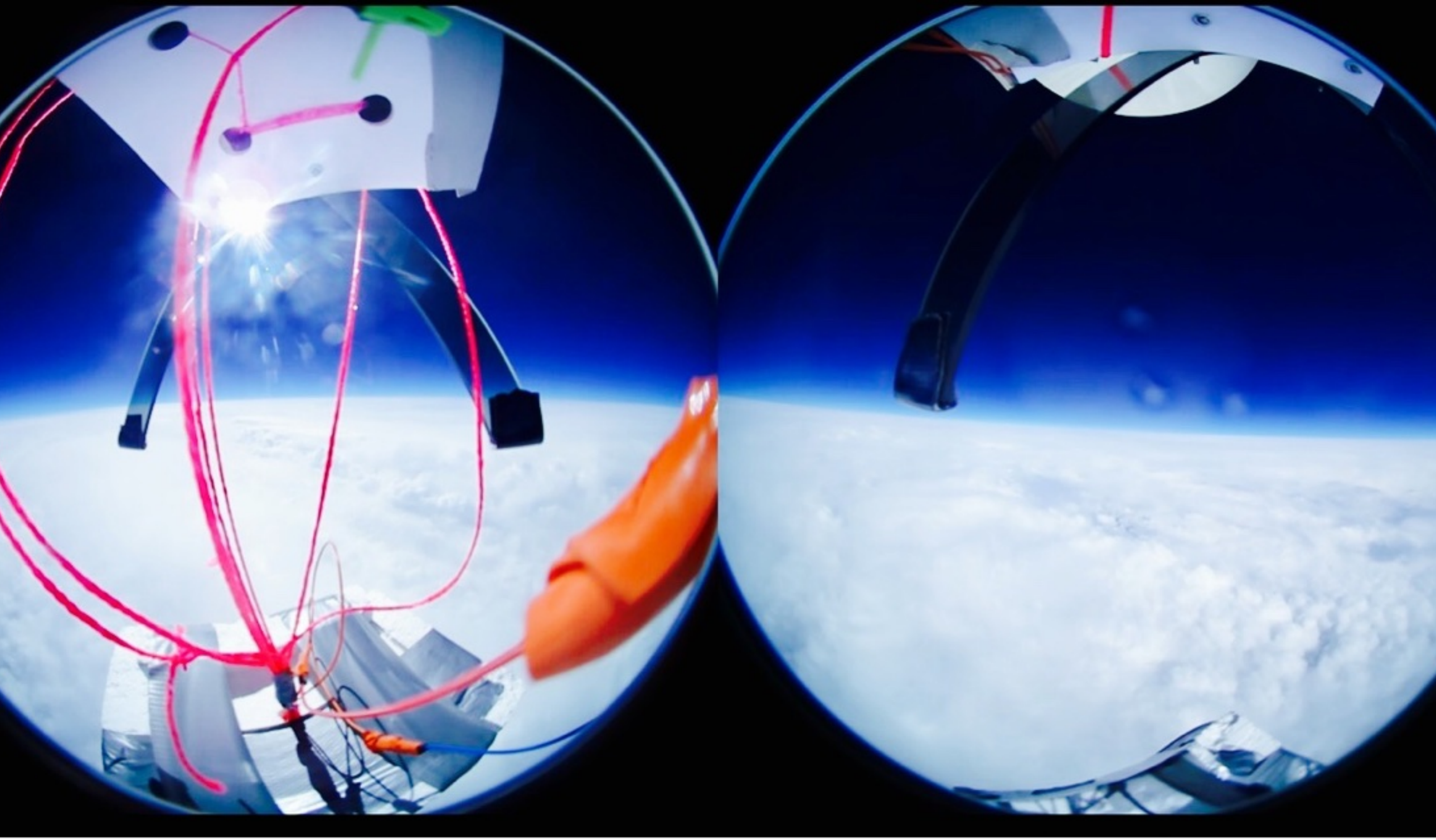
Engineering Portfolio

Alexis Mitchnick
University of Pennsylvania School of Engineering and Applied Science
Major: Mechanical Engineering, Minors: Computer Science, Mathematics



HIGH-ALTITUDE BALLOON

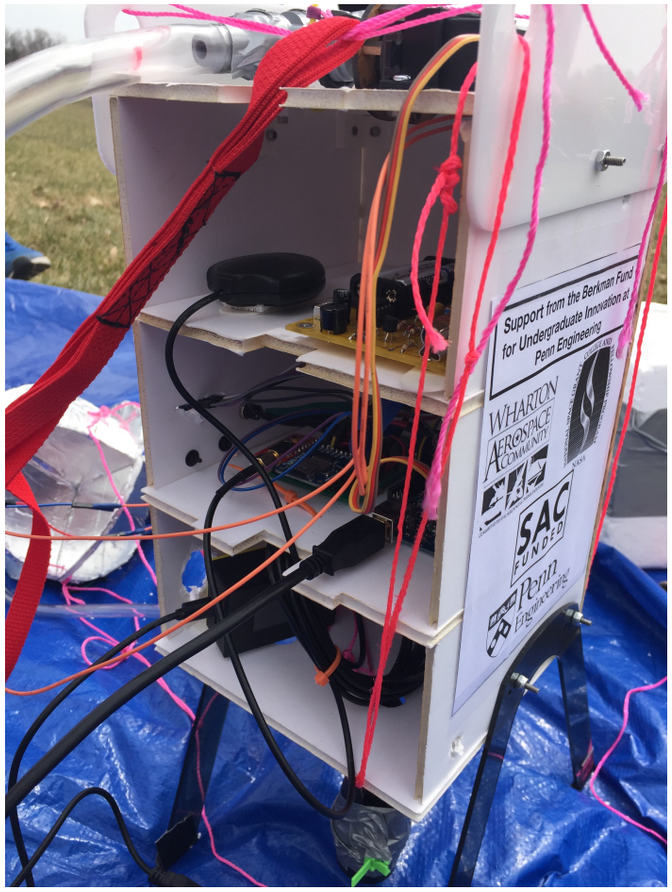
PENN AEROSPACE CLUB
BALLOON TEAM

Project Summary

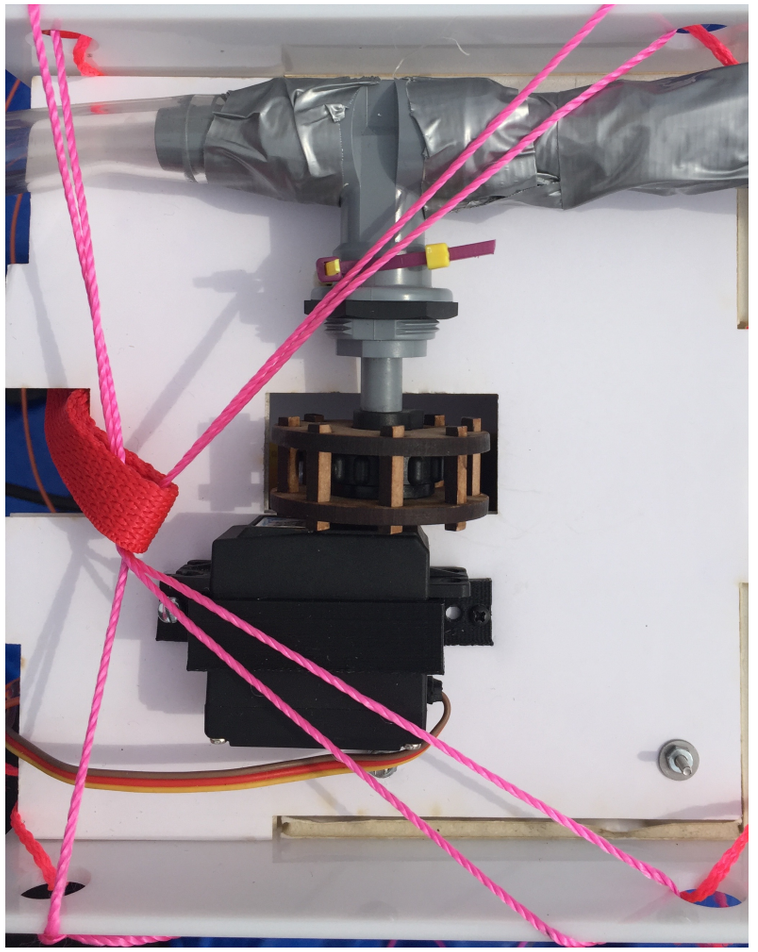
As the Altitude Control Engineer for the Penn Aerospace Balloon Team, I work with a select group of fellow engineering students to design, test, and implement a complex, on-command venting system for high-altitude balloons.

Software

- **Simulink and Matlab:** We generated controlled and non-controlled balloon trajectory simulations (in both Matlab and Simulink) to obtain the data necessary to create a successful and robust altitude control system.
- **Arduino:** We programmed an Arduino Uno to control the movement of continuous rotation servo



This is our complete payload from the most recent balloon launch in April 2017.



This is our altitude control system mounted to the top of the payload shown to the left.

