

Struggling Spirals: A Harmonograph's Challenge

By Spencer Halsey and Samhita Shantharam

How do we create an apparatus that can successfully show the relation between different frequencies with our limited time and budget? This is the question we challenged ourselves with when we picked the Harmonograph project. We weren't sure what it would entail, but we knew that we were in for an exciting adventure.

Harmonographs have an intriguing amount of scientific theory behind their inner workings. The pendulums themselves show a simple harmonic motion, which relates the amplitude, frequency, and phase. When all of these factors align, two or more perpendicular pendulums work together to create Lissajous Figures. A Lissajous Figure is a near-infinite number of curves that are formed by perpendicular sinusoidal, defined as having the form of a sine curve, waves. When one of the waves moves in the x-direction, the other one moves in the y-direction. This is the pendulomic motion that you see from the apparatus. Earlier versions of the harmonograph saw a machine that used sand in place of ink to shape the Lissajous curve, but those designs have evolved to include a table that holds the weight of several pendulums.

The first step we took to building a harmonograph was perfecting a way to support the pendulums. This required a frame that could withstand oscillating weights and be able to support itself. The lightweight design of a table would allow us to transport the entire project wherever it needed to go and give the pendulums plenty of room to swing. We originally designed the table to be three feet tall, but we quickly realized that the pendulums needed an excess of space that three feet simply could not account for. The final product was a four-foot-tall table with parallel supports on all sides to prevent the table from moving. We drilled three circular three-inch holes into the surface of the table which allowed a wide range of movement and space for the pendulums.

The next step was to create the pendulums themselves. We looked over multiple designs on the internet and settled on a hybrid of two different designs. Our pendulums would include a platform at the bottom to hold weights, a wooden bar with screws in the middle to hold the pendulum up, and a socket joint at the top to connect the pendulum to the arms. Our first prototype consisted of a foam block, where we drilled holes for L-brackets. We found that while the foam was a bit problematic, it clearly showed that our ideas would work. From there we went on to prototype a pendulum made out of PVC and a wooden block. This opened the door for us, allowing us to see the full swinging motion of our pendulums, and proving that our design would work. The next step was to start creating our final pendulum designs. Through the process, we realized that the wooden dowels that we were using for our pendulums wouldn't work with the small, rectangular holed weights. This caused us to Dremel down the pendulums, thus letting the weights slide into a snug fit. Along with that, we created a variety of

woodblock mounts to hold two pendulums up. With all of these things in place, we assembled our final pendulums, and to our surprise, they worked. But we ran into a small issue of friction.

The main goal that we sought to achieve in our design process was to minimize the amount of friction caused at multiple points in the machine. The most pressing concerns were the friction between the pen and the paper and the friction between the pendulum and the wooden table. A pen was needed that could draw cleanly given the least amount of pressure because of the platform's constant rotation. To help with keeping weight off of the pen, we decided to hammer divots into metal plates and attach metal screws to the wooden block on the dowel that could fit into these depressions. This allowed the screws to sit in these concavities and have nearly 180 degrees of motion. The dowel that the paper was mounted on used a gimbal joint which allowed it to have full freedom in its response to the oscillation of the other two pendulums.

Our harmonograph worked surprisingly well for the time and experience we had when putting it together. At first, it had a few balancing issues, but with a few minor adjustments, everything worked out. The majority of the project lived up to, if not surpassed our expectations. We both thought that the project would be a complete failure, with us not being able to get anything to work. Instead, we got a sturdy table, which is harder to make than you think, smooth pendulums that conserved energy, and a pen contraption that can cleanly draw the Lissajous Figures. The gratification that we got from seeing that machine function was insurmountable, we felt that we had really accomplished something great. We didn't have a struggle-free process. In fact, the majority of the battle was fine-tuning the machine to get the results that we wanted. After building a table from scratch, trying out several different joints, and calculating where we should place the weights, it seemed like we were on track to have a perfect harmonograph by the end. However, the first test resulted in a scribble that any toddler could have done. The last few days were spent poring over where we missed a liability and what could be done about it and through this tunnel of work we didn't realize how much our design was improved by this. In the end, when we stepped back and braced ourselves for another meaningless scribble, there was a Lissajous figure waiting for us on the table.

Throughout our entire time at Catapult, we learned a few valuable lessons. We learned quickly into the process that there are a lot of variables that can go wrong. Being able to take a step back and look at everything objectively helped us move past the issues that we encountered. It let us refocus on the big picture of the project, learn and have fun. All in all, we had an amazing time at Catapult, and it's impacted our lives forever.

Psgroup 02/03

Sailboats often travel faster than the wind using angles, depending on the boat they can sail up to 3 or 4 times faster than the wind. When sailing at an angle the sail acts as a wing creating an area of high pressure that helps push the boat forward. However when sailing downwind a sail powered vehicle can only travel as fast as the wind. This however does not mean that it is impossible for all wind powered vehicles to travel faster than the wind downwind.

Our group is made up of people from all different places across the country. We have all been brought together at catapult through our love of engineering. All of us have different skills and experiences that help enhance our teams overall capabilities. The challenge we faced was to build a vehicle capable of traveling faster than the wind when moving dead downwind using only wind power.

Youtube personality Veritasium proved that this was possible when he drove a vehicle known as blackbird dead downwind at a speed faster than the wind itself. Blackbird was invented by Rick Cavallaro and is now owned by Neil Cutcliffe. Blackbird uses a fan style

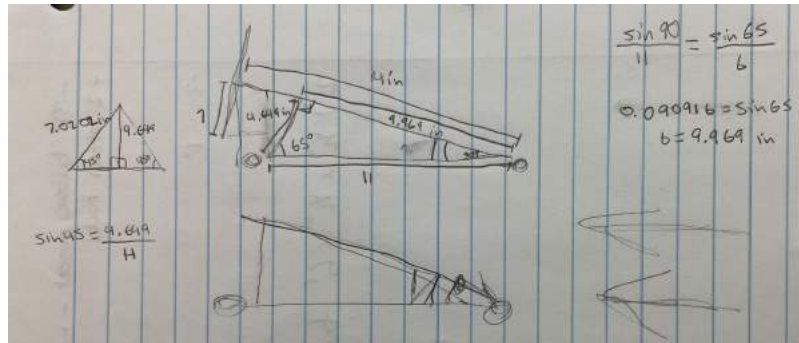


propeller to drive the vehicle through the wind. Cavallaro developed this by looking at sailboats traveling downwind at angles. If you imagine the ocean was a cylinder and the wind was blowing from the top to the bottom of the cylinder then place the boat on that

cylinder traveling downwind at an angle the boat will spiral its way to the bottom. It will do this faster than the wind itself is blowing because it is travelling at an angle. If you then add another boat exactly on the opposite side and send them down at the same time it is essentially a propeller with the sails as the blades. Propellers work by accelerating air past them which drives it forward. The propeller is driven by the tail winds at first then when it passes the speed of the wind it begins using the head wind to continue accelerating past the speed of the wind. This is why the air behind the propeller is moving slower than the other air around the vehicle.

We began our design process by looking at multiple youtube videos on the topic starting with the one done by Veritasium on him testing the Blackbird. These videos helped us understand the physics behind the vehicle and gave us some ideas for how to design our own vehicle. We first drew multiple pictures and did some trigonometry to get an estimate of all the

necessary dimensions in order to keep the propeller off the ground. Once these calculations



were done we finalized our model in Inventor. Our original idea was to build the vehicle out of balsa due to its ease of use and low weight however we ran into issues getting enough wood. This led us to a rethink in our plans, we altered our design to be similar to one we had seen on the internet. This allowed us to quickly 3D print



our parts and put something together to test. We ended up using wooden dowels and metal axles to connect our 3D printed gearbox and a 3D printed stanchion we found on the internet. We were still able to use the same gears we had originally planned on using which was very fortunate. Since our gears had already been printed we spent our time waiting for the other prints testing our gears. We built a wooden square that we could use to mesh the gears and test their tolerances. Most of our build process consisted of

sanding parts down so that they were the right size. We also needed to drill out the wheels so that they could fit on the axles. We put most of the parts together using pressure, ie: bearings and wheels, and used glue for the parts that were a little too loose.

