

Greg³

Made by Delta



Introduction to the Team

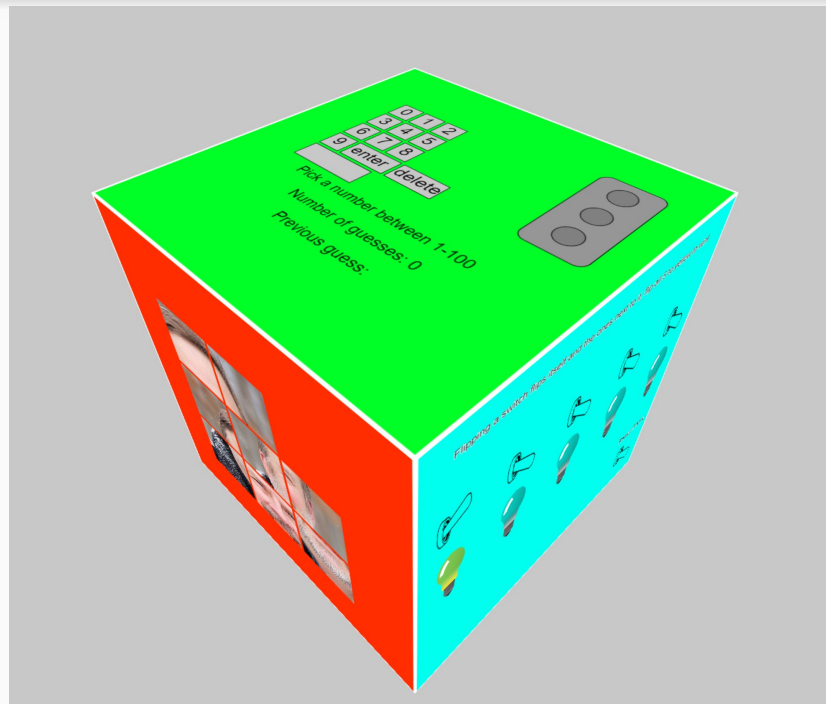
Team Members

- Josh Romisher
- Dakota Sicher
- Josh Thompson
- Julia Pangonis
- Sidney Raabe
- Cyrial Kouatchoua Kamda
- Bailey Wimer
- Jasraj Singh
- Dominic Babusci
- Thomas Moore
- Isak Sabelko
- Mariha Ahammed

Our Project

Greg^3 Project Description

- Greg^3 is a puzzle webtoy built by Delta Group
- Implemented using P5.JS
- Incorporates WebGL
- Series of puzzles on the faces of a 3D rendered cube
- Incorporated class structure and inheritance
- Race against the clock to complete all six puzzles for a final reward screen



Live Demo (Desktop Only)

<https://deltagroupswe.github.io/DeltaMain/public/pages/home/index.html>

<https://tinyurl.com/2nj4d366>

Our Approach

Technologies used

Development Technologies:

- P5.js
- Firebase
- mPicker
- WebGL
- JavaScript
- HTML/CSS
- Git

Project Management Tools:

- Github
- Github Issues/Projects
- Google Drive
- Discord



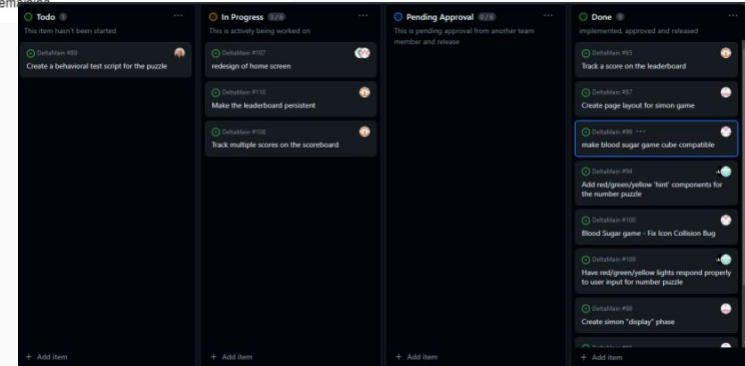
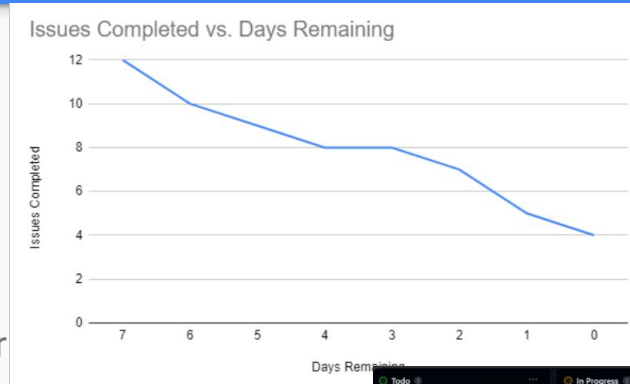
Firebase