Electro-Mechanical Design

- Our design includes a continuous-rotation servo, a mounting cage, and a needle port valve connected to a tube that leads to the balloon.
- Designed and laser-cut the mounting cage that connects the servo to the valve.
- **How it works:** After the balloon reaches a specific altitude (decided based on simulated trajectory), the Arduino alerts the servo to rotate 90 degrees to open the valve and release helium. Once the valve has been open for a specified time span, the Arduino alerts the servo to rotate back, causing the valve to close and the helium flow to stop.

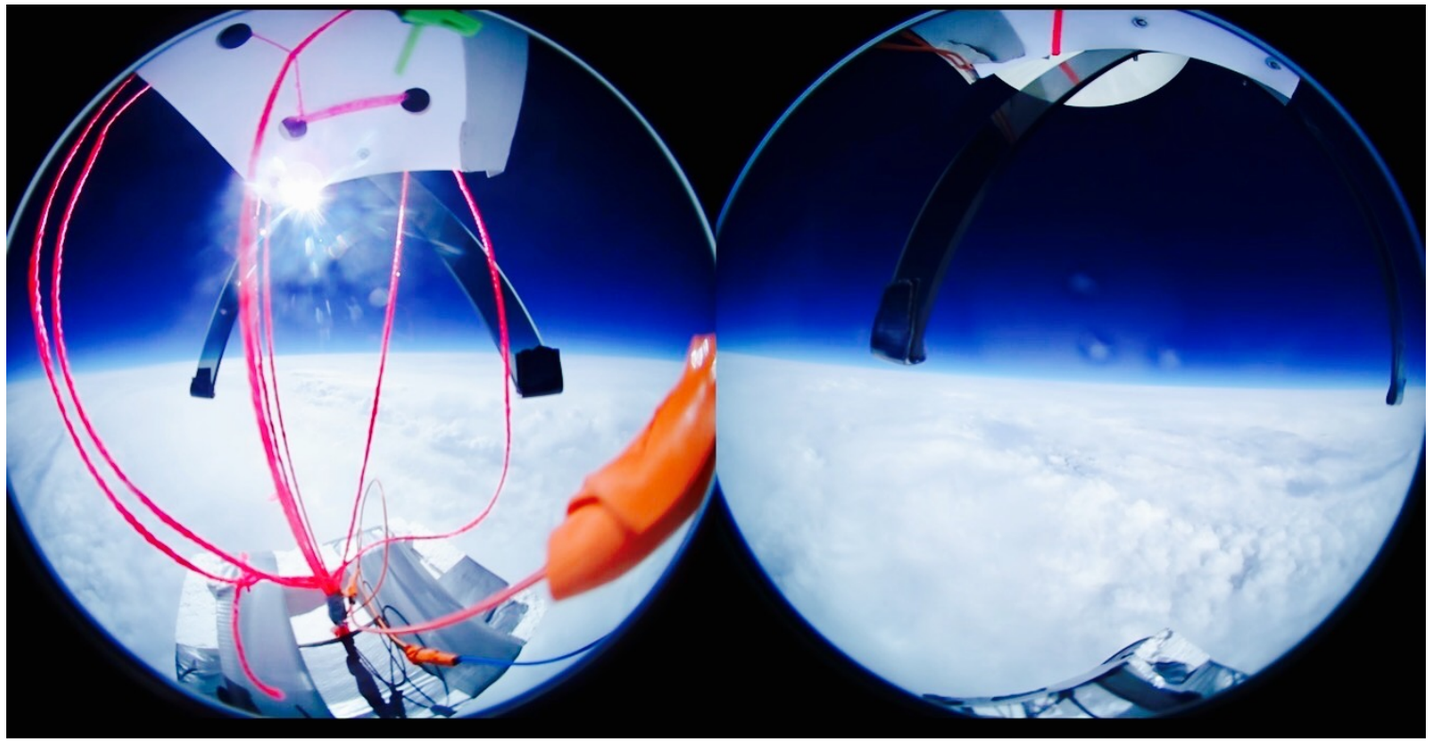
Team Collaboration

- Participate in team-wide preliminary and critical design reviews
- Attend bi-semester balloon launches, where the team tests each aspect of the payload
- Use GitHub version control repository to collaborate with other team members on Arduino and Matlab code

Learn More

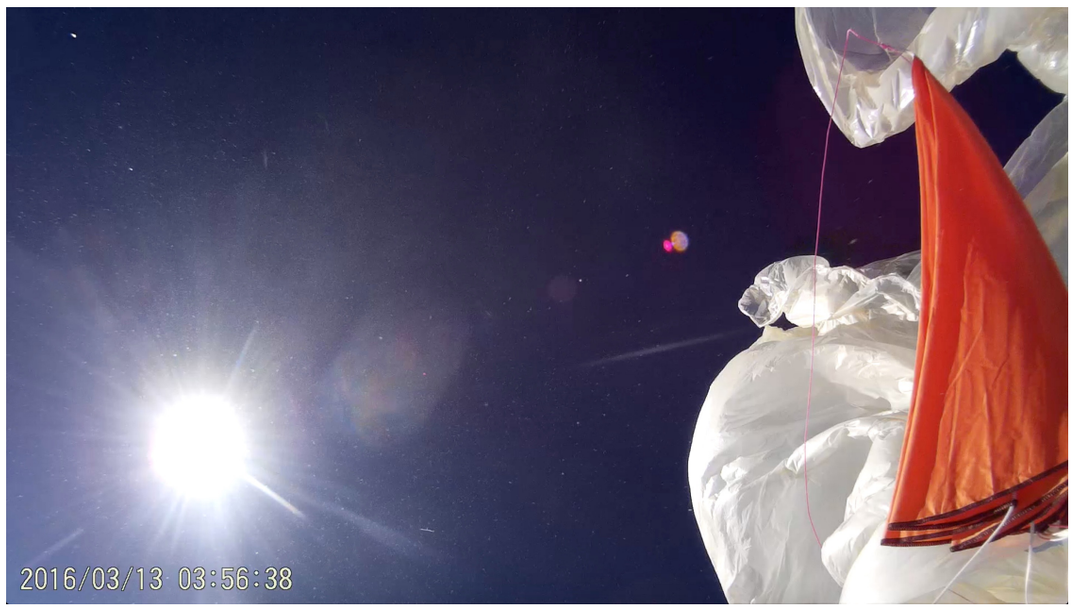
If you wish to learn more about this project or the Penn Aerspace Club, visit our site **pennaero.com** or our GitHub repository, **github.com/PennAerospaceClub**