Testing began for us by rolling it down a slight slope to ensure it would travel straight and the propeller spun properly. Initially it pulled to the left a ton however we made some adjustments and were able to correct it, we also found that the prop shaft was rubbing on the frame which stopped it from running smoothly. After mankind made these adjustments we glued everything in place to solidify it and took it to the treadmill to test for real. Our first set of tests showed that our gears worked and the parts could take the stress of high speed running however we couldn't get the vehicle to drive itself forward, it kept sliding back. Our idea for a revision was to shrink the front wheels where the drive was being put on the road. This would increase the fan speed and hopefully provide more thrust. This did provide more thrust however not enough to move the vehicle forward. Unfortunately we were out of time at this point and were unable to achieve our goal.

Although we were not successful in our attempts we had a lot of fun and learned a lot about the engineering process. The one thing we all agreed was the most important was the lack of guidance, now that may seem like a bad thing but it really wasn't. Our faculty mentor was there anytime we wanted help but it was almost completely up to us to design and build the project as well as get the parts. This made it a huge learning opportunity for us, one where we would experience failure if we did not do a proper job of planning and building it. Finally another thing we all loved was the exposure to the wonderful campus resources and shops where we could work on our own. Overall this was a very rewarding experience that we all enjoyed and learned from.

Music Box Project Summary

Most people have probably used a music box at some point in their life. But do you know the process it takes to create one? Commercially produced music boxes involve a complicated process with customized and precise machinery. Many of these tools are not available to us, or most anyone, except for the companies that make them. Much experience is also required when making any musical instrument, as the theory that explains their behavior cannot fully detail all of the intricacies of a design, known only to those who work in such a field as instrument-making. This meant we had to improvise, and create our own plan for designing and building our music box, with none of the typical instruments available to us and without the experience of professionals which has been developed for many years.

Problem Statement

Our goal was to create a music machine that could span at least an octave in under two weeks. We had a total of \$60 to spend on materials for the project. We lacked any prior knowledge of how music boxes were made, or how they worked.

Background Information

Different lengths of metal produce different pitches of sound. Our goal was to create a machine that would harness this property to create music, precisely like a music box. Music boxes are composed of a comb, which is plucked to produce notes, and a cylinder, which has pegs on it that pluck the comb as the cylinder rotates. Some have a wind-up assembly, while other designs are directly driven by the user via a crank. These machines are typically quite precise and expertly made. In our case, we made due with what was available to us, and managed to create a similar machine.

Design Process

We started off trying to determine what sort of music creating machine we wanted. Although concepts such as sound-producing tubes and hammers were considered, we eventually settled on the classic design used in a music box—metal strips plucked by a cylinder. We were initially unsure of how to produce the specific wavelengths of sound we wanted, but our faculty mentor, Dr. Jones, provided us with an equation. This equation told us the fundamental frequency of a strip of metal, provided we knew the length, width, thickness, and properties of the metal in question. After writing a bit of code to automate the process, we determined the lengths needed to play a C major scale, and cut a comb using the water jet. A water jet is a machine used to cut through a variety of materials using a highly pressurized jet of water containing abrasive garnet grit. It allowed us to cut the metal with incredible precision and speed. Using this machine we were able to cut many different prototypes of our comb. In the first set of prototypes we cut 3 different combs, all of varying sizes. We did this in order to test how different sizes would affect the sound produced.

After a bit of experimentation, we came to the conclusion that, while length mattered, the width of the comb had no effect on the quality or pitch of the tone produced. We made two more combs with specific notes to the key signature and accidentals played in the melody of Shostacovich's Waltz No. 2— one with the prongs close together, and one with the prongs

spread farther apart. Using Solidworks, we 3D modeled a cylinder with forty-eight rows of holes, with each hole in a column corresponding to one of the thirteen pitches we would be playing. The reason for there being forty-eight rows of thirteen holes is because we wanted to be able to play the first sixteen measures of the song. After 3D printing the cylinder, both one for the comb with smaller gaps between prongs and one for the larger gaps between prongs, we filled the holes according to the notes in the song, and tested the smaller comb.

Realizing that the strips of metal on the edges of the comb produced incorrect notes, and deciding that the thinner comb sounded better, we redesigned the comb to have extra notes on the ends in order to maintain the correct pitch that our equation showed. We then sandwiched it between two plates to ensure we had the length (and thus the correct wavelength of sound) we wanted. This also helped with stability and the tone quality of the comb. This third design also had tabs on the ends to secure it to the box. We then constructed the box with acoustics in mind, knowing that putting the comb in a corner would amplify the sound. We also cut holes in two pieces of wood to hold up the cylinder as it rotated.

However, when putting the project together, we hit a literal snag—the notes, being stainless steel, did not bend enough for the cylinder's prongs to move past it. The prongs got stuck on the comb, instead of plucking it like we had intended. To resolve this issue, we attached the comb to the box with a layer of foam that allowed it to move slightly up and down. In addition, we reversed the rotation of the cylinder so that it would hit the underside of the comb. After being pushed up by the cylinder, gravity would push the comb down into place for the next note. The only thing left was to create a handle to turn the cylinder with. We had already designed the cylinder with this in mind, and it had a hole on the side to fit some sort of rod in.

Results

The result of our work was a fairly large and oversimplified version of the common music box. Because of the limitations we had with this project, it was, needless to say, not perfect. The tone quality was not the best, and the machine will likely not last as long as a normal music box, since it is easily breakable. However, it does, in fact, function as a music box. And it does exactly what we intend it to do. It plays over an octave and has a distinguishable melody. And, in our opinion, we would say that this means that it was a success.

Reflection

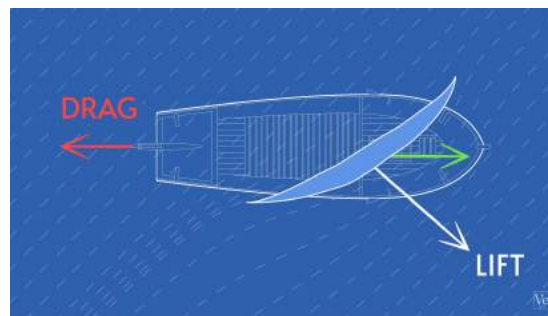
Building a music box from scratch is no easy feat, and the information and experiences we gained along the way were incredibly beneficial for our futures, especially if we are to become engineers. It also helped show us the value of an iterative design process where the faults and issues of a former model can be identified and fixed in later designs. We used this process for our combs, which we printed three times with a new, improved design.

This project was an amazing learning opportunity, and allowed us to see what it really is like to work on a team as people who strive towards a common goal. It is a common occurrence in high school to be put on a team with no drive for what is being accomplished, and to struggle to get others to participate. Not once in working on this project did this happen. Working with others that actually care about and have a passion for what is being done is incredible. We are immeasurably grateful for this experience.

Breaking Physics

Our project was to create a solely wind powered car that was able to travel downwind faster than the wind. This concept has been a big controversy for years, and still is, since it seems impossible to most. It has tricked some of the world's brightest engineers and scientists and, even when the solution is explained, most still have trouble believing it. Recently, a professor at the University of California, Los Angeles, bet \$10,000 that it was impossible. As such, the science youtuber Alexander (Veritasium) Kusenko set out to prove that it worked and succeeded. We wanted to build it, not only to prove to ourselves and others that it worked, but to also discover how it works. Effectively explaining the physics behind how the cart functions is difficult, but we hoped to do just that.