Google Drive

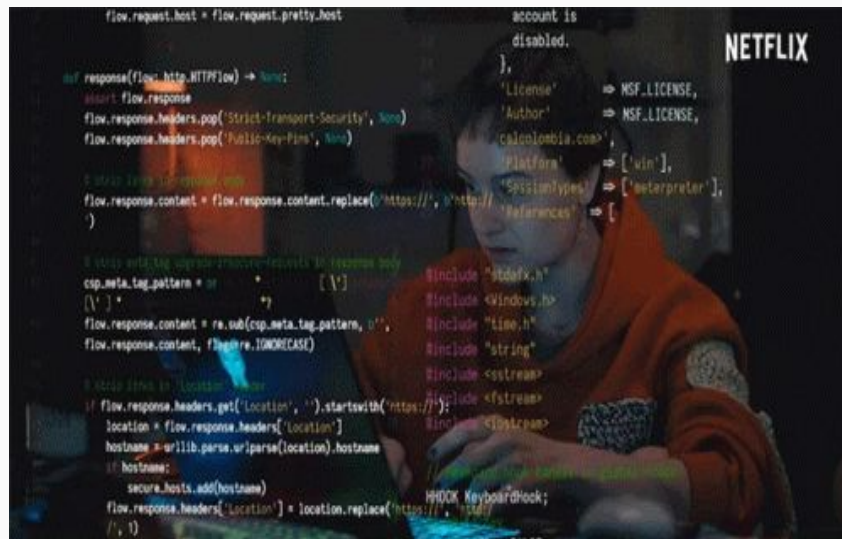
The process methodology

- We used Agile methodology throughout our project.
- The group had weekly scrum meetings to plan our sprints.
- Each sprint, team members used our Kanban board to plan and organize the sprint
- We ended each sprint with a “shippable product”
- Example for sprint 10

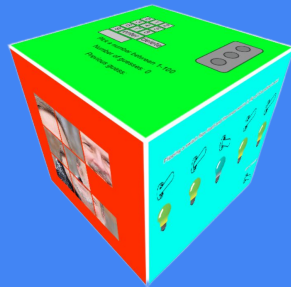


Testing and verification

- We struggled with creating tests given that we had a 3d implementation
- We had used Webdriver to put a text box that would jump to a cube face corresponding with the color typed into the text box
- The project was a learning experience as to the importance of setting up testing from the get-go
- We did all of our work on branches and tested each of the added features on the branches.
 - Once the branches were complete we made a pull request



3D Implementation



The main component of our game is the 3D cube.

The cube is rendered on a canvas using WebGL mode. The user is able to rotate and interact with the cube using the mouse and can select faces and play the game by clicking on a face of the cube. This was accomplished using and updating mPicker.

(<https://github.com/physicsdavid/mPicker>)

Each of the individual puzzle games is implemented using a p5.graphics object. The game draws to the graphics object, which is then applied as a texture on one face of the cube. Mouse and keyboard input is first scaled from its position on the 3D canvas before being passed to a handler method for the selected cube face's puzzle game. The puzzle game then executes the game logic and redraws the graphic to update the cube face.