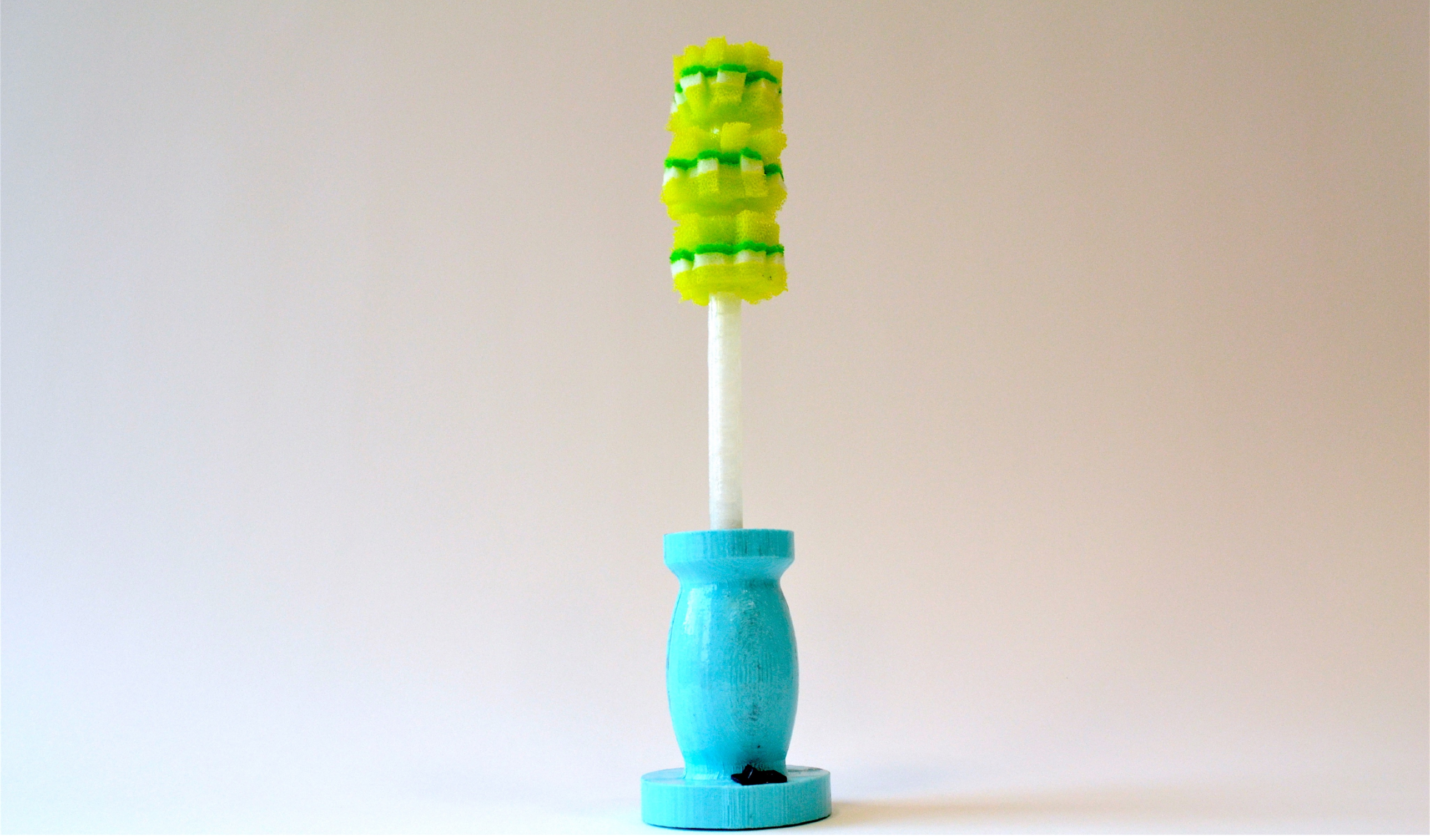
Capturing near-space



The photos below were taken from a wide-angle camera mounted inside our payload

The above photo was captured from a 360 camera, hanging from the bottom of our payload





MOTORIZED BOTTLE BRUSH

Project Summary

As a semester-long team project in a Product Design and Development class, I collaborated with three other students to design, model, build, and market a motorized water bottle brush.

Electro-Mechanical Design

- **Handle and Rod:** 3D-printed, designed with Solidworks with ergonomics and aesthetics in mind
- **Electronics:** Battery pack and compact dc motor rest inside the handle and button is embedded in the base.
- **Brush Sponges:** Hand-cut brush ‘bristles’ from ordinary sponges, which were adhered to the rod for strength and cleaning power

Product Stats

- **How it Works:** When the button is switched on, the rod rotates at ~200 RPM to provide minimal-effort, maximum-strength cleaning.
- **Dimensions:** Brush is 10” long from base to tip and sponges are 2.5” in diameter

Marketing and Research

- Performed multiple user interviews to determine optimal design and understand market preference
- Researched current similar products in the market to analyze other solutions and potential competition
- Presented multiple design pitches to the class to obtain user feedback
- Brainstormed and defined a brand name, logo, and tagline to market our product at class-wide design fair
- After creating prototype, performed proof-of-concept user testing

This is the final prototype for our product.

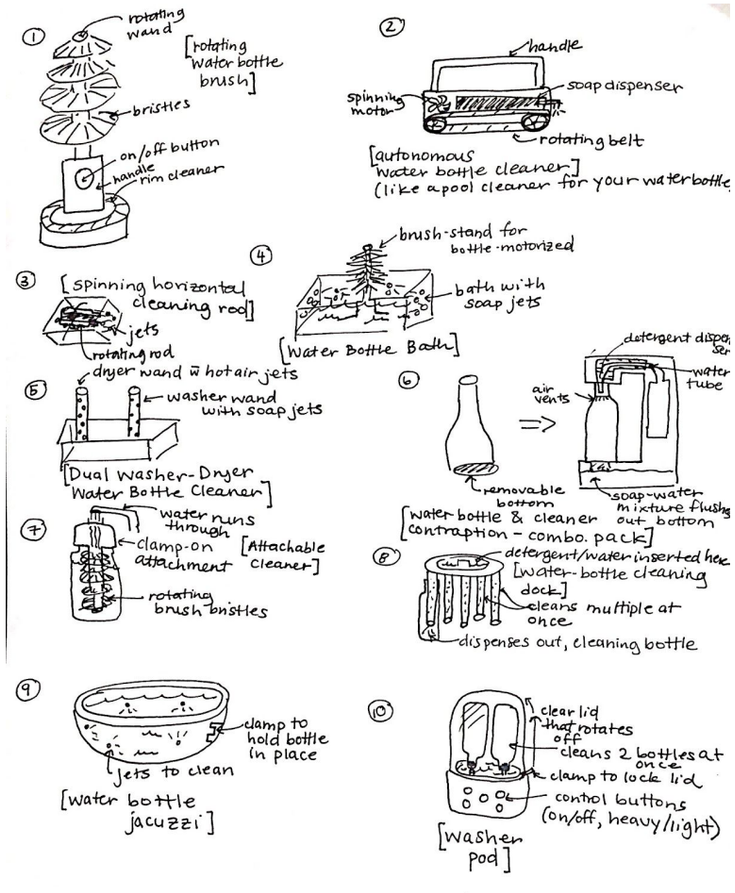
This is the sales flyer for our product, including our brand name, scrubd, a tagline, and graphics.

Design Iteration

- Generated sketches for 10 different potential bottle-cleaner products
- Created schematic drawings of chosen design concept
- Built several preliminary prototypes before presenting final prototype at design fair
- Modeled design concept in Solidworks and created exploded assembly before prototyping

Design Process

These images are three different views of our Solidworks rendering of an earlier iteration of the product (the third is an assembly exploded view, showing the electronics inside the handle)



These are our concept-generation sketches, which explore different product possibilities meeting the bottle-washing need

IWWMW... IMPROVE CLEANING OUR WATER BOTTLE: with the **MOTORIZED WATER BOTTLE BRUSH**

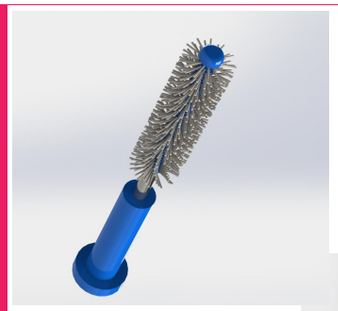
ALEXIS MITCHNICK & DANIELLE GELB

Helps busy students efficiently wash their water bottles (which often get neglected!)

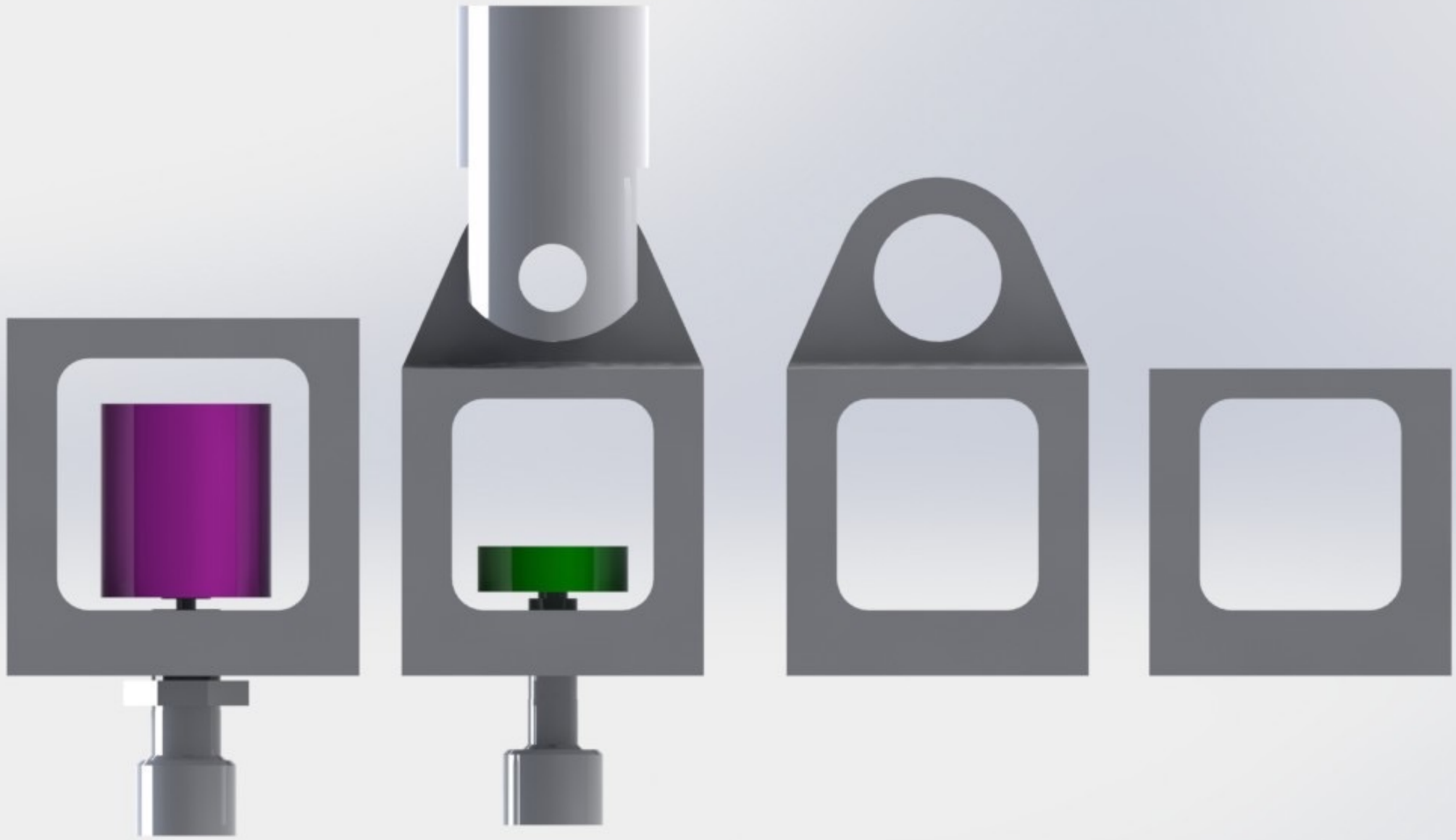
USER NEEDS:

- Product sufficiently cleans water bottle
- Quick and easy to use
- Universal use (works for any bottle)

- Has a flat base - can sit it on your countertop!
- Dispenses soap
- Brush spins automatically when turned on
- Battery operated
- Perfect for easy, quick, and efficient water-bottle-washing!