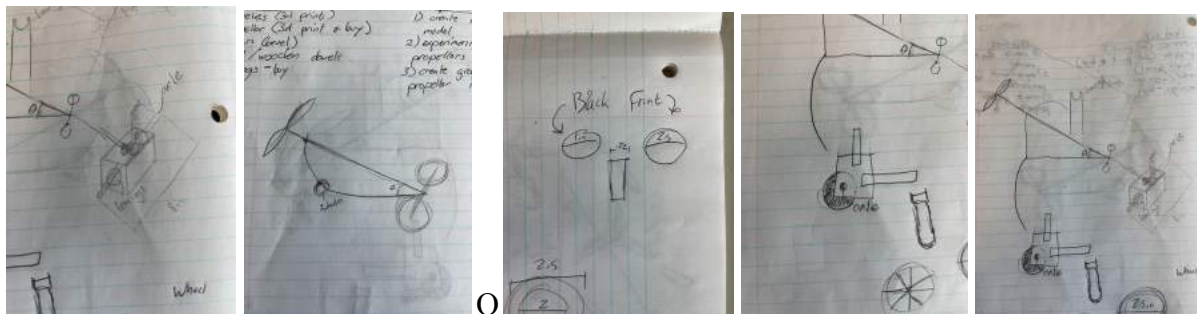
To explain how our car works you must know the Law of Conservation of Energy, Fluid dynamics, and what an Inertial reference frame is. The Law of Conservation of energy states that all energy must be conserved, meaning no energy can be created or destroyed. Many use this law to argue against validity of this concept because the car moving faster than the wind seems to break this law. A big part of why this cart works is due to fluid dynamics and the large propeller at the end of the cart. The idea of traveling faster than the wind while being powered solely by the wind is an old concept used especially in sail boats; however, the boats would travel at an angle to the wind, rather than with it.



Unlike the boat, our cart depends on a propeller to work properly. Propellers work by speeding up the air behind the propeller, lowering its pressure, and letting outside air pressure push on the front of the propeller. This process causes it, and whatever it's attached to, to move. One thing that must be remembered about our cart is that the wheels drive the propeller and not the other way around, but the propeller still plays a large part in making our project function. Since the wheels and propeller are geared together one to one, as the wheels spin faster, so does the propeller, which in turn generates forward thrust, like a fan. The cart is surrounded by wind, as the propeller slices through the air, it makes it so that the wind behind the cart is moving slower than the rest of the wind around the cart. This happens because the propeller takes some of the energy from the coming wind and converts it into kinetic energy, causing the cart to be propelled forward faster than the wind. To create the desired effect, the cart must follow a ratio of propeller pitch to wheel circumference to the drive shaft gearing. Our cart is geared one to one, so the drive shaft gearing can be eliminated from the equation, resulting in a ratio of propeller pitch to wheel circumference. The ratio of $\frac{\text{Propeller Distance}}{\text{Wheel Distance}}$ is known as the vehicle speed ratio, or VSR. For our cart to work successfully, it would have to have a VSR of .7 or less, meaning the wheels would have to be fairly large. The diameter of our wheels was 10 divided by

pi, leaving us with a circumference of 10 inches over a propeller pitch of 4.7 inches. This ratio is crucial for the cart to work, anything over a VSR of .7 won't work, no matter how fast the cart goes. Testing was also very important, since without testing we would not know what to fix in order to make our cart work. We tested our cart on a treadmill, rather than using a gust of wind and still ground. This physically works the same as if the ground were still because of the Inertial Reference Frame. An Inertial Reference Frame is a frame of reference that is not accelerating. One of the most common examples is when someone is on a train moving at a constant velocity. When on the train, the passenger can not tell if the train is moving forwards or if the ground is moving backwards. In physics it doesn't matter if the ground or object is moving since both scenarios are treated the same; this explains why testing our car on a treadmill is the same as testing it with wind. With wind, the air is moving and the floor is still; with the treadmill, the floor is moving but the air is still. With that we have the basis of how our car works.

To start off, we needed to design our car, so we turned to the internet for some inspiration. We watched a few different videos, mostly by Veritasium and Xyla Foxlin. After we had some ideas, and a very vague understanding of how the carts worked, we started to sketch up ideas.

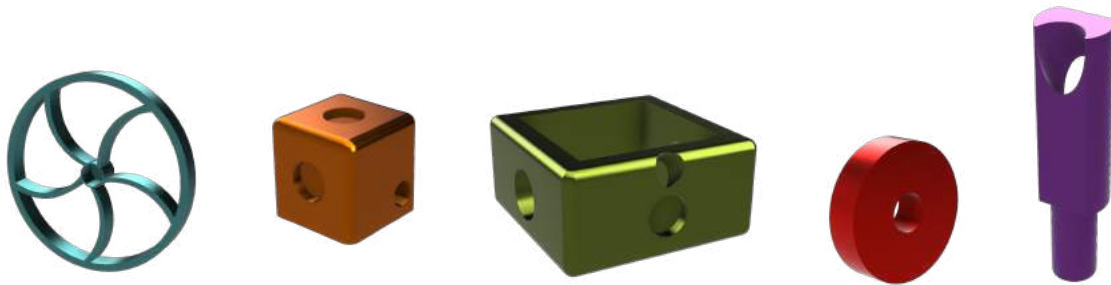


At first we drew up vague ideas, slowly shaping and refining them into our final design.

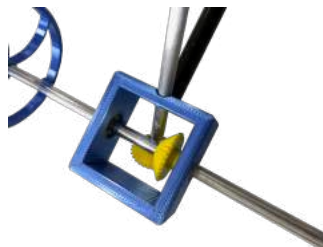


Next we started looking for parts and designing the Computer-Aided Design, CAD, files for parts we wanted to 3D print. We quickly found some aluminum rods in the supply room and

ordered gears, ball bearings, a propeller, and another aluminum shaft for the propeller axel. After that, we used CAD to make the gear box, connector piece, top collar, and wheels.



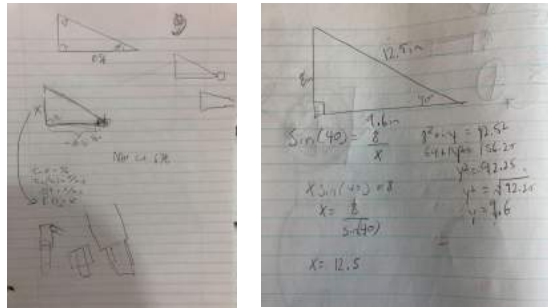
Once we finished designing our pieces, we printed them and slowly tweaked them. One of the most challenging parts of our design was getting the angle for the stick on the gearbox correct.



The angle on the gearbox affected many parts of the car. The first part was the angle of the top collar. The top collar's angle was dependent on the angle of the gearbox; however, since our car was a right triangle once we decided on the angle for the gear box it was easy to find the angle for the top collar.



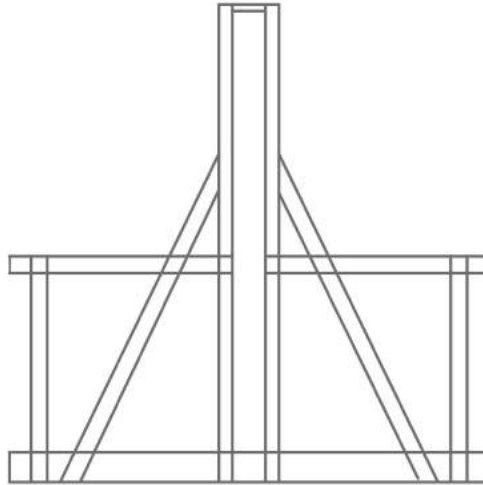
The gearbox's angle also affected what sized gears we needed and how they had to be attached. We realized that because the angle on the gear box was too steep, and our original gears were too small they could not mesh together. In order to solve this problem we used 3-d printed gears that were bigger. The angles we chose for our car also determined how long some of the pieces of the body would be. We first chose how long our car would be, after that we did some quick trigonometry to find what the height of the car should be and what the length of the propellers axel should be, or its hypotenuse.



We adjusted and tweaked our design quite a bit before assembling the entire thing, but once it was assembled, we were happy with its design.

Once we finished assembling the cart, now dubbed ‘Choppy’, we made our way to the SRC building to begin testing. To test we placed the cart on a treadmill, held the front of it, making sure to keep arms away from the spinning propeller, and bumped the treadmill speed to 12 miles per hour. In theory, if the propeller was attached correctly and the ratio was proper, the cart should have started driving forward on the treadmill, but ours did not. We tested 4 times the first day and, by the end, our gears had ground themselves down to where they didn’t work anymore. The cart clearly wasn’t going faster than the wind, nor was it close yet. We decided to re-print two more gears, glue on the good wheels, and re-adjust the propeller to make sure it was facing the correct way. After we completed our fixes, we headed back to the treadmill. On our way to the SRC we ran an involuntary durability test by dropping the cart, and one of our teammates, down the stairs and discovered a weak point in our design. The place where the gearbox and bottom part of the frame meet was not secure, since the hole where they connected was too shallow the bar would pop out. After the durability test, the cart had trouble staying in one piece, often breaking apart at the newly discovered weak point. We tested on the treadmill 3 more times, the frame breaking apart twice. The cart had a better time during the second set of tests, getting closer to going on it’s own, rather than easily flying back off the treadmill. Finally, we rebuilt the cart one last time and worked on figuring out where we went wrong.