Project Structure

- Used classes and inheritance
 - Created a base “Puzzle” class for easy puzzle implementation
- Separated assets, components, pages
- Separated our libraries from our source code

```
class Puzzle {
  constructor(renderer, difficulty = 0) {
    this.renderer = renderer;
    this.difficulty = difficulty;
    console.log('Constructing the puzzle!');
  }

  setupGame(){}

  drawGame() {
    console.log('Displaying the puzzle!');
  }

  isSolved() {
    console.log('Checking if the puzzle is solved!');
  }

  handleClick(mx, my) {
    console.log('Handling puzzle\'s mouse event!');
  }

  handleDoubleClicked(mx, my){}

  handleMousePressed(mx, my){}
  handleMouseReleased(mx, my){}
  handleMouseDragged(mx, my){}

  handleKeyPressed(key){}
  handleKeyReleased(key){}
```

```
// This is a base class for all puzzles
// All methods in this class MUST be implemented by all puzzles
// Each puzzle should exist as a class with at least the following methods
class NumberPuzzle extends Puzzle {
  // This is the constructor. Each puzzle should use a difficulty variable,
  // even if the puzzle does not change with different difficulties
  constructor(renderer, difficulty = 0) {
    //this.renderer = renderer; // this is a p5.graphics object
    //optional: could only use one of these as it will be a square graphics object
    //this.w = renderer.width;
    //this.h = renderer.height;

    //this.difficulty = difficulty;

    super(renderer, difficulty);
    this.boxWidth = renderer.width;
    //let randomNumber = Math.floor(Math.random() * 100) + 1;

    this.inputValue = '';
    this.titleText = 'Pick a number between 1-100';
    this.guessCount = 0;
    this.previousGuess = '';
    this.inputValueLength = 0;
    this.maxInputValue = 100;
    this.gameSolved = false;
    this.greenLightColor = 'grey';
    this.yellowLightColor = 'grey';
    this.redLightColor = 'grey';

    console.log('Constructing the puzzle!');
    //...//game specific stuff
    //setting contents for the numPad
    this.numPadContents = {
      0: '0', // ...
      1: '1', // ...
      2: '2', // ...
      3: '3', // ...
      4: '4', // ...
      5: '5', // ...
      6: '6', // ...
      7: '7', // ...
      8: '8', // ...
      9: '9'
    };

    this.enterButton = 'enter';
    this.deleteButton = 'delete';
  }
}
```

```
lib
├── addons
├── mPicker
├── p5.js
├── p5.min.js
├── README.md
├── style.css
├── public
├── assets
├── csvs
├── sounds
├── sprites
├── components
├── hintButton
├── puzzles
├── timer
├── button.js
├── pages
├── cube
├── gallery
├── home
├── leaderboard
├── .gitignore
└── README.md
```

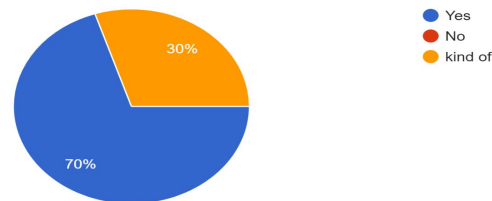
Feedback

Survey Results

- Overall people found our instructions easy to follow
- People were split on the visual appeal as well as how easy it was to play

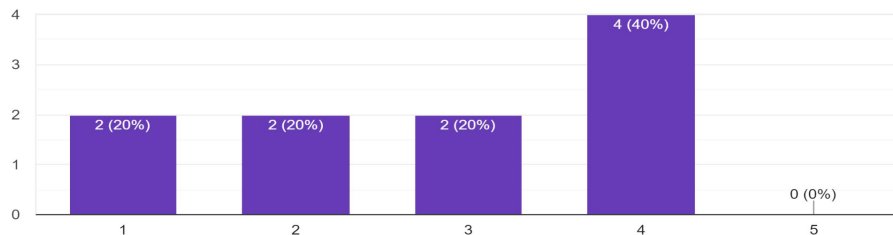
Were the instructions easy to follow?

10 responses



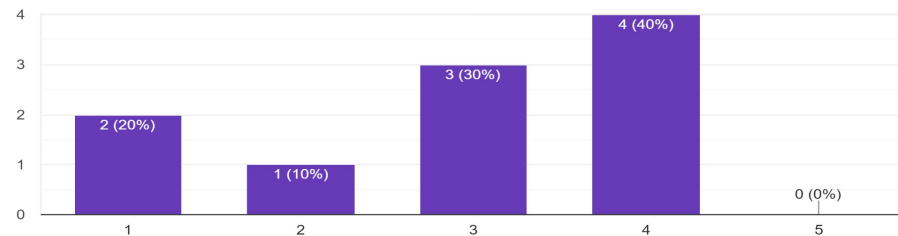
How would you rate the visual aspect of the game

10 responses



How easy was the game to play

10 responses



“easy to navigate”

“Very bright & visually
appealing ”

“Funny face”

“simple enough to understand
to get me started.”