The slides to the left are part of the visual that accompanied one of several design pitches, explaining the needs satisfied by our product



ROEHRIG DYNO FIXTURE

ANZE SUSPENSION

Project Summary

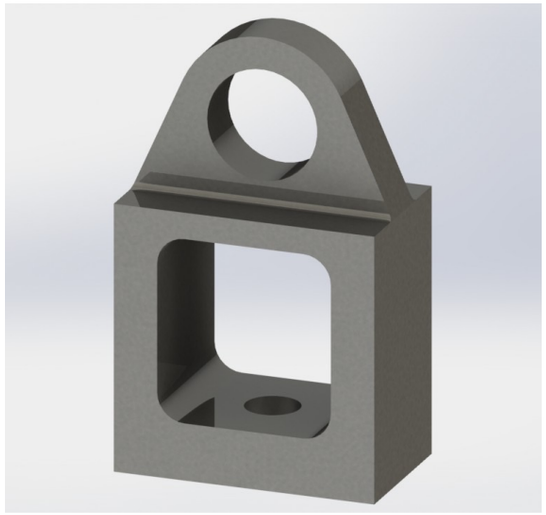
As a mechanical engineering intern at ANZE Suspension, I designed a tool to streamline the installation of race car dampers into a Roehrig Dyno testing machine.

Analysis and Outlining

- Performed a motion study of shock technician, analyzing complete damper service procedure to devise solutions to streamline damper testing in the Roehrig Dyno machine
- Took measurements of all damper adjusters and machine components to determine correct dimensions for the tool
- Determined which damper shaft thread sizes appeared most often to resolve which hole sizes should be machined and optimize tool usage
- Performed cost-benefit analysis of potential tool materials
- Created a project outline that highlighted a problem statement, design goals, design requirements and variables, and potential solutions

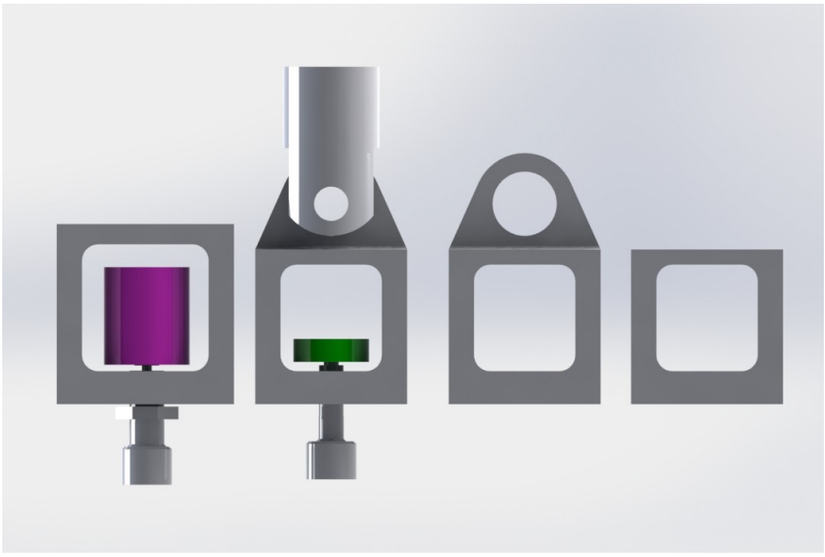
Tool Modeling and Testing

- Created 6 different design models in Solidworks (4 shown in rendering on the right)
- Utilized Solidworks configuration modeling to display different thread sizes for bottom hole
- Examined FEA data in Solidworks to determine part strength and durability
- Generated 2D drawings in Solidworks for prototype production, including dimension tolerances for necessary press fits
- Created family assemblies in Solidworks to analyze interactions between the fixture, the dyno, and the damper adjusters



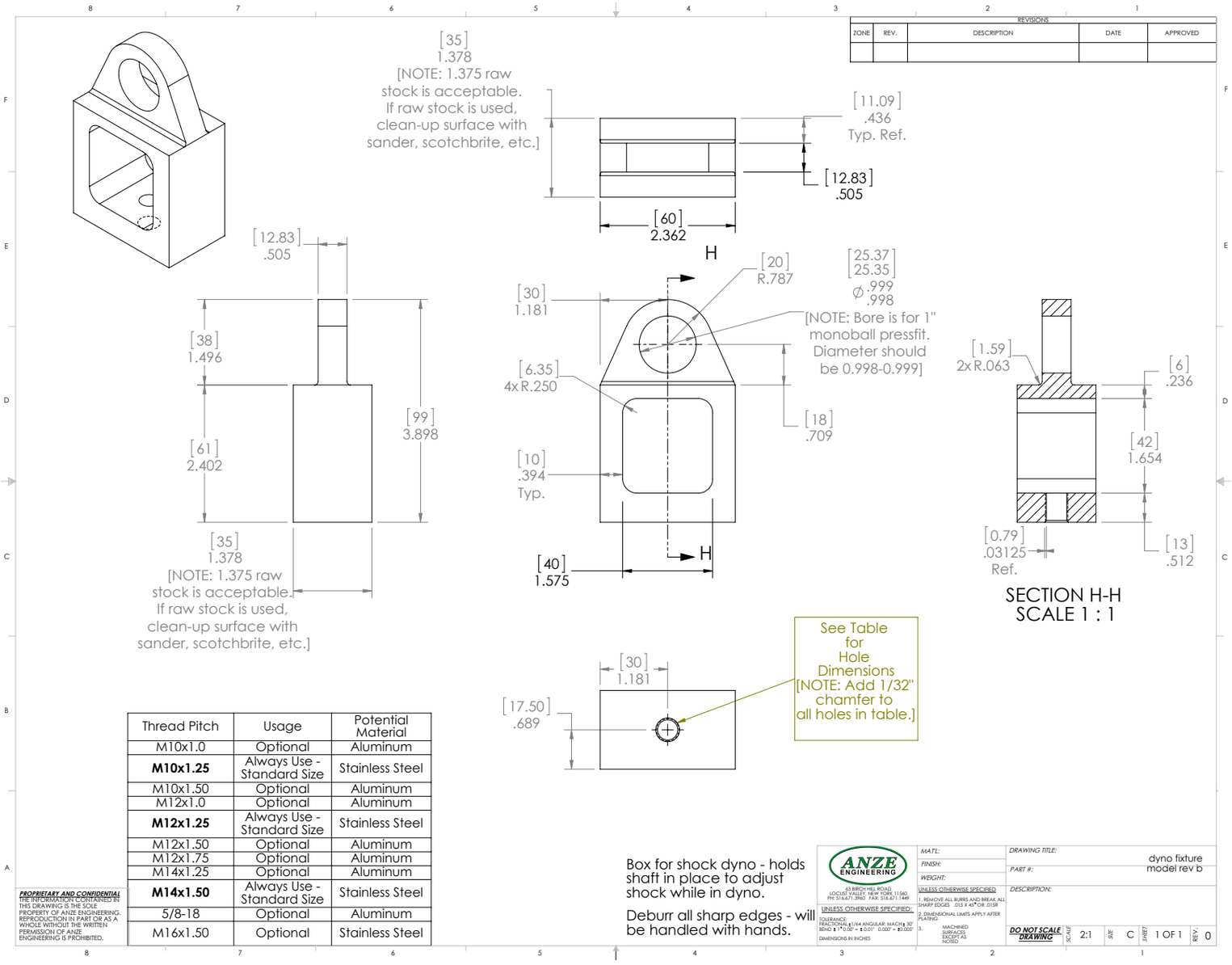
Solidworks rendering of a segment of the family assembly for the fixture, including 4 different fixture models, the dyno clevis, and two different damper adjusters

Solidworks rendering of the final design for the dyno fixture



Results

- Fixture improves damper testing efficiency by ~15 mins/day by eliminating the need to remove the damper to adjust it - now shock technician can adjust dampers with ease while they remain installed in the machine
- 3 different prototypes, one for each of the most common shaft thread sizes, were machined in _____ stainless steel for maximum strength and durability
- 1” monoball was pressfit into top eyelet so fixture can be easily installed using a clevis attached to the dyno



