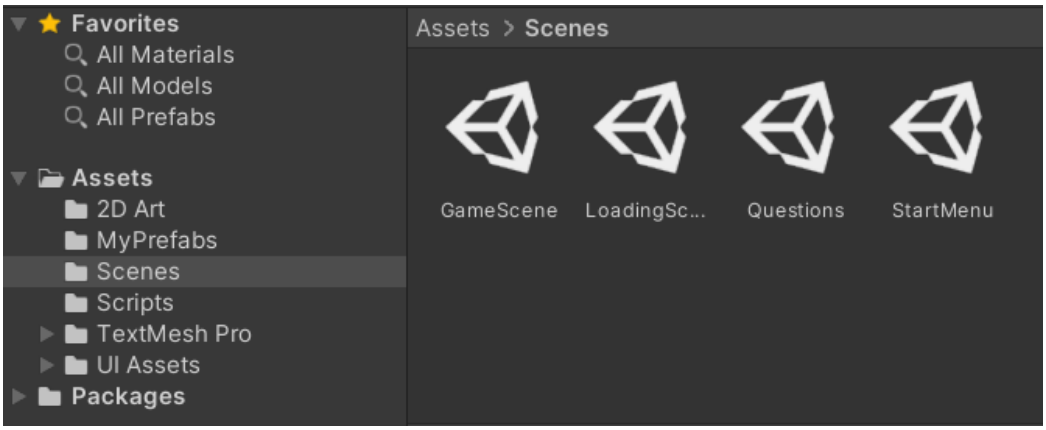

Figure 27: Cake Game Scenes

Within the GameScene itself, the following are the game objects that the developer should familiarize themselves with.

Basic art assets: These can all be found in the "2D Art" folder. These are used for the sprites in-game objects.

Flavors (an empty game object) has types called Mint, Red Velvet, and Chocolate. These are the clickable flavors you see on the right in the flavor table. Each of these individual objects utilizes the "FlavorScript" C# script which handles their logic of how many pieces they have and will change to a grayish color once zero. You can see these in Figure 28.

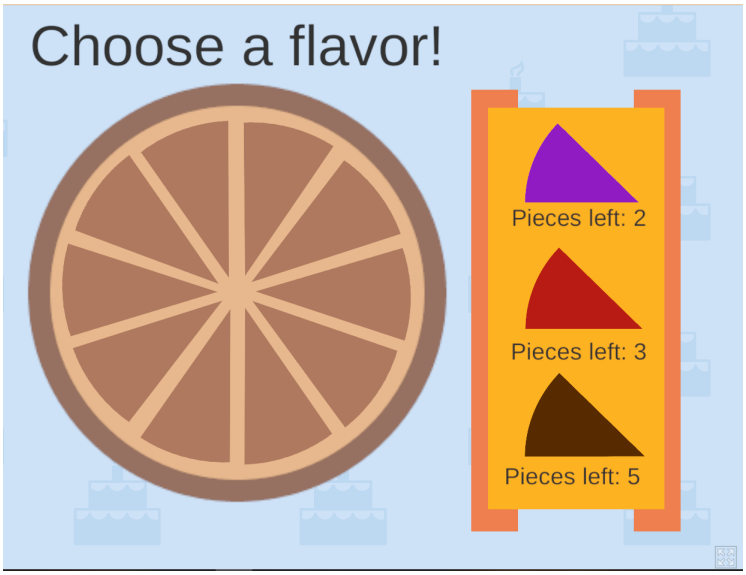

Figure 28: Cake Game Game Scene

The next important game objects are called "**Cake Slice [#]**" where # is a number zero through ten. These are the gray slices you see on the baking tray in Figure 28. These objects all use the "ShapeBehavior" script which changes the color of the piece color when hovered over and the pieces change to the correct color flavor when clicked upon.

These were the most important aspects as they handle most of the logic of the game. Most other stuff seen are simple art assets and basic UI objects like buttons that Unity has made very simple to edit and use.

8.2.3 Tower Game

The development of the tower game uses the AnyChart JavaScript library for the data visualization components of the game. All global variables are declared in the beginning, including two DOM elements named canvas and instructions. The game is contained in the canvas element and the text instruction is the instruction element. The game follows a functional programming paradigm and is started with the "start" function which creates the initial start screen. The user clicks are handled by the "canvas.onpointerdown" function and there are three possible actions. If the "mode" is gameover and the "round" is not 4, the "restart" function is called and variables are initialized for the respective round. Else if the "mode" is gameover and the "round" is 4, the "graph" function is called and the bar graph is generated. Else the game is ongoing and the block drops. In the final case when the game is ongoing, the animation for the block drop is handled by the "animate" function. To change the block horizontal speed, change "xspeed" in the "restart" function. To change how the block speeds up, change how "xspeed" is changed in the "animate" function. To change the block size, change the global "width" and "height".