During testing we learned something extremely important: that our cart didn’t work. Choppy had failed, but our question was why? Our best theory is that our cart was geared improperly due to the angle of the propeller axle. Our cart is in the shape of a right triangle with a 40 and 50 degree angle, but this makes it so that the propeller axle comes in at too steep an angle. We initially wanted to use a set of hard, plastic gears, but they ended up being too small and were unable to mesh due to the angle; gear one narrowly missed gear 2. Had we done a 20 70 90 triangle rather than a 50 40 90, it might have been a more gentle slope, making the gears mesh properly instead of grinding each other to dust. We also would have changed our gearbox to have a deeper hole for the frame to go into. Technically, we already fixed this, but it still would have to have been changed on the CAD file. By the end of build time, we had many pieces that didn’t fit with each other, so if we were to do this again, we would try to better flesh out our plan and always ask about 3D printer tolerance. We learned a lot while working on this project, enough to perhaps build a working cart in the future.



Introduction

We are team War Machine, and we were tasked with building a trebuchet that would be able to launch a tennis ball a far distance. We decided to build a variation of the trebuchet called a floating arm trebuchet as we believed this would give our build greater range and energy.

Problem Statement

An army of evil boilermakers is attempting to invade Rose-Hulman! The only way they can be stopped is by aerial bombardment by tennis balls launched from a far distance. We need to build a device to allow us to prevent the evil plague from spreading.

Background Information

The main physical principle behind the operation of a trebuchet is conservation of energy. Assuming the forces of friction and air resistance are negligible, the energy of the trebuchet system should remain constant. This means that the initial potential energy stored in the trebuchet arm as a result of its height will be transformed into kinetic energy as the gravity does work on the trebuchet and the arm drops. The principle of conservation of energy also motivated our choice of a floating arm trebuchet, as the structure of a floating arm trebuchet allows the arm to drop further, giving the projectile more speed than in a normal trebuchet.

Design Process

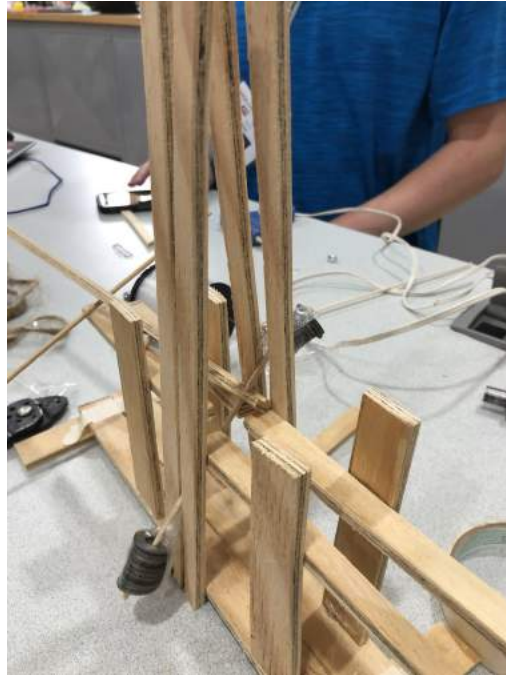
We began by researching different types of trebuchets and quickly chose the floating arm trebuchet for our design. Although floating arm trebuchets are more difficult

Team War Machine

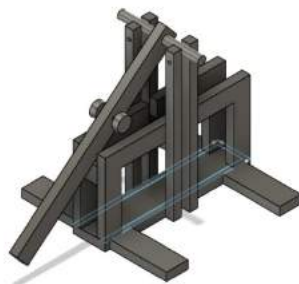
Group #06

Amaan Ahmed, Conner Kalvar, Jake Buchsbaum

to build, the ability of the arm to move and drop allows the trebuchet to transfer more potential energy into motional energy, causing the ball to launch at a faster speed. After consulting some designs of trebuchets online, we constructed an initial prototype of the trebuchet design. However, due to a lack of materials and precise measurements this prototype.



We then began to build the second form of our trebuchet using CAD modeling. After modeling, we used the 3D printer to create a small model to test the design with. From this model, we were able to adjust some dimensions, like the distance between the towers, for our final build.



We then constructed the final model of the trebuchet. Once it was completed, we began to test multiple different variables, like the pin angle and rope length, in order to find the best possible range.



Results

We found the optimal rope length to be 3.5 feet and the optimal pin angle to be 75 degrees. With these values, the trebuchet launched the tennis ball a distance of 167 feet. The trebuchet was not able to meet our full expectations, but the reasons for this are beyond our control. Environmental factors like friction and air resistance mean that energy can never truly be conserved, so our trebuchet will never be able to meet its full potential. However, despite these setbacks, we were still able to successfully construct the trebuchet and launch the ball a good distance.

Reflection

This project gave us a greater understanding and appreciation of the process of designing and constructing a solution to a problem. We also learned about how to adapt to and work around limitations on our solution, whether they be man made (price and size constraints) or natural (working around friction and air resistance).

Team Name: Bees

Introduction:

Our idea for the game is about a bee shooting stingers to defend flowers from attacking insects. Our plan is to have an easy, medium, and hard level as well as an endless mode that, if we can implement it, will get progressively harder as the game goes on. The flowers will be on the left side of the screen and the insects will approach from the right. There will be multiple types of enemies with varying speeds, hit points, and flight patterns. When an enemy reaches the base, a flower will die. When there are no remaining flowers, the players lose the game.

User and software interaction:

- Start menu with option to select 1 or 2 players. Then option to select easy, medium, hard, and endless modes
- 2 players, each player controls a bee
 - Player one: wasd to move and space to shoot
 - Player two: arrow keys to move and ctrl
 - or each player connects a controller

Problem Statement:

The world does not currently have enough fun arcade style bee games. Not a day goes by where I don't hear a colleague, a friend, or even a family member go on about how much they've been longing for such a game. Society has been in ruin, politics are ruling our society, the minds of our children are rotting away day by day! But the brilliant minds of Kyle Griffin and Sam Doerfler have the solution, *Bee Game!*

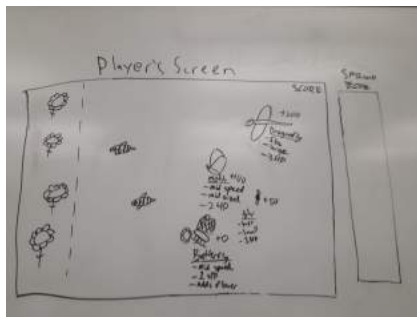
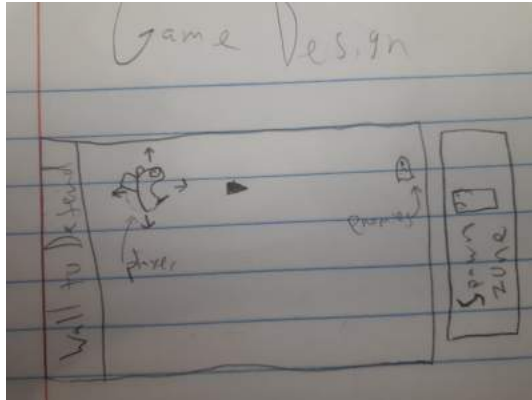
Background Information:

Picture this in your mind; a comforting warm shower, water streaming down like god himself sent it down from the heavens above, ideas forming, neurons sparking, brain cells dying, and **BANG POOF** it struck me, *Bee Game!*. - Co-Lead Game Code Director Samuel Doerfler

Design Process:

In the beginning there was nothing. Then the Co-Lead Game Code Director called main(), and *Bee Game!* was born. Flashback to 17 years ago, two of our generation's greatest minds were brought into this world, their fates sealed with a prophecy, to make a bee game unlike any ever seen before, one that straight up swags. Our heros were presented with 2 options to fulfil said forsaken prophecy: One where the bee shoots enemies ruthlessly assaulting their garden from the right, almost as if Satan himself sent these insects after the garden. Maybe yet a gaming experience where

unrelenting waves of hurt are bombarding the bee's garden in the form of unseemly insects from all sides, with the heroic bee defending their flowers with not only their body, but with their mind and soul as well. The prophesied heroes, brave, charming, handsome, yet most importantly humble, chose the former option, if only so they could feel as if they themselves are fighting in the unending war with Satan and his minions, and more importantly because it would be easier to code.