Thread Pitch	Usage	Potential Material
M10x1.0	Optional	Aluminum
M10x1.25	Always Use - Standard Size	Stainless Steel
M10x1.50	Optional	Aluminum
M12x1.0	Optional	Aluminum
M12x1.25	Always Use - Standard Size	Stainless Steel
M12x1.50	Optional	Aluminum
M12x1.75	Optional	Aluminum
M14x1.25	Optional	Aluminum
M14x1.50	Always Use - Standard Size	Stainless Steel
5/8-18	Optional	Aluminum
M16x1.50	Optional	Stainless Steel

This is the Solidworks 2D drawing I created, which was sent to the company machine shop for prototype manufacturing

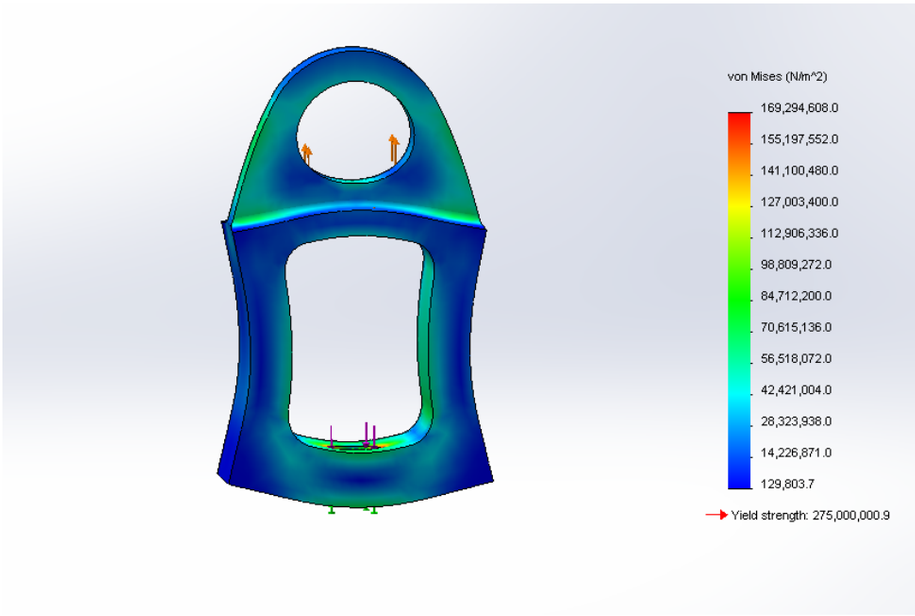


Image of FEA results when the tool was tested under the necessary condition of 3000 lb in tension

Dyno Box Fixture

Problem Statement

- Want a tool to streamline installation of dampers into dyno
- Want to be able to adjust dampers while they are in the dyno - currently have to take damper out of dyno, take eyelet off, adjust damper, reinstall eyelet, put damper back in dyno, test damper

Design Goals

- Shorten time needed to complete dyno test on damper

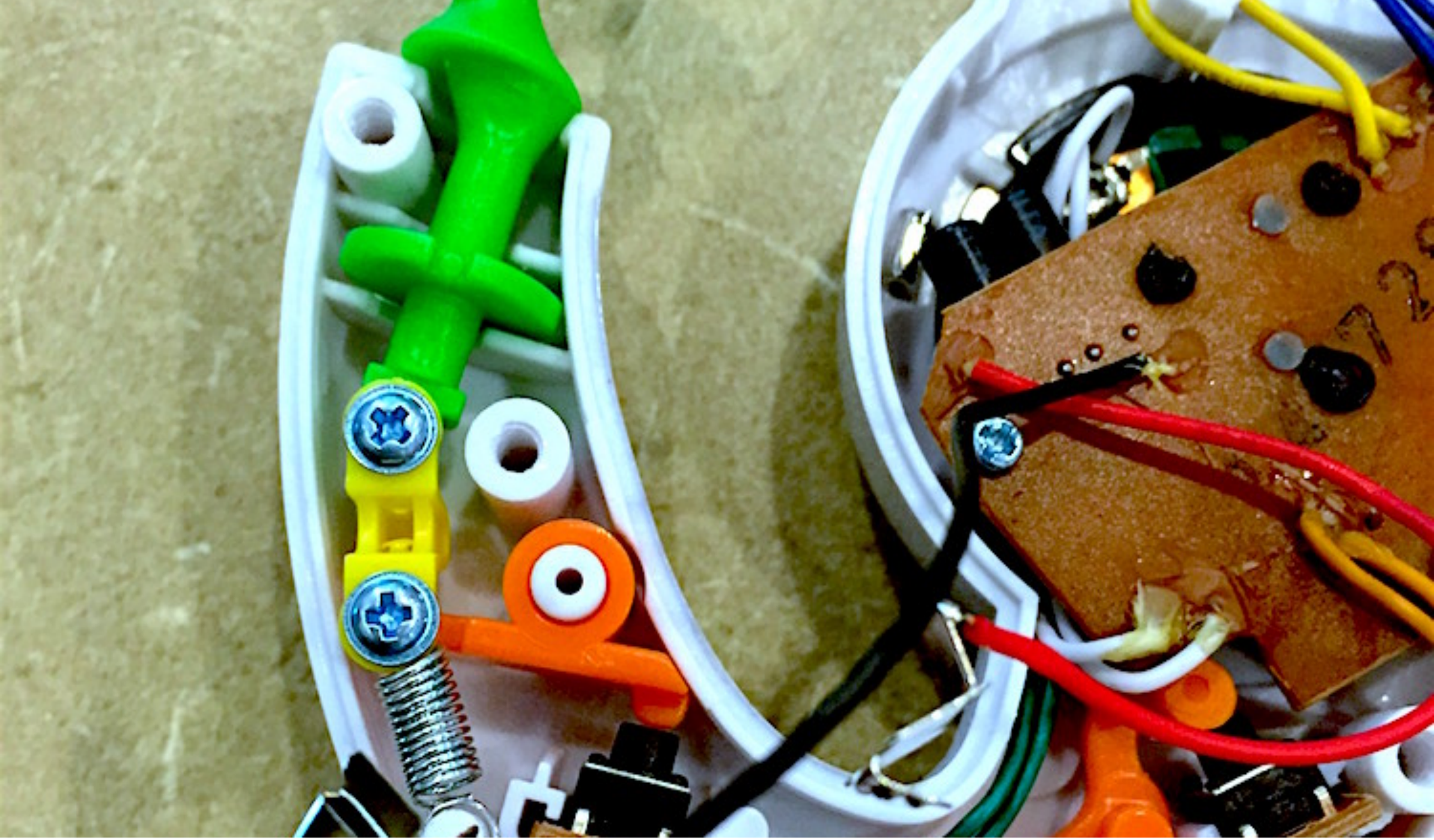
Givens

- Strongest dampers produce 300 lb compression, 2500-3000 lb rebound
- Largest adjuster knob dimensions that must fit in box:
  - X - 30.0 mm diameter (if including Koni 86-10,11: 46.3 mm)
  - Y - 17.3 mm height (if including Koni 86-10,11: 36.8 mm)
- Shortest stud threads length: ~20 mm (double check)
- Bottom of clevis hole to bottom of clevis: 6mm

Requirements

- Must allow access to adjuster (in order to be able to adjust damper while it is still in dyno)
- Solution must accommodate all the different size shaft threads
- Must be strong enough to withstand the force of the strongest dampers (see given)
  - Factor of safety = ____
- Should be short vertically (cannot add too much length vertically, otherwise longer dampers would not fit into machine)
- Should be narrow while still fitting all knobs and allowing fingers to fit in to adjust knob
- Eyelet should have monoball press fit inside
- Threads should not exceed length of threads on shortest stud
- Eyelet and box configuration must not interfere with dyno clevis

Excerpt from the project outline I devised



DISSECT

Project Summary

As a modeling challenge in Introduction to Mechnical Design, I, along with two others, deconstructed an electro-mechanical children's toy, Bop-It, with 40+ parts and modeled each part in Solidworks to create a functional assembly