8.2.4 Math Game

The development of the math game uses the AnyChart JavaScript library for the data visualization components of the game. All variables associated with HTML and CSS components are declared at the beginning. There are two variables named correct and incorrect that get incremented each time the user answers a question either correctly or incorrectly. These two data points are visualized using a bar chart at the end of the game. The game itself is implemented using JavaScript, HTML, and CSS and follows a functional programming paradigm. There are functions to randomize the number of questions (numQuestions), randomize the input blank, and randomize the questions. These random aspects of the game are handled using a random value generator that makes use of the Math object in JavaScript. The game itself is handled using event listeners and buttons with specific styles from HTML and CSS. There is a questionGenerator function that generates a random question and keeps track of the operator and two numbers. Within this function, the solution of the problem is evaluated using the JavaScript eval function. The input blank is then randomly generated and the user input is matched with the answer value from the evaluation when the submit button is clicked. If the user input is empty, there is an error check to ensure that the input cannot be empty. There is also a function that continues the game (continueGame) and generates a new question. When this function is called, the numQuestions variable gets decremented. When numQuestions reaches 0, the stopGame function is called. This function ends the game and transitions to the data visualization portion where the bar chart is displayed.

8.2.5 Ocean Game

The development of this game uses the MUI react component library. The code for this game is on the file “OceanGame.jsx” with some extra styling on the file “OceanGame.css.” This guide assumes that the user is able to find the MUI react component library on the web and has access to the documentation. The create portion of this game is just 390 empty images that produce fish when clicked on (row1) and 39 that produce kelp or coral when clicked on (row2). There are five variables that use React state hooks. The Boolean variable tells the game which mode it is in (create or data mode). The number variables keep count of how many of each object have been added to the game. The two handleImageClick methods are for when an image is clicked in. handleImageClick is called when images in row1 are clicked on and handleImageClick1 is called when images in row2 are clicked on. The button text variable determines the text on the button in the bottom left hand corner. The four variables, i.e., addedRedFish, addedBlueFish, addedKelp, and addedCoral, are the four variables that make up the pictures in the data visualization. The Chart and Game variables are the React code for data mode and game mode, respectively. The inUse variable determines which mode to display and displays it on the page. The grids within the return statement of the ocean game method are purely for styling the game and ensuring the game looks proper. In order to ensure that the game is playable, the box which the create portion of the game is in, is set to a specific width to ensure the entire box is clickable.

8.2.6 Lego Game

This section assumes that the reader has either already read through the previous section, 8.2.2, as it covers basic Unity concepts, or that the reader is familiar with Unity.

As you can see in Figure 29, the Lego Game makes use of six games for the flow of the game. “GameScene” has a part of the game where the player sorts Legos into bins. “TowerCreation” is where the player stacks Legos to create the bar charts. “StartMenu” is the first screen a player sees; it has the instructions for the game. “Questions” and “Question 2” have the basic questions about fractions for the player. “EndScene” is as the name describes; it will have a number displayed to the player showing how many times they incorrectly answered the multiple choice questions. When editing the games, most work would be relegated to “GameScene” and “TowerCreation,” depending on which part of the game a future developer would need to edit.

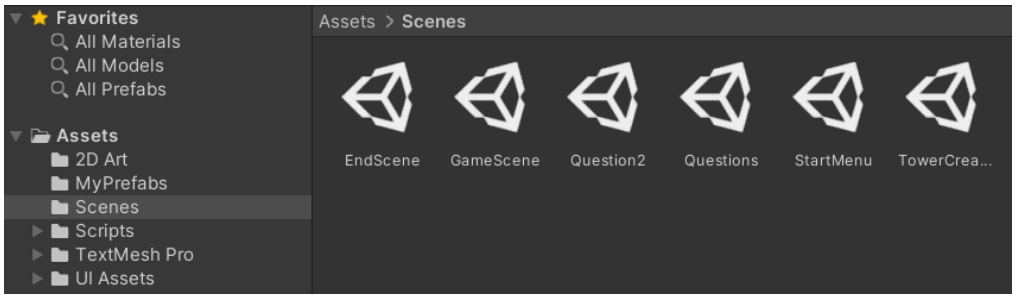

Figure 29: Lego Game Scenes

Some key scripts handle the logic of the game. These scripts are attached to multiple scenes and multiple objects. The first of these is called “LegoPiece” which handles being clicked on with asynchronous calls like “OnMouseOver()” and “OnMouseExit().” This script handles a few edge cases like clicking on multiple of these objects at once and when the object should change color. The script “OrganizerTubs” will make the mouse drop the “LegoPiece” object it is holding, otherwise it will just change color when hovering over it. There are twelve objects in the scene “GameScene” that utilize the script “LegoPiece,” and three objects that utilize “OrganizerTubs.”

The next important script is “InvisLegos.” This script handles the logic of the translucent piece that will replace a “LegoPiece” object with its current position and move up the height of a Lego piece. It will also destroy itself once the player has placed all Lego pieces of the correct color. In the scene “TowerCreation,” there are again twelve pieces that utilize “LegoPiece” and three objects that utilize “InvisLegos.”

The above scripts handle most of the game logic. The rest of the scripts handle very simple things, e.g., move to the next scene on the press of a button or read in some data to display with some text. The rest of the components in the game are mostly assets and GUI elements.