Results:

The game performs without any stuttering, there are no known bugs, and the hitboxes extend a little bit past the enemies so people don't feel cheated by a slight miss. The graphics look nice, the score functions perfectly, the game over screen works as intended. All around it met expectations, we knew that we would only make difficulties with extra time, but we agreed that it would be too complicated to complete while balancing the difficulties. Overall we feel like we met most of our expectations gameplay-wise, and exceeded our expectations for the visual aesthetic.

Reflection:

Throughout this project, some of the key lessons learned were to communicate well, stay organized, and to put in the work needed if you want good results. Without good communication and organization, code can easily become a mess. Additionally, the coding process can be quite frustrating, especially when trying to find small errors that mess up everything so being thorough can help prevent this.

Bethesda Jr.

Ross and Jake

Our team came together and created Bethesda Jr. to design a game.

OUR GAME

We wanted to make our own spin on Pac man with free movement control and multiple floor states. Every floor would have special hidden collectibles, keys, and items to help play. To advance to the next floor, the player is required to get that floor's key and get to the stairs. All the while, enemies are roaming the halls.

DESIGN PROCESS

We broke the game down into separate instances and Ideas to find out what we need to do in order to build our game, we first worked on figuring out the movement logic for the player and then the enemy. Once we coded that in, we built the maze and began the long process of building a collision system for the player. We settled on a system that revolved around checking for a specific color. That worked but with some caveats, the biggest one being that you could still clip into the walls, but not through them. We figured out that it did this because it was only checking the color based off the top left corner of the player sprite. In order to fix that, we tried adding parameters to the check wall function that would create a box around the sprite so it would not go into walls. This failed and the collision did not work correctly at all. Eventually, we figured out that each direction needed specific parameters for collision to work correctly. Doing that creates a box around the player sprite that is checked for collision instead of just a single point.

Animating the player sprite was simple, we set a variable to count to a certain point before it resets and every time it reaches that point it switches the image. This works well but it showed us about bringing outside variables into a class.

Our initial process for designing the enemy AI was to simply have the enemy check for a wall and, once it hits it, the enemy turns in a random direction. This worked, but the game was very easy, even when we added many enemies. The problem was that they could be evaded very easily since they only moved in random directions. We decided to add a way for the AI to see where the player was located and move towards the player. We made the AI check a radius around itself for the player and to follow the player if they are detected. However, this disregards the walls of the maze. In order to fix that issue, we had the AI check for the distance to the player and the distance to the wall in front of the enemy, If the wall was closer than the player, then the enemy would not be able to see the player. This increases the difficulty of the game but one way we could have increased the difficulty further, we could have made the AI

store the first couple of directions the player goes in and used those to determine where the AI turns until it runs out of stored directions.

The next thing we worked on were coins. At this point, we began to run low on time so we had to cut some of our ideas in order to make a playable game. The coins were all placed manually which does subtract from the game's ability to be replayed due to each round feeling very similar, but it does save a lot of time compared to designing a system to randomly place coins throughout the maze. In addition to adding coins, we added a coin counter so that the player can see how many coins they have left to collect to win.

RESULTS

Our result was a game where the player runs around the map collecting coins while enemies search for the player and chase after them. This was much more simplistic than our initial ideas, but we have a completed, playable game. We put a lot of effort into getting this game made, but our lack of experience coding made things much more difficult. Things like the player collision slowed us down heavily due to their complexity. However, we still feel like we did very well, we were able to shift our expectations and work to a new, more realistic goal.

REFLECTION

We learned a lot about programming throughout this project. We learned how to make more efficient systems and how to work on code together. In the end, we were not able to complete every feature we wanted to put into our game. This feels like an insight into the programming field. They have strict deadlines with lots of things to build. Sure, if our group wanted to put everything we wanted to into our game, we probably could, but we would have had to work on our game in our rooms for hours more each day and our result would probably have been very buggy due to focusing on the features instead of building something more polished. It would have been very taxing on us to try that and it would be hard to have pride in our game because it would feel less complete. Its difficult to imagine that many software developers get put under situations like this so often. It does give insight on what the world of software is like, but also showed us how to persevere in times of challenge and keep pushing to make the best product we can.



Introduction

Shooty Game is a top-down survival game. This game includes many features such as upgrades (Health, Speed, Damage, Ammo, and Accuracy), multiple types of enemies (Normal, Tank, and Speed), and visual/sound effects. These added features allow the player to have a unique experience each time the game is played.

Problem Statement

For the project, our task was each individual group must create a unique game using prior or learned knowledge within the period of 11 days. These groups would be made within your respective class and you will discuss the idea and objective of the game.

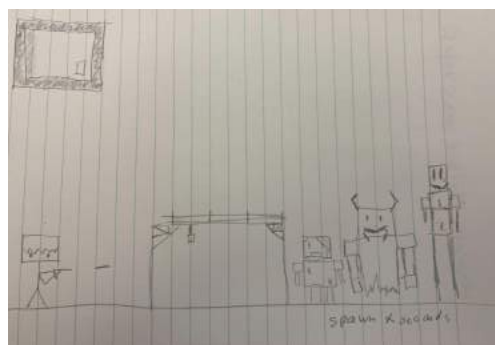
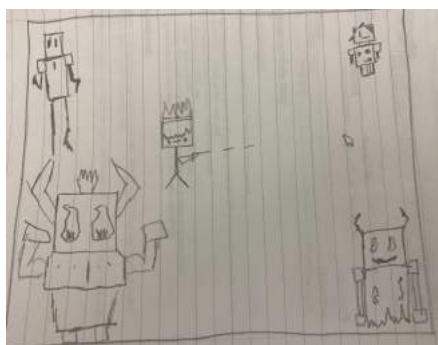
Background Information

Some of the prior knowledge we had while developing our project was the opportunity to take a coding class at Oliver's high school in which he learned some Python. Other knowledge we had was taking other engineering classes. Hunter had taken multiple engineering classes which included programming robots with C++. And although it was a different coding format, it helped me understand coding a bit better.

Design Process

In our project, we discussed multiple ideas for the project. One of the ideas was a side view survival game. Enemies would swarm from the two ends of the map attempting to attack you. Some of these enemies can include Normal, Speed, and Tank. Our second idea was much like the first idea, but instead of a side-view, it would be a top-down view. We decided to choose the top-down view because instead of enemies spawning from 2 points, they spawn around the map. This allows the player to have more freedom of movement and use more of the upgrades given on the upgrade screen.

While developing our project, we used Python, PixelArt, and other websites to make our game. Throughout the development of the game, we would test run the game. During the testing phase, we repeatedly tested our game. We made notes on what worked, what didn't, and what could be changed. While testing our game, we noticed that balancing was a major issue; some of the upgrades cost too much, some of the enemies needed to be more powerful, the amount of money you got for killing an enemy needed to be changed, and the game didn't look aesthetically pleasing.





Results

After we had completed the fundamentals of the project, being enemies, player movement, character design, map, upgrades, and shooting, we decided to add other game features with the extra time. These features allowed the player to have a better experience or even help them in the game. These features are sound effects and visual effects. These changes notified the player if they needed to reload, buy upgrades, or if they're shooting.

A minor problem we had acquired was the mp3 files not playing. On the Microsoft Surface Pro, the mp3 files would load, but on the HP EliteBook 2740p, it would not load. But, this is most likely due to a hardware issue rather than a coding issue. To test if it was a hardware issue, we downloaded other mp3 files and tested if they would load, and they did not.