“The instructions page was
useful. Most of the puzzles
were self explanatory.”

“it would've been a bit easier if controls for each game were shown while playing”

“Some things were not explained”

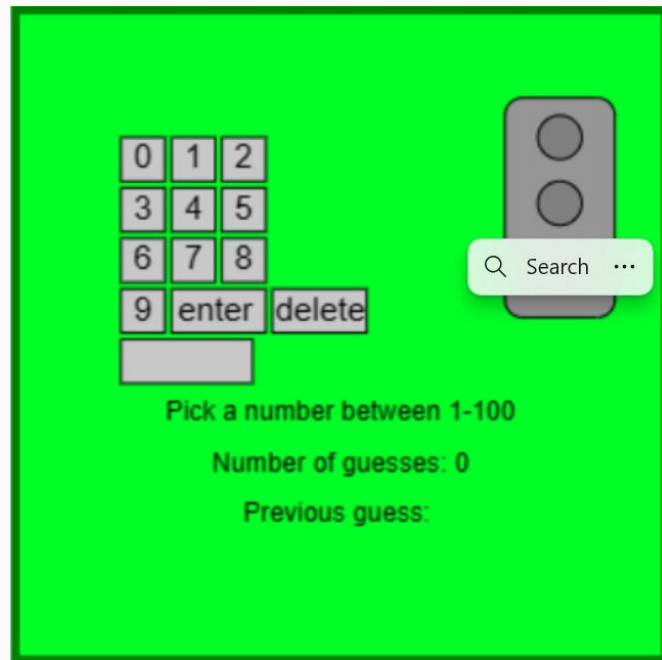
“Add instructions for the blue game because I didn't understand the purpose”

“if you make me do math again i'll turn you inside out with a hydraulic press”

Individual Contributions

Josh Romisher

- Worked on number puzzle
- Implemented mostly structural elements of number puzzle along with some logic
- Helped translate project into a class based implementation versus all standalone functions
- Learned a lot about working with JavaScript and it's associated libraries



Julia Pangonis

- Originally we created the puzzle using standalone functions without organization
- We then used our renderer to create classes to work better with other puzzles
- I helped develop the logic for the Number Puzzle, along with the number pad and the light to determine how close the guess is

```
// This is a base class for all puzzles
// All methods in this class MUST be implemented by all puzzles
// Each puzzle should exist as a class with at least the following methods
class NumberPuzzle extends Puzzle {
  // This is the constructor. Each puzzle should use a difficulty variable,
  // even if the puzzle does not change with different difficulties
  constructor(renderer, difficulty = 0) {
    //this.renderer = renderer; // this is a p5.graphics object
    //optional: could only use one of these as it will be a square graphics object
    //this.w = renderer.width;
    //this.h = renderer.height;

    //this.difficulty = difficulty;

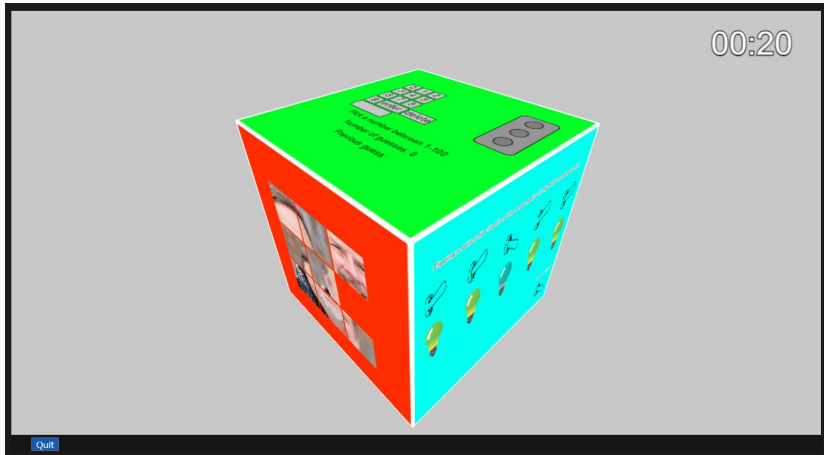
    super(renderer, difficulty);
    this.boxWidth = renderer.width;
    //let ranNumber = Math.floor(Math.random() * 100) + 1;

    this.inputValue = '';
    this.titleText = 'Pick a number between 1-100';
    this.guessCount = 0;
    this.previousGuess = '';
    this.inputValueLength = 0;
    this.maxInputValue = 100;
    this.gamesSolved = false;
    this.greenLightColor = 'grey';
    this.yellowLightColor = 'grey';
    this.redLightColor = 'grey';

    console.log("Constructing the puzzle");
    //...//game specific stuff
    //Setting contents for the numPad
    this.numPadContents = [
      0: '0', //      ... ..
      1: '1', //      | 0 | 1 | 2 |
      2: '2', //      ... ..
      3: '3', //      | 3 | 4 | 5 |
      4: '4', //      ... ..
      5: '5', //      | 6 | 7 | 8 |
      6: '6', //      ... ..
      7: '7', //      | 9 | enter | delete |
      8: '8', //      ... ..
      9: '9'
    ];

    this.enterButton = 'enter';
    this.deleteButton = 'delete';
  }
}
```