9.0 Lessons Learned

Throughout the project, a consistent hardship was managing our timeline and schedule. We tended to have weeks where we got a lot done, followed by weeks where no real progress was made. This was primarily due to the poor delegation of tasks. A single member would take on the work of doing the website while another took on the work of creating a test game, while the rest of the group was not sure what to work on. After a meeting with Dr. Hamouda where she pointed out this issue, we were able to delegate work more efficiently and speed up the pace at which we were able to develop. Unfortunately, this did put us behind schedule on integration since we were behind on game development. To compensate for this, we were able to have a couple of members focus on integration, leaving two of the games on the back burner. A couple of key lessons were learned from this: leave time in your schedule for mishaps, delegate work from the beginning, and value client input.

Another problem that the group encountered was designing around integration. Initially, we decided to develop the games in Unity, but during the initial attempt to integrate Unity games into the website using WebGL, we discovered that it would be much easier to integrate into React if the games were developed in JavaScript and CSS. Our solution was to switch the development of subsequent games to JavaScript immediately, which made our integration phase much less problematic. Ultimately, we learned that it is essential to have your integration method and logistics in mind when selecting development methods for different kinds of software

10.0 Future Work

10.1 Front-end Future Work

The Front-end of the project could use a couple of key updates, primarily more game implementations. Additional Front-end work could include a “lessons” page that will provide more formal teaching to students.

10.2 Back-end Future Work

Currently, our implementation does not include a back-end. In the future, a database can be connected to the website to allow users to signup/log in and save game progress. Our team planned to use MongoDB as the database. https://www.mongodb.com/basics can be helpful to get started.

A basic database schema would be as shown in Figure 30.

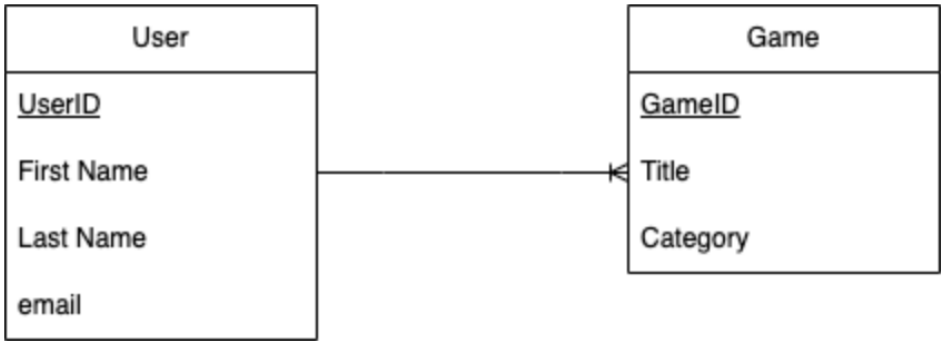

Figure 30: Future Database Schema

11.0 Acknowledgements

We would like to thank our client, Dr. Sally Hamouda, for working with us throughout the semester, and consistently giving thoughtful feedback that helped us make better decisions for the project. We would also like to acknowledge our instructor, Dr. Edward Fox, for providing us with guidance and advice.

Instructor: Dr. Edward Fox
fox@vt.edu

Client: Dr. Sally Hamouda
shamouda@vt.edu

12.0 References

[1] AnyChart, “Anychart is a lightweight and robust JavaScript charting library,” *AnyChart,* 2022. [Online]. Available: https://www.anychart.com/. [Accessed: 12-Dec-2022].

[2] Coolmath.com, “Cool math games - free online math games, Cool Puzzles, and more,” *Cool Math Games - Free Online Math Games, Cool Puzzles, and More,* 2022. [Online]. Available: https://www.coolmathgames.com/. [Accessed: 29-Nov-2022].

[3] Figma, “The Collaborative Interface Design Tool.,” *Figma*, 2022. [Online]. Available: https://www.figma.com/. [Accessed: 12-Dec-2022].

[4] MongoDB, “Beginners Guide: Mongodb Basics,” *MongoDB,* 2022. [Online]. Available: https://www.mongodb.com/basics. [Accessed: 29-Nov-2022].

[5] MUI, “The React Component Library You always wanted,” *MUI,* 2022. [Online]. Available: https://mui.com/. [Accessed: 29-Nov-2022].

[6] npm, “Downloading and installing node.js and NPM,” *npm Docs,* 2022. [Online]. Available: https://docs.npmjs.com/downloading-and-installing-node-js-and-npm. [Accessed: 29-Nov-2022].

[7] React, “React – a JavaScript library for building user interfaces,” *– A JavaScript library for building user interfaces,* 2022. [Online]. Available: https://reactjs.org/. [Accessed: 29-Nov-2022].

[8] T. Loginov,“Creating JavaScript bar chart,” *AnyChart News*, 13-Aug-2022. [Online]. Available: https://www.anychart.com/blog/2017/10/25/javascript-bar-chart-tutorial/. [Accessed: 29-Nov-2022].

[9] Unity Technologies, *Unity,* 2022. [Online]. Available: https://unity.com/. [Accessed: 29-Nov-2022].

[10] Vercel, “Develop. preview. ship. for the best frontend teams,” *Vercel,* 2022. [Online]. Available: https://vercel.com/. [Accessed: 29-Nov-2022].