Team Collaboration

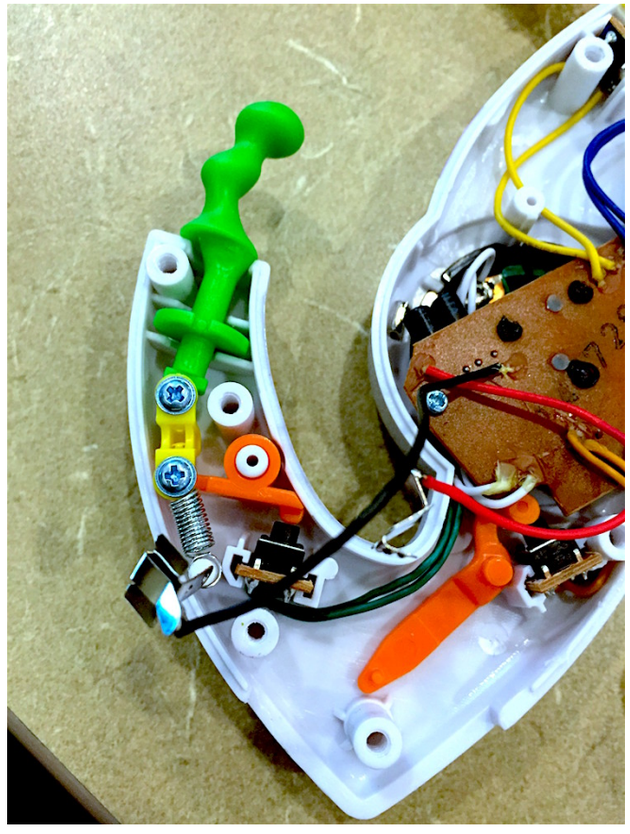
- Worked with two other students to complete the Bop-It model
- Dissected model as a team and delegated different pieces to complete the project in an efficient and timely manner
- Responsible for modeling all internal electronics, springs, buttons, and the central outer "Bop-It" button

Project Outcome

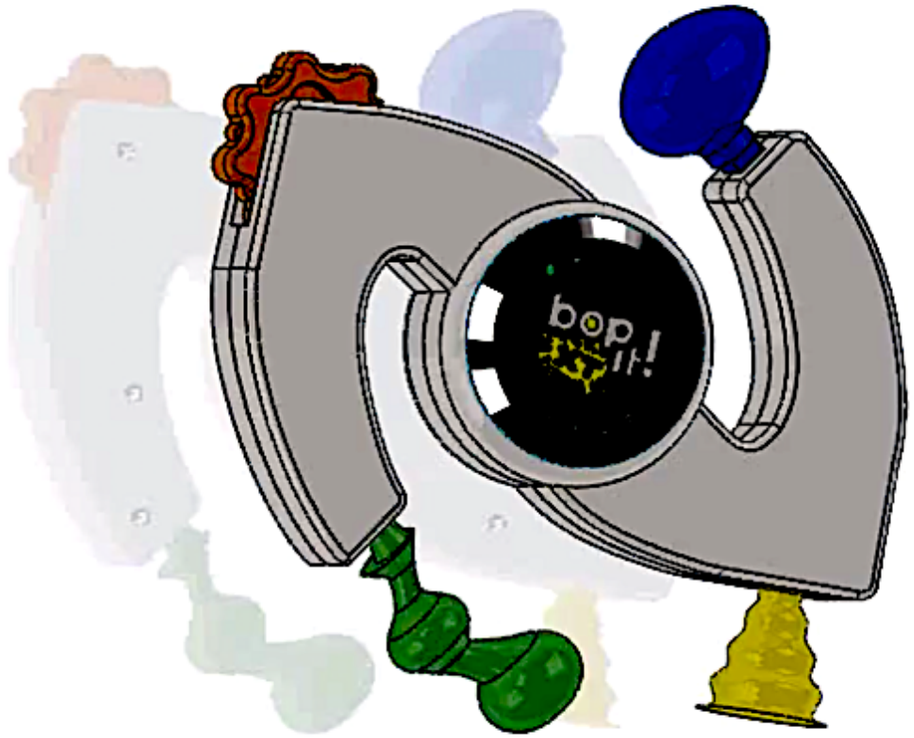
- Generated an exploded-view video in Solidworks to display the dissected model
- Video shows every piece of the model, as well as its correct placement inside the toy

Learn More

- The exploded-view video can be viewed at https://www.youtube.com/watch?v=FpNL1W46u10

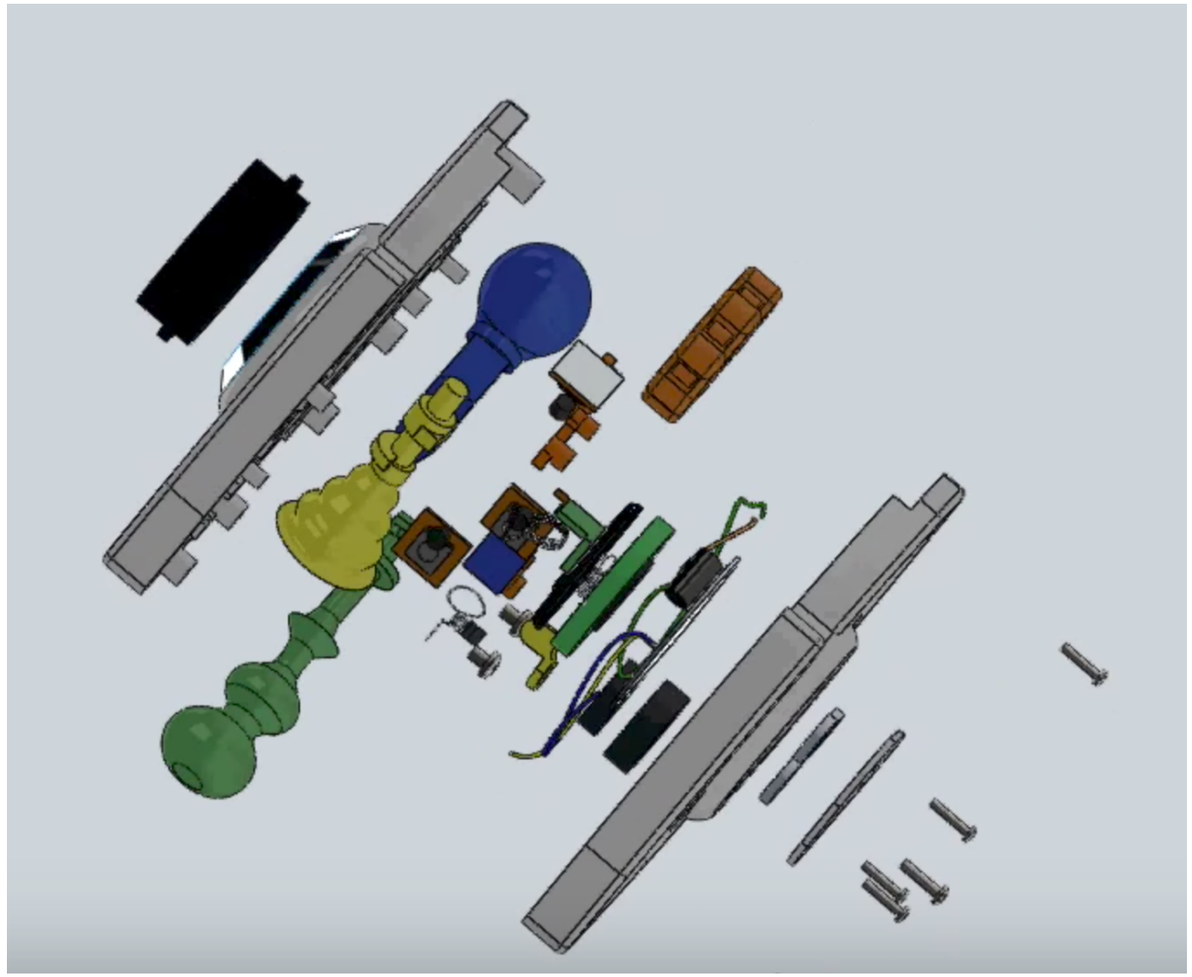


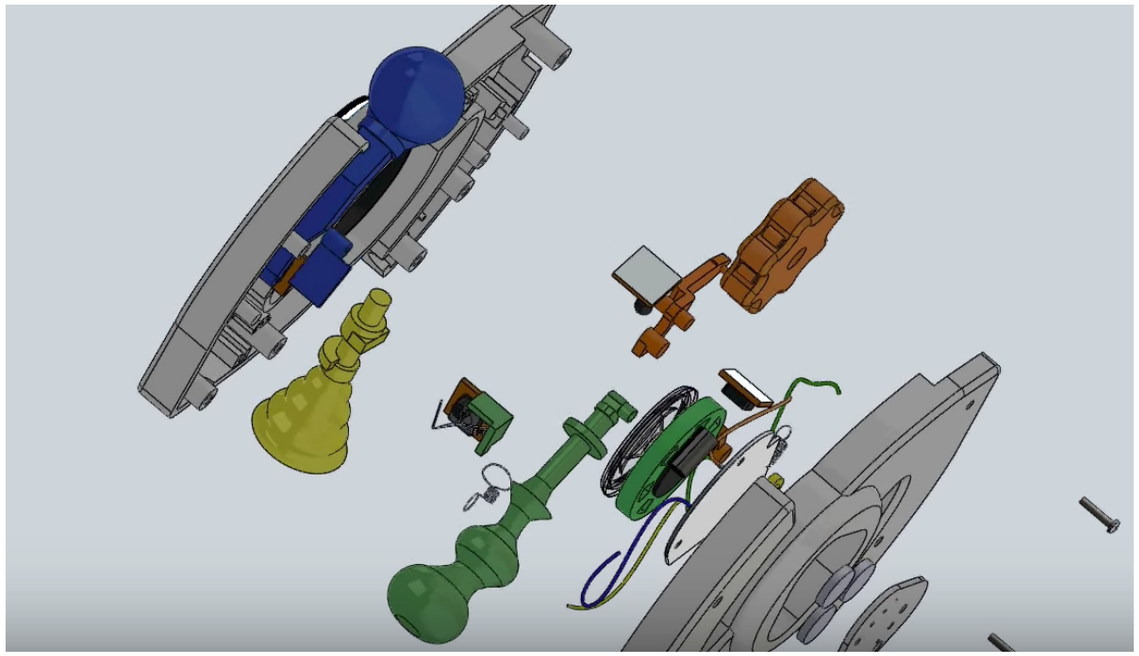
This image, taken as we were deconstructing, displays the intricate inner mechanisms of the toy