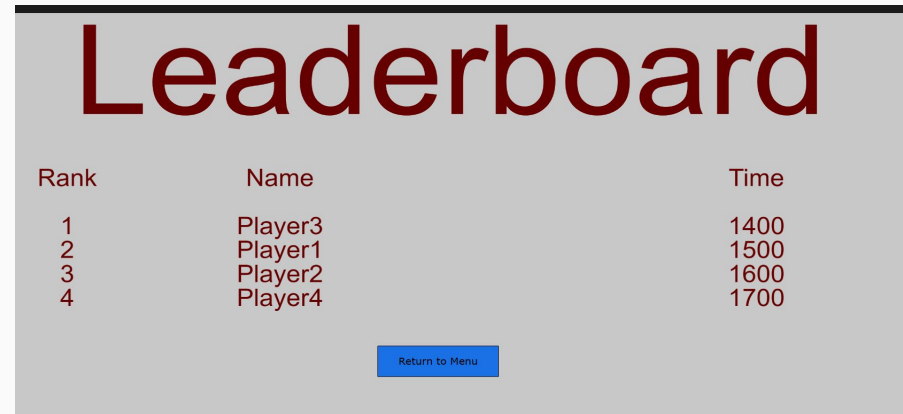
Dakota Sicher



- Developed the 3D implementation
 - Class scheme for puzzles
 - Rendering of games on the cube
 - Input handling
 - 3D object selection
 - Cube movement
- Instructional popup
- Win Screen Animation

Josh Thompson

- Visual elements on the Leaderboard page
- Custom Button class (cut)
- Learned:
 - Javascript
 - Github in groups
 - Communication in large groups

A screenshot of a web page titled "Leaderboard" in a large, dark red serif font. Below the title is a table with three columns: "Rank", "Name", and "Time", all in a dark red sans-serif font. The table contains four rows of data. At the bottom right of the page is a small blue rectangular button with the text "Return to Menu" in white.

Rank	Name	Time
1	Player3	1400
2	Player1	1500
3	Player2	1600
4	Player4	1700

Return to Menu

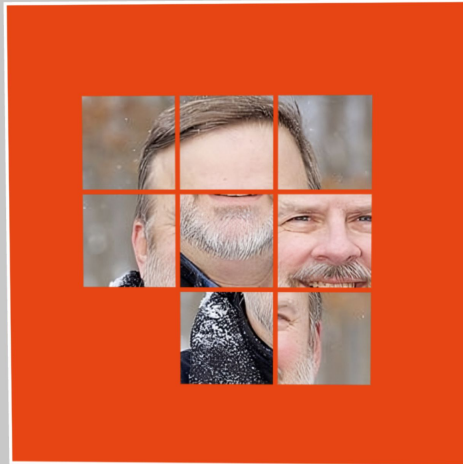
Sidney Raabe

Main Contributions:

- Slider Puzzle

What I Learned:

- GitHub and branches
- Agile methodology in practice



```
class SliderPuzzle extends Puzzle {
  constructor(renderer, difficulty = 0) { // this is a p5.graphics object
    //optional: could only use one of these as it will be a square
    super(renderer, difficulty)
    this.w = renderer.width;

    //console.log('Constructing the puzzle');
    this.puzzleState = {
      A1: "A1", //      ----
      B1: "B1", //      | A1 || B1 || C1 |
      C1: "C1", //      ----
      A2: "A2", //      | A2 || B2 || C2 |
      B2: "B2", //      ----
      C2: "C2", //      | A3 || B3 || C3 |
      A3: "A3", //      ----
      B3: "B3", //      ----
      C3: " " //      C3 is empty in final puzzle state
    };
    this.emptyCell = "C3";
  }

  setupGame() {
    this.boxSize = this.w / 5; // Size of each box
    this.spacing = this.w / 100; // Spacing between boxes
    this.startX = this.w / 6; // Starting X position
    this.startY = this.w / 5; // Starting Y position
    this.debug();
    this.shufflePuzzleState();
  }

  // This is the draw function for the puzzle.
  // Everything that the puzzle needs to do to display on the screen
  drawGame() {
    let emptyCell = this.getEmptyCell();
    //console.log('Displaying the puzzle')
    //this.debug();
    let x = this.startX;
    let y = this.startY;
  }
}
```

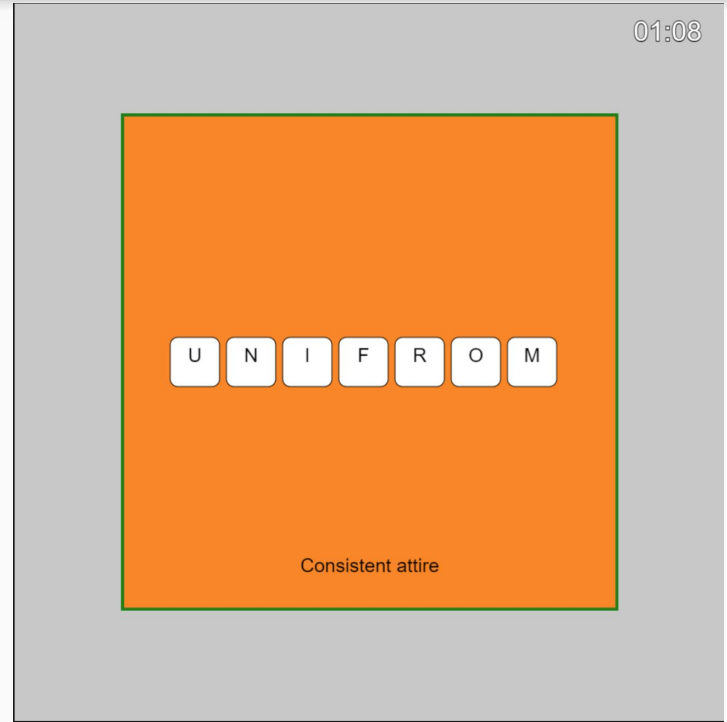
Cyrial Kouatchoua Kamda

Word Puzzle:

- Built the original class based implementation of the word puzzle engine
- Built the implementation of the word puzzle tile system for display
- Worked on the result display and initial inheritance feature integration of the word puzzle from the puzzle base class

What I Learned:

- I have never worked with a gaming system before
- I have never worked with p5.js before



Bailey Wimer

“Greg Says” Puzzle:

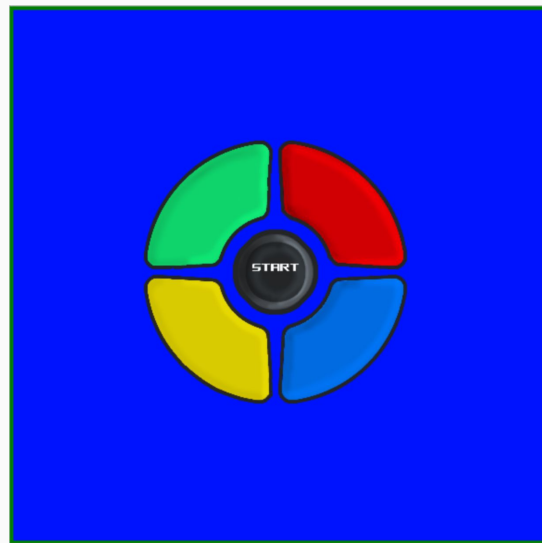
- Created class for game logic, including “idle”, “display”, and “input” states
- Created game art based on Simon Says game