Reflection

After completing our project, we had developed more knowledge in Python. Using the tutorials allowed us to gain or refresh our knowledge on Python. Next time when we remake the game, we would add some more game features such as a leaderboard, a boss, optimizing the code, and removing unnecessary coding. This would make the code less complicated and therefore allow it to run faster. Also, adding a boss would allow a better player experience and force them to buy more upgrades.



The goal of our project was to learn to code, and use this new knowledge to code a game. This was made more difficult for our team, since we had no prior coding experience, and had to create something from nothing. Not only that, but that something should be a game that excites users to play, and entertains them for a bit. To put it simply, you might as well have told a mechanical/design engineer to start writing software. What an oddly specific example. We had to learn fast and validate ideas faster, even though this process ended up posing a problem for us later.

Problems create difficulty, and we were having a difficult time. So, it seems like the problem at hand was created by ourselves. How do we create any kind of game if we have no knowledge of making games? Without doing anything other than signing up for the class, we had already laid ourselves a burden. Others in our class might say that they built up something positive to “ascend”, but for us, it was more like we were creating some positive to get out of a whole we were already in.

The first thing we had to do was learn and memorize. It was tough to do so quickly in such a short amount of time, but the videos we received sped up the process greatly. I, at least, came out more knowledgeable on coding than I ever had been, but every line of code I wrote would create an error of some kind. Not the best place to start, but a place to start nonetheless. My partner seemed like he came out a little more capable though, so I was sure we could create at least something. Honestly, “something” would have been enough for me.

The first thing we thought about were retro games. Fortunately for us, they usually have a simple premise, contain easy-to-emulate graphics, and are still loads of fun. Of these retro games, we chose something like *Galaga*, a space game in which you shoot up at enemies and dodge their attacks. However, we still wanted to make something different. Copying would feel unoriginal, and probably leave us with the feeling that we didn’t make it. So, we thought of combining two games in order to make it our own. This second game was *Clash Royale*, in which you place cards to deploy troops to attack your opponent. These cards cost certain amounts of

a resource, which can only be recovered over a set period of time, so managing it is key. These two games together, along with a small sprinkle of originality, would create *Stellar Strife*. In this game, you would battle a local opponent using cards with different values, which spawn certain types of fighter ships. Each fighter would have different abilities, like shielding, range, and speed. The goal would be to push your opponent back and attack their main ship. Once destroyed, the destroyer would win. Although this idea was a great idea, with possibilities rapidly being added, we forgot something in the midst of our theoretical creation. We knew nothing; zilch, zero. We had done a small amount of practice with basic coding principles, and our completion date was cruelly close. The second major problem we had was also of our own making. It took all the way until Monday to get our heads out of the clouds.

We started to realize that not only should we not add too many variables, but our ideas shouldn't be variables either. So instead of thinking up something so lofty and idealistic to us that our minds were left to keep wandering, we decided to bring back the simple, down to earth elements of actual retro games. A local co-op would still provide the fun lost to our nonexistent expertise, but we would also keep the main ships, in order to hold onto the protect-and-attack feeling that we wanted. We would also keep the top down gameplay, space theme, and retro vibe. We had still combined *Galaga* and *Clash Royale*, but we had separated their individual aspects more, making it seem both more original and feasible. It would be, and is our end product.

In the end, we weren't able to get to everything we wanted to. However, some things fit into place well, and we were able to create a fun game. Explosions look vibrant, and it is somewhat intense! Honestly, I think it's better than we could've imagined it at the beginning of this week.

We entered this class to learn, and learn we did. Not only about Python, but also pressure of time constraints, and the difficulty of asserting your own ideas into other people. Lots of annoying things happened over such a short span of time, but that just made the good things more satisfying. Although the process was very rushed, I think we created a cool, competitive game that leaves players a little less

bored, and with little to know starting knowledge. We were writing code till the last second

Yash Agarwal and William Coulter

Team Name: Enterprise

Game Name: Stellar Strife

Ultimate PACMAN

This is PAC MAN, a very famous game that is many people's childhood favorite, but with a twist. Everything is random. The classic PAC MAN maze you know is no more every time you boot up a level the maze is procedurally generated, that means that when you play a level it will be completely unique no one will have played it before and no one will play it ever again. This will give you a different challenge each time but still keeping the original feel of PAC MAN. We coded in the language PYTHON, a multi purpose language that is probably the most widely used language today. The heart of our game is the A* (A star) algorithm, this is a maze generation and solving program that we used to generate a grid of numbers that would become our maze. This is an example of the type of grid that would generate:

Generate

```
1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 0 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1
```

Prune

```
1 1 1 1 1 1 1 1 1
1 3 0 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 0 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 2 1
1 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1
```

Solve

```
1 1 1 1 1 1 1 1 1
1 3 0 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 0 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 2 1
1 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1
```

We then used a program that took this grid and turned it into our maze in the form of an image. It did this by taking the numbers in the grid and assigning them to colored squares 1s = white and were walls and borders theis defined where PAC MAN could gor or not. 0s= black they defined the passage ways that PAC MAN could be in. 2s= green this is the end point of the maze. 3s = red this is the start location of PAC MAN. These squares were placed in their corresponding position and then we were given an image that we could display in the game. After that we created our PAC MAN, we implemented movement that is similar to the original game and the classic PAC MAN animation. We then made it so that PAC MAN could not go outside of the borders we defined. After that we created the start screen that allowed you to choose between one of 3 predefined difficulties or endless mode that gave you infinite levels that slowly increased in difficulty. We then created the win conditions we added the dots that pac man eats and made the game end once you had eaten them all. After that we had to do bug fixes and work out some of the errors we had and the game was complete.

In my opinion the game came out so much better than I could have ever expected. We were able to implement more features than we originally thought we were going to and the mazes were much better and more complex than we thought we were going to get.

Git Logs

—| A* Generator |—

- [c245321](#) - (3 days ago) Fixed name of python file - Cole Fleming (HEAD -> master, origin/master, origin/HEAD)
- [f3de55e](#) - (3 days ago) Added Custom Color Map along with image generation functions and troubleshooting. - Cole Fleming
- [fa02b11](#) - (3 days ago) Added Level Image Generation - Cole Fleming
- [f5a35ff](#) - (4 days ago) Added Maze Generation and Made Sure ending position was at the coordinate that matches zero, that is the furthest away from the starting point - Cole Fleming
- [52bd068](#) - (4 days ago) Implemented Alternate Maze Algorithms, and added system argument "-r", which will change the maze generation to my original algorithm, a random generation equation. - Cole Fleming
- [8cf0c16](#) - (4 days ago) Added Pattern Recognition (For Traps) - Cole Fleming
- [716d1ed](#) - (5 days ago) Added Executables and Added More Options to System Arguments - Cole Fleming
- [eb38def](#) - (5 days ago) Updated main.py to fix compatibility issues with system arguments - Cole Fleming
- [2e368c4](#) - (5 days ago) Update main.py - Cole Fleming
- [0cb5f6c](#) - (6 days ago) Added First Step of Bloating Algorithm. Need to add infinite recursion next, although its a bit late for me - Cole Fleming
- [864c1be](#) - (6 days ago) Added Grid Creation (Randomized, No Grouping) and also added the ability to grab the data value of nearby squares relative to a coordinate - Cole Fleming
- [6b5401b](#) - (6 days ago) Initial commit - Cole Fleming

—| PacMan - (Front End) |—

- [7e2c369](#) - (4 days ago) Beautified The Code with Gray, then YAPF, then ran it through regex to convert to tabs. - Cole Fleming (HEAD -> master, origin/master, origin/HEAD)
- [4b98c25](#) - (4 days ago) Added Variations based on Direction for Image Switching - Cole Fleming
- [04623a4](#) - (5 days ago) Fixed Animation and Added Modulus Variable for Speed - Cole Fleming
- [b23343a](#) - (5 days ago) Added Animation (No Direction Changing Yet) - Cole Fleming
- [1675ed0](#) - (5 days ago) Beautified Code - Cole Fleming
- [38ab27a](#) - (5 days ago) Fixed Issues with Old Code - Cole Fleming
- [93c06b5](#) - (5 days ago) Initial commit - Cole Fleming