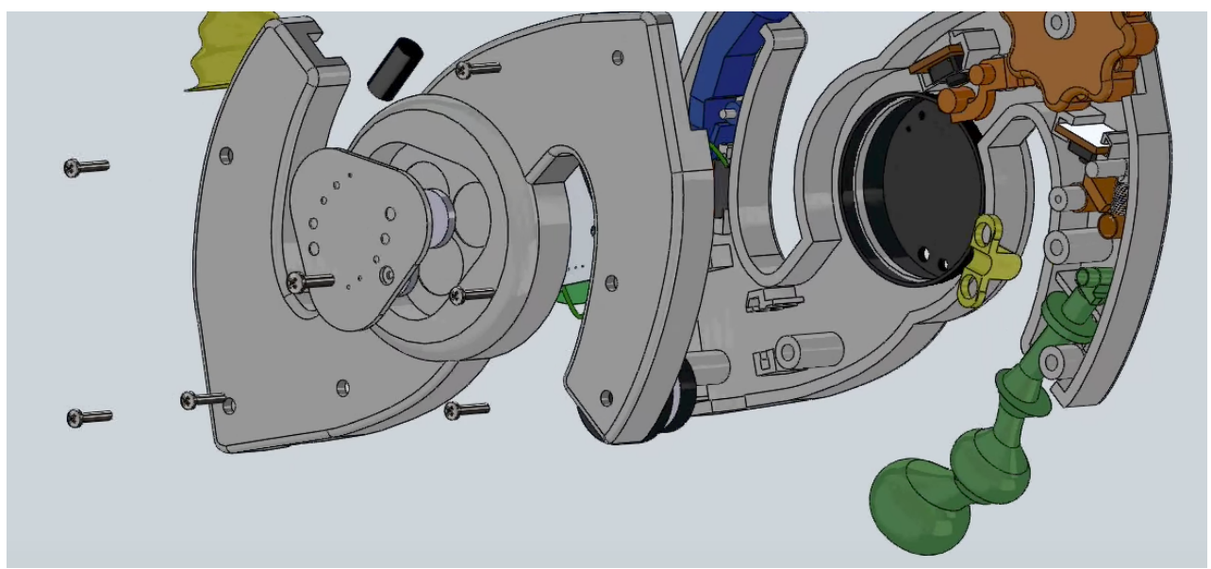


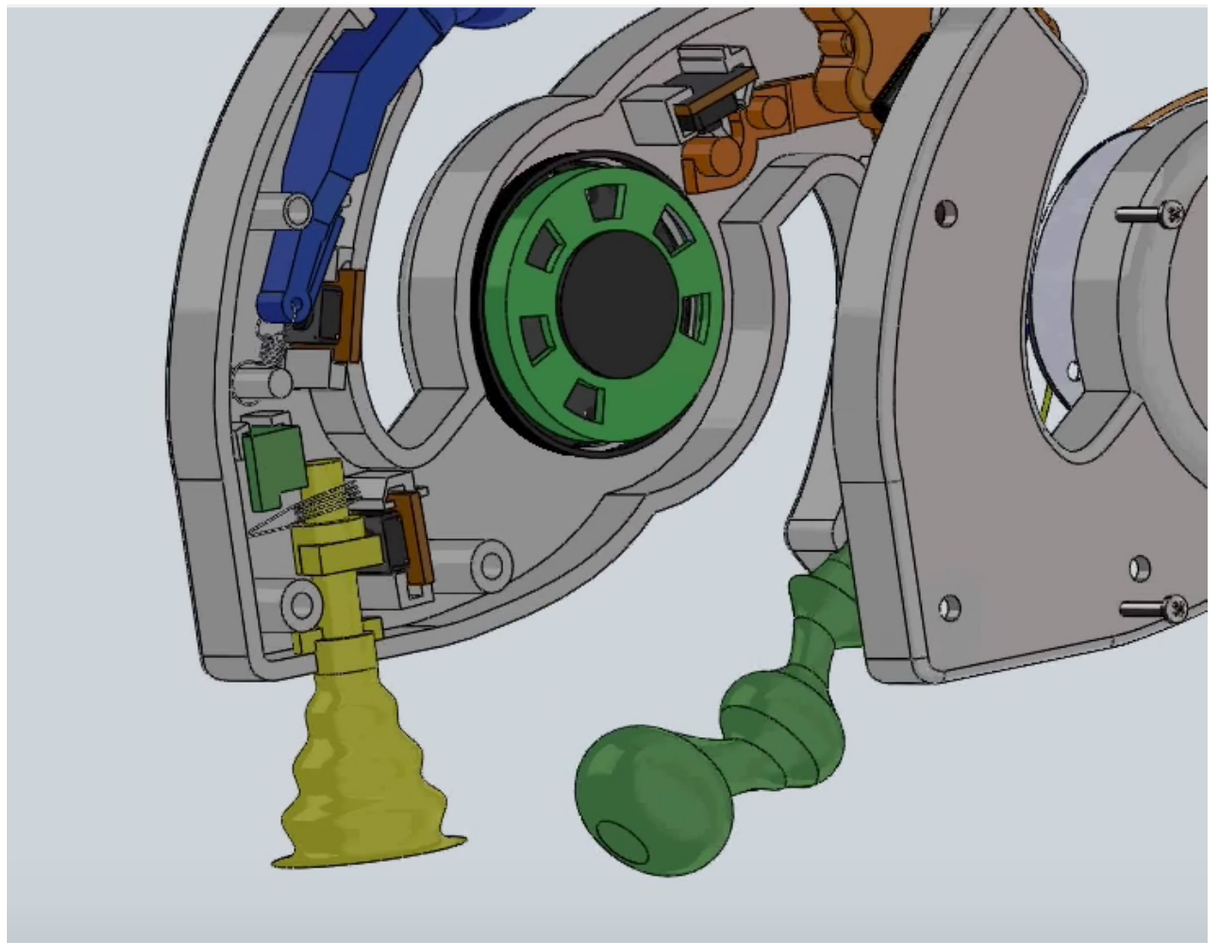
To the right shows the Solidworks rendering of the full Bop-It assembly

Frames from Exploded-View Video









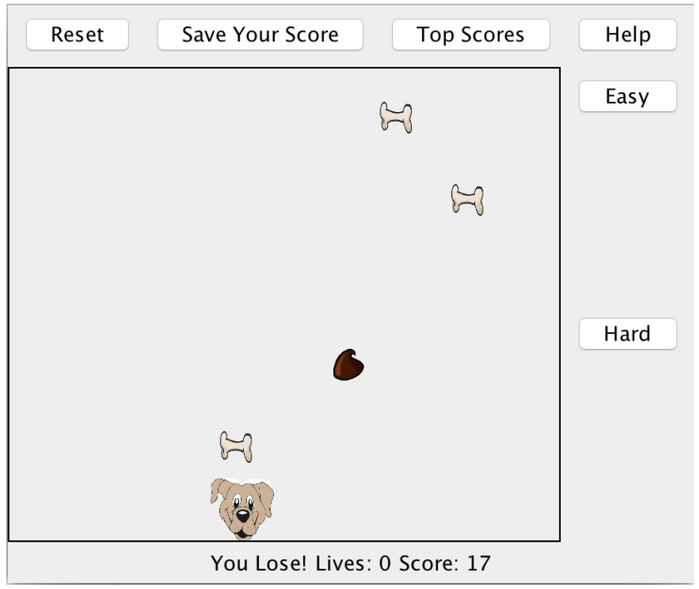
```
public static final int INIT_Y = 260;
public static final int INIT_VEL_X = 0;
public static final int INIT_VEL_Y = 0;

private static BufferedImage img;

// constructor uses all initialized constants, i
public Dog(int courtWidth, int courtHeight) {
    super(INIT_VEL_X, INIT_VEL_Y, INIT_X, INIT_Y
          courtHeight);
try {
    if (img == null) {
        img = ImageIO.read(new File(img_file
    }
```

Project Summary

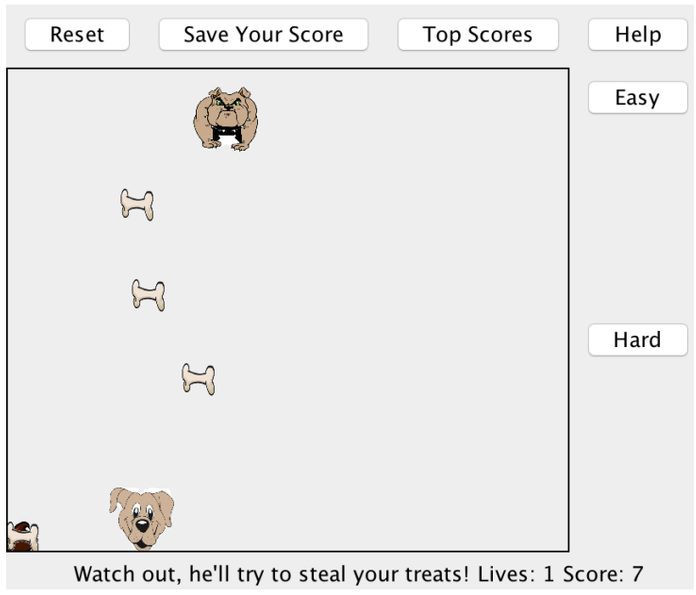
As an intensive independent project in Programming Languages and Techniques I, I used Java to design and program my own computer game from scratch.