Overall Project:

- Created project organization structure
- Created base class for puzzle implementation

What I learned:

- How to operate more successfully in an Agile environment



Jasraj Singh

Blood Sugar Puzzle:

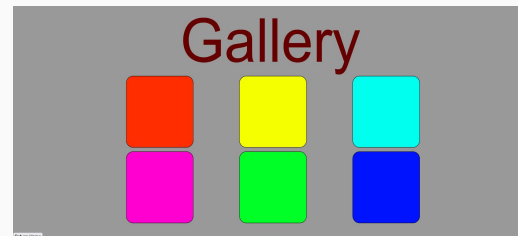
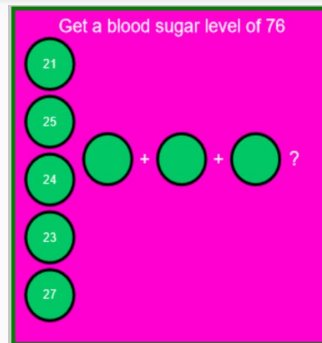
- Developed Icon, Container, and UI Classes
- Implemented a Puzzle manager to handle game objects

Gallery Page:

- Cut from final product
- Would have been available after completion

Things learned:

- How to take an idea into a finished product
- First-hand experience with sprint



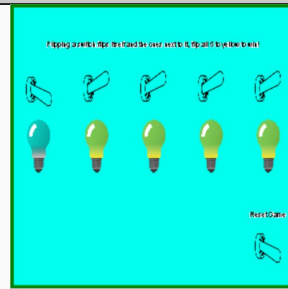
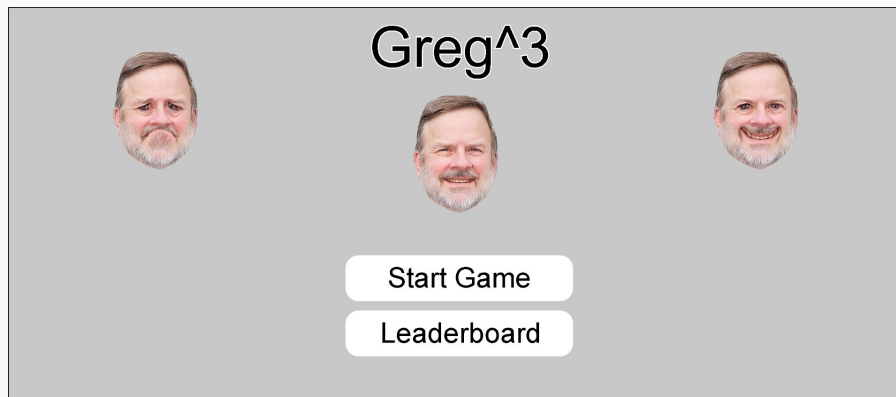
Dominic Babusci

Home Page:

- Added new navigation buttons with on hover capabilities
- Created home screen “Greggy” design

Switch Puzzle:

- Originally used a class based implementation
- Later changed the class to extend the base puzzle class and use the renderer