—| Final (My Version) |—

- [70b7e3e](#) - (22 hours ago) Added Spec File - Cole Fleming
- [6cee77a](#) - (22 hours ago) Added Fonts and Fixed File Path issues with cross device compatibility - Cole Fleming
- [ba7ce9b](#) - (2 days ago) Updated Poetry Config - Cole Fleming
- [ecabbb1](#) - (2 days ago) Beautified Again - Cole Fleming
- [ca37312](#) - (2 days ago) Initial commit - Cole Fleming

—| Final (Combined) |—

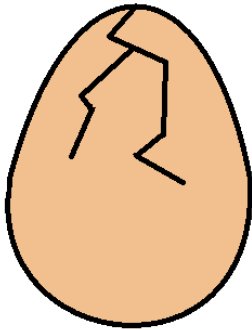
- [45158a2](#) - (2 hours ago) finalized project - Phin Elliot (HEAD -> main, origin/main, origin/HEAD)
- [7e29047](#) - (29 hours ago) Update for Git Diff - Phin Elliot
- [a5cf93f](#) - (29 hours ago) decision making and collisons - Phin Elliot
- [37a72de](#) - (3 days ago) decision making and collision - Phin Elliot
- [6e640fd](#) - (3 days ago) start of maze genoration - Phin Elliot
- [9ecfeed](#) - (3 days ago) start of maze genoration - Phin Elliot
- [54576ab](#) - (4 days ago) start of maze genoration - Phin Elliot
- [8c68c4b](#) - (4 days ago) Commit Changes (Test for Diff) - Phin Elliot
- [70336c9](#) - (4 days ago) collison and border - Phin Elliot
- [248098b](#) - (4 days ago) git push --set-upstream origin main - Phin Elliot

Team Number: 12

Team Name: Cracked Games

Team Members: Madison Masterson, Bonde Borca, Deniz Demir

Logo:



Cracked Games was tasked with creating a game using python in ten days. Our team decided to create a Tamagotchi and Duck Life inspired game that has the pet care aspect of Tamagotchi and the training and racing aspect of Duck Life. We named this game EGGOLUTION. We decided to name the game EGGOLUTION because in the game you raise a pet that starts as an egg and evolves. Once we had an idea and a name we started brainstorming ideas for the game. Once we found the features we wanted for the game we created flows. Flows show how the program responds to the player's interactions. Flows are important to map out because they help figure out what needs to be programmed into the game and how to go about programming. Once we had a good outline of what we wanted for EGGOLUTION we started programming.

For the features we wanted there had to be a lot of stats (statistics) for the program to keep track of. We wanted to have a wellbeing stat that went down over time as the pet gets hungry, dirty, or overweight. Feeding, cleaning, and exercising your pet makes the wellbeing stat go back up. If wellbeing reaches zero the pet dies and the player has to start over with a new pet. We also wanted a strength and agility stat for the mini game and races. When you play the mini game the strength stat goes up and how much it goes up depends on what score you get in the mini game. The strength stat decides how fast your pet can accelerate and your pet's top speed in races. To help the player keep track of stats we wanted a stats screen to show the pet's wellbeing, strength, and agility. These stats are the base of EGGOLUTION.

For our game we wanted to have two mini games, a training game and racing game, which meant we were really making three games so while working on the main UI (user interface) another part of the team was also working on the training mini game then the racing mini game in order to make the deadline. The training game ups strength which makes your pet faster in the racing mini game so in order to progress in the racing mini game you have to play the training mini game. This is where the Duck Life inspiration comes in because we wanted a core gameplay loop that involves training your pet to become the best racer.

We also wanted the pet care game play aspect of Tamagotchi. We decided on implementing exercise, cleaning, and feeding into the game for the pet care game play loop. You can check on how your pet is doing in the stats screen and take care of it accordingly. Another feature we implemented was evolution which comes from both Tamagotchi and Duck Life. The pet starts as an egg and evolves into a child after 60 seconds then after feeding the pet ten times it evolves into a teen. To reach the final stage of evolution, adult, you have to win the race mini game ten times. All of these features are what makes up our game.

Over the course of development things went very well. We started with implementing all the stats that had to be kept track of and the other part of the team started making the first mini game. After the variables were set we started creating backgrounds and icons for the game and implementing them. Once the button icons were implemented we added click detection so when you clicked on the buttons the program would perform the corresponding task. Around this time the first mini game was finished so it was added into the main game and the second mini game started development. As the first mini game finished we were also finishing up button functions so the user could feed the pet, clean the pet, and check the stats screen. As everything started to come together we started running into bugs which are expected in a games development. Part of our team dealt with the bugs while the rest finished the second mini game. Once all the pieces were complete we put them all together to finish up our game.

When game development finished we then moved on to game testing to catch any last minute bugs and patch them out. We only found a few bugs that were easy to patch out. We also decided to add some last minute features to the game because we had the time. Thus the game was finally done and we were ready to present it.

Later on after the game was done we reflected on how game development went and there are a few things we could have done better but over all the project went very well. We met all the deadlines, we had a functional game, and most of the original features we wanted in the game made it in the final game. On the few things we could have done better we could have planned out coding better. Now any coder developing just about anything can always say they could have planned coding out better but in our case some bugs and merge issues could have been avoided if we had planned better and saved us time. The lack of planning also got one major feature cut. Originally we wanted three mini games with the third mini game being called holes where the pet runs down a series of platforms by falling through holes which would raise the agility stat. Anything having to do with the third mini game had to be cut because the mini game itself was cut. Better planning could have given us the time to make the third mini game, but luckily the third mini game was the only major feature cut. When it came to the mini games they were made separately then added into the main game. The way the mini games were made unfortunately made them a bit hard to implement into the main game so better communication between the parts of the teams working on different parts of the project would have helped prevent these implementation issues.

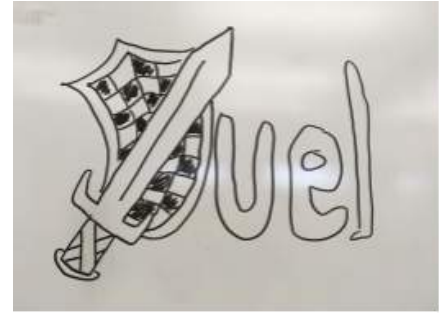
Overall Cracked Games is very happy with how EGGOLUTION turned out and development went very well considering we only had ten days to make a functioning and appealing game. We hope people will enjoy EEGOLUTION as much as we do.

Duel

Team Name: Team Name (Yes, seriously)

Group #: 13

Members: Chancellor Teibel and
Elias Iceman



Pictured: Duel logo

What is Duel?

Duel is a game about a clash between two knights, Sir-Pokey Knight III and his arch nemesis, the ruinous Face-Shield. An action centric battle to the death, Duel simulates the strategic chops needed to come out on top. Programming this game was anything but easy. Our team consisted of two programmers, none of which were very experienced with the coding language used in this project: Python. With about two weeks to develop and execute our concept, we were tasked with learning Python and its visual counterpart: Pygame and creating a fun experience for players.



Pictured: Strategic Chops

Background

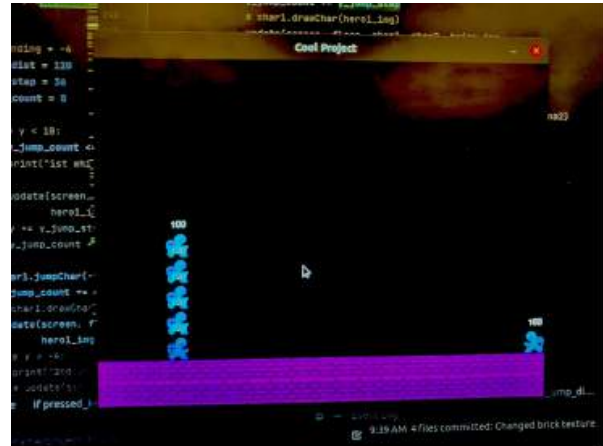
The design process for this project was very frustrating at a glance. We were working from a distinct disadvantage. Our tech issues (due to no fault of Rose) persisted for two days that we could have used to become further acquainted with Python and Pygame. Luckily, we were able to keep pace with surrounding teams by conceptualizing the game via a whiteboard and fresh dry erase markers. We had a distinct, yet simple concept that fit within Catapult's timeframe. Very few of our goals were left behind as we continued with development.