Applied Programming Techniques

- **Physics-based Animations:** Used the physics behind N-body simulation to add a 'twist' to the game when the player clicks the "Hard" level. In this situation, there is a second dog on the screen that tries to "steal" thebones. In other words, the falling objects are slightly attracted (using N-body calculations) to the other dog, causing their velocities and the waythey fall to be slightly altered, and the objects themselves harder to catch.
- **I/O File:** Used an I/O file to allow the player to save his/her score and see recent top scores. This involves a method that reads in a high scores file and updates the file to allow the player to add his/her score to the list. There are two buttons: one to add a player's score to the file, another to allow the player to see the list of recent scores.
- **Object-oriented design using inheritance & sub-typing:** All of the classes in my game utilize this core concept. Because each class represents a different component of the game and should operate the way any game object should, all of these classes extend an overarching GameObj class.
- **Modeling state using Collections:** Used this core concept to represent the falling bones and chocolates. I created a LinkedList of randomly generated Bone/Chocolate objects. Using this type of collection allowed me to easily add and remove the objects as they appeared, as the dog was catching them and as they hit the floor. Additionally, I used an iterator with my LinkedList, which allowed me to concurrently remove items from my collection, as they simultaneously hit the dog or the floor.



Snapshot of the end of a game, after losing 3 lives and catching 17 bones.

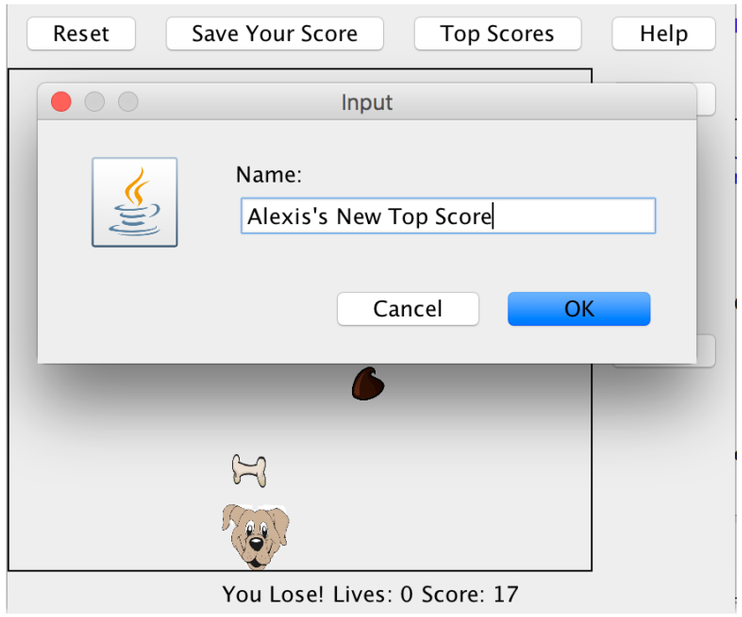


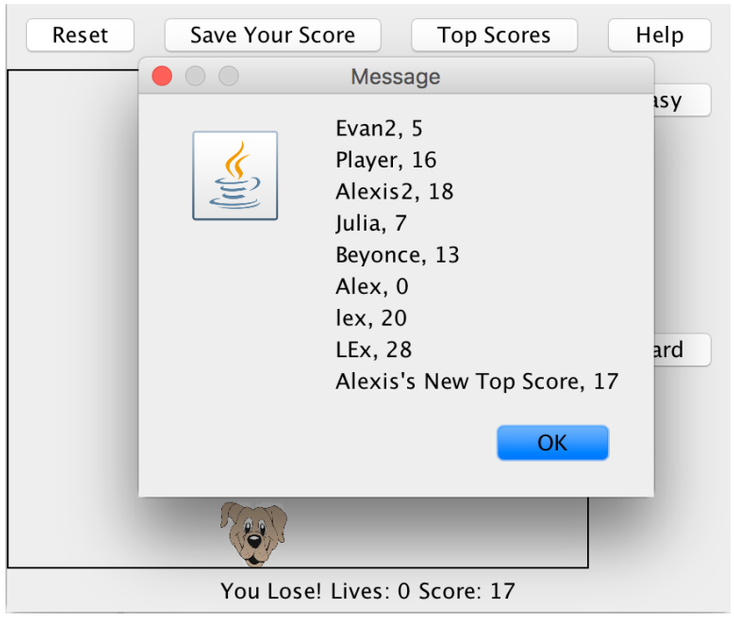
Snapshot mid-game of the "Hard" level, where the dog above alters the placement of the bones, making them harder to catch.

Game Details

- **How it's Played:** There are two different types of objects falling from above that the user (displayed as a dog) has to either collect before they touch the ground (bones), or avoid completely (chocolate). The user can accumulate points by collecting the necessary objects or contrarily, lose lives by failing to avoid or catch specific objects. There are multiple levels, and a leaderboard saves and keeps track of user high scores.

Snapshots and Code Excerpts





The two snapshots above show the saving and displaying of player top scores

```
public void addHighScore() {
    // initialize all variables
    File file = null;
    BufferedReader in = null;
    String newLine = null;
    PrintWriter out = null;
    TreeSet<String> userNames = new TreeSet<String>();
    final JFrame nameFrame = new JFrame();
    // create pop-up window for name entry when method is called
    String nameEntry = JOptionPane.showInputDialog(nameFrame, "Name: ");

    try {
        // read in high scores file
        file = new File("highScores.txt");
        in = new BufferedReader(new FileReader(file));
    }
    catch(FileNotFoundException e) {
        e.printStackTrace();
    }
```

The excerpt above displays some of the code that allows the player to add a new score to the high scores file

The excerpt to the right displays the code controlling the lives, score, and removal of bones and chocolate from the screen

```
void tick() {
    if (playing) {
        // allow dog to move
        dog.move();
        // iterate through collection of bones and chocolates
        Iterator<GameObj> iterate = bonesAndChoc.iterator();
        while(iterate.hasNext()) {
            GameObj obj = iterate.next();
            // if dog catches bone, update score
            // allow bone to disappear after dog catches
            if (obj.getType() == 1) {
                if(obj.intersects(dog)) {
                    score++;
                    scoreLabel.setText("Score: " + score);
                    iterate.remove();
                }
                // if dog doesn't catch bone, decrease lives
                // bone disappears as hits the floor
                else if (obj.hitWall() == Direction.DOWN) {
                    numLives--;
                    livesLabel.setText("Lives: " + numLives);
                    iterate.remove();
                }
            }
            else {
                // if dog catches chocolate, decrease lives
                // allow choc to disappear after dog catches
                if (obj.intersects(dog)) {
                    numLives--;
                    livesLabel.setText("Lives: " + numLives);
                    iterate.remove();
                }
```

```
if (playing) {
    // avoid dividing by zero, if enemy dog overlaps object, don't calc
    if(dog2.pos_x != obj.pos_x && dog2.pos_y != obj.pos_y) {
            // calculate distance between dog and object
            changeInX = dog2.pos_x - obj.pos_x;
            changeInY = dog2.pos_y - obj.pos_y;
            dist = Math.sqrt((changeInX * changeInX) +
                             (changeInY * changeInY));
            // gravitational constant
            double constG = 6.67e-11;
            // calc force, creating approximate masses for dog and obj
            force = ((constG * 1000.0 * 10.0)/(dist * dist));
            // calc x force and y force
            forceX = force * (changeInX / dist);
            forceY = force * (changeInY / dist);
    }
    // alter accel of object based on calc force
    accelX = forceX / (10.0);
    accelY = forceY / (10.0);
    // update velocity and position of object based on calc accel
    obj.v_x += (accelX * 1.0);
    obj.v_y += (accelY * 1.0);
    obj.pos_x += (1.0 * obj.v_x);
    obj.pos_y += (1.0 * obj.v_y);

}
```

The excerpt to the left displays the code controlling the physics-based animations on the “Hard” level

CONTACT

If you have any questions or wish to learn more,
feel free to contact me:

By Phone: 1(631)721-6122

By Email: alexismitchnick@gmail.com

Visit Website: alexismitchnick.com