Thomas Moore

- Created Timer class
 - Helped implement timer onto cube screen.
- Created functionality for leaderboard
 - Tested three different methods
 - Cookies, JSON, and firebase
 - Not fully implemented to firebase
- Refactored leaderboard code for visuals
- What I learned
 - Agile methodologies in a controlled environment
 - Git, javascript, JSON, and the importance of version control

```
class Leaderboard {
  constructor() {
    this.topTextSize = winWidth * 0.13;
    this.topTextX = winWidth / 2;
    this.topTextY = winHeight * (2 / 14);

    this.midTextSize = winWidth * 0.025;
    this.midTextXRank = winWidth / 8;
    this.midTextXName = winWidth / 3;
    this.midTextXTime = winWidth / 1.25;
    this.midTextY = winHeight * (5 / 14);
  }
}
```

```
class Timer {
  constructor() {
    this.startTime = 0;
    this.elapsedTime = 0;
    this.isRunning = false;
  }

  // function for setting up timer
  setupTimer() {
    this.startTime = millis();
  }

  // function for updating timer
  updateTimer() {
    if (this.isRunning) {
      this.elapsedTime = millis() - this.startTime;
    }
  }

  // function for starting timer
  startTimer() {
    if (!this.isRunning) {
      this.startTime = millis() - this.elapsedTime;
      this.isRunning = true;
    }
  }

  // function for stopping timer
  stopTimer() {
    if (this.isRunning) {
      this.isRunning = false;
    }
  }

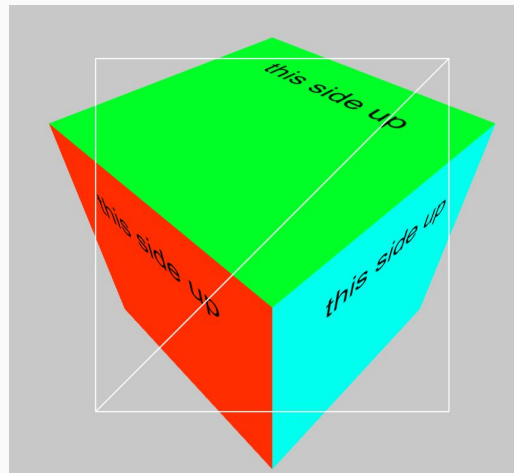
  // function for resetting timer
  resetTimer() {
    this.isRunning = false;
    this.elapsedTime = 0;
    this.startTime = millis();
  }

  // function for formatting time from int to string
  formatTime(time) {
    let seconds = Math.floor(time / 1000);
    let minutes = Math.floor(seconds / 60);
    seconds = seconds % 60;
    return nf(minutes, 2) + ':' + nf(seconds, 2); // Formatting int to strings
  }
}
```

00:41

Isak Sabelko

- Original Cube Implementation:
 - I had created the original implementation of the cube with different colors for each face
 - Also allowed for users to rotate the cube and click on one of the cubes faces to jump to the specific face
 - We since moved to the new rendering to allow games to be directly on the face instead of just colors or potential images of the games
- Switch Puzzle:
 - I had added physical switches and light bulbs to the game
 - I also implemented sounds as well as the win or lose screen
- What I Learned:
 - I learned that the first implementation is not always the best implementation and that a project will always change to meet requirements
 - I learned a lot about working together as a big group in order to be efficient in making sure all requirements are met and in a reasonable span of time



Mariha Ahammed

- Hint button troubleshooting and implementation
- Established communication channels
- Never really worked with p5js before so learned a lot about that.
- Also got a deeper insight into team collaboration through kanban/github

Outcomes

What we learned

- Splitting work into individual tasks for weekly sprints
- How to function in an Agile environment
- Using GitHub to track a project
- Deploying a web app to GitHub pages
- How to use the p5.js JavaScript library
- Use WebGL to generate 3d visuals
- Use classes to create modules
- Developing a project with a large team

```
import java.sql.*;
import java.awt.*;

/**
 *
 * @author Jeff
 */
public class Main {

    public static String AppName = "SQL Mail";
    public static String AppVersion = " 0.0.1 ";
    public static String AppAuthor = "Jeffrey Cobb";
    public static String AppDate = "August 8th, 2007";
    public static String AppPath = System.getProperty("user.dir");
    public static String AppDriver = "smallsql.database.S50driver";
    public static String AppDBHeader = "jdbc:smallsql:";
    public static String AppDBPath = AppPath + "/sqlmail";
    public static String AppPreferences = AppPath + "/sqlmail_prefs";
    /** Creates a new instance of Main */
    public Main() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception {
        // TODO code application logic here

        boolean bDBConnect = false;
        int result = 0;
        FrmMain SQLMailForm = new FrmMain();
        System.out.println("\r\n" + AppName + "\r\nVersion" + AppVersion + "\r\nAuthor: " + AppAuthor +
        .. " + AppDate + "\r\n");

        Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension screen = tk.getScreenSize();
        System.out.println(screen.getWidth() + " --- " + screen.getHeight());
    }
}
```

RESULTS:
Simply_Millie3

The Future

What we are gonna do

- Implement more levels
- Add instructions per puzzle
- New puzzles
- Fix interactions with current puzzles
- Address user feedback

Thanks!

Thank you for spending some time to check out **Greg^3**

Any feedback/questions?

Thanks!

-Delta