Building Duel

As mentioned, we were provided with tutorials on how to use the features of Python and Pygame to create a functional program. A list of each project and the technical skills it lended to the coders is provided below. Additionally, Rose provided us with video resources through Panopto: a secure video sharing platform.

1. IntroToPython - Provided basic information on how to open, format, and label the window that our game would be hosted in.
2. MovingSmile - Assessed how to create and move objects on a screen (window) and read keyboard inputs from a user. This was useful in programming every control input in Duel. The game makes use of keys as its method of controlling player characters. Later in the project, this program became critical in understanding how to correctly implement movement keys and other actions.
3. DogBark - Teaches implementation of custom sounds and images, while also making use of mouse clicks to activate certain events in the program. While our game did not implement sound, it did use imported sprites drawn in the pixel art app piskel. Being able to import and paste these sprites into our program contributed to the aesthetic and readability of the game environment, allowing the user to understand its concept and mechanics in a short amount of time. Mouse clicks were utilized in the game's menu, enabling the user to press the "Play" button to access the real meat of the game. Game menus provide a level of formality to an experience, and acquaint the user with its premise before the game even starts.
4. ClickInTheCircle - Very similar to DogBark, except it checks for user inputs within a certain space on the screen. This allowed for the implementation of a button on the menu screen, rather than being able to click any area to start the game.
5. Raindrops - Introduced collision detection between different objects on the screen (and gave quite a headache to the team). Collision is a very important part of action games as a whole; it makes sure that the user cannot phase through objects, and allows for swords to actually hit the player rather than going right through them. Collision ended up being one of the major issues in getting the game to run, and was the determinant in whether or not this game would have to limit its scope to something restrictive and unentertaining. Much like MovingSmile, Raindrops became an important reference for collision detection in Duel.

These lessons helped contribute to a relatively smooth programming experience (with a few bugs along the way). With about a week left of time to complete our program, we made quick work with a lot of the essential features. Once those were completed we made a menu for the game and added a pinch of depth to a relatively simple game using features such as a stamina bar and the dodge feature, both of which encourage a careful, yet aggressive playstyle that pressures your opponent to make decisions.



Pictured: Oops! All glitches!

Mistakes and Moving Forward

1. As our team continued with the project, our code eventually became messier and messier. Later we realized how simple some of our obtuse solutions actually were, but without the proper experience it would be very difficult to clean them up without affecting other portions of the program.
2. Animations for extending the player's sword would add a further level of depth to the game and contribute to its overall aesthetic.
3. A better understanding of Python's timer features would help with implementing smoother gameplay overall. The time crunch (get it?) didn't help much.

Conclusion

It's our hope that we achieved our goal of creating a fun, competitive dueling simulator with the resources available to us. What matters more than our own satisfaction is the satisfaction of the player in interacting with our game and its systems. Mission accomplished? That's up to you.

Duel

Team Name: Team Name (Yes, seriously)

Group #: 13

What is Duel?

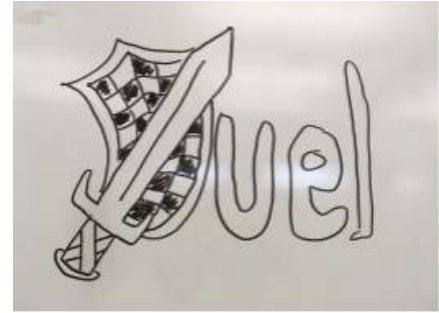
Duel is a game about a clash between two knights, Sir-Pokey Knight III and his arch nemesis, the ruinous Face-Shield. An action centric battle to the death, Duel simulates the strategic chops needed to come out on top.

Programming this game was anything but easy. Our team consisted



Pictured: Strategic Chops

of two programmers, none of which were very experienced with the coding language used in this project: Python. With about two weeks to develop and execute our concept, we were tasked with learning Python and its visual counterpart: Pygame and creating a fun experience for players.



Pictured: Duel logo

Background

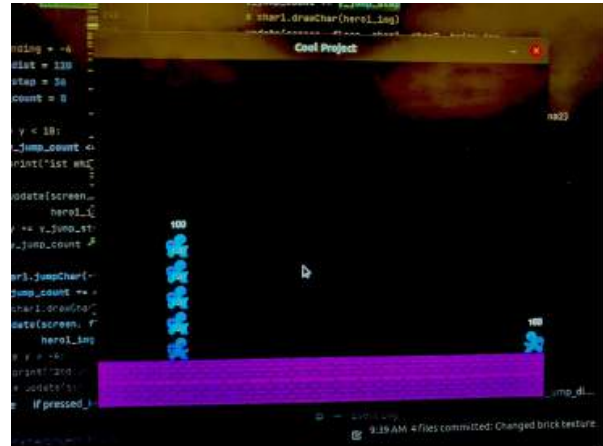
The design process for this project was very frustrating at a glance. We were working from a distinct disadvantage. Our tech issues (due to no fault of Rose) persisted for two days that we could have used to become further acquainted with Python and Pygame. Luckily, we were able to keep pace with surrounding teams by conceptualizing the game via a whiteboard and fresh dry erase markers. We had a distinct, yet simple concept that fit within Catapult's timeframe. Very few of our goals were left behind as we continued with development.

Building Duel

As mentioned, we were provided with tutorials on how to use the features of Python and Pygame to create a functional program. A list of each project and the technical skills it lended to the coders is provided below. Additionally, Rose provided us with video resources through Panopto: a secure video sharing platform.

1. IntroToPython - Provided basic information on how to open, format, and label the window that our game would be hosted in.
2. MovingSmile - Assessed how to create and move objects on a screen (window) and read keyboard inputs from a user. This was useful in programming every control input in Duel. The game makes use of keys as its method of controlling player characters. Later in the project, this program became critical in understanding how to correctly implement movement keys and other actions.
3. DogBark - Teaches implementation of custom sounds and images, while also making use of mouse clicks to activate certain events in the program. While our game did not implement sound, it did use imported sprites drawn in the pixel art app piskel. Being able to import and paste these sprites into our program contributed to the aesthetic and readability of the game environment, allowing the user to understand its concept and mechanics in a short amount of time. Mouse clicks were utilized in the game's menu, enabling the user to press the "Play" button to access the real meat of the game. Game menus provide a level of formality to an experience, and acquaint the user with its premise before the game even starts.
4. ClickInTheCircle - Very similar to DogBark, except it checks for user inputs within a certain space on the screen. This allowed for the implementation of a button on the menu screen, rather than being able to click any area to start the game.
5. Raindrops - Introduced collision detection between different objects on the screen (and gave quite a headache to the team). Collision is a very important part of action games as a whole; it makes sure that the user cannot phase through objects, and allows for swords to actually hit the player rather than going right through them. Collision ended up being one of the major issues in getting the game to run, and was the determinant in whether or not this game would have to limit its scope to something restrictive and unentertaining. Much like MovingSmile, Raindrops became an important reference for collision detection in Duel.

These lessons helped contribute to a relatively smooth programming experience (with a few bugs along the way). With about a week left of time to complete our program, we made quick work with a lot of the essential features. Once those were completed we made a menu for the game and added a pinch of depth to a relatively simple game using features such as a stamina bar and the dodge feature, both of which encourage a careful, yet aggressive playstyle that pressures your opponent to make decisions.



Pictured: Oops! All glitches!

Mistakes and Moving Forward

1. As our team continued with the project, our code eventually became messier and messier. Later we realized how simple some of our obtuse solutions actually were, but without the proper experience it would be very difficult to clean them up without affecting other portions of the program.
2. Animations for extending the player's sword would add a further level of depth to the game and contribute to its overall aesthetic.
3. A better understanding of Python's timer features would help with implementing smoother gameplay overall. The time crunch (get it?) didn't help much.

Conclusion

It's our hope that we achieved our goal of creating a fun, competitive dueling simulator with the resources available to us. What matters more than our own satisfaction is the satisfaction of the player in interacting with our game and its systems. Mission accomplished? That's up to you.

Pokémon Makers



In the video gaming world, there are many different types of games to play: from RPGs (Role-Playing Games) to sports games to shooting games, the sky's the limit when it comes to finding a game for a person's interest. But narrowing it down, the Pokémon games have been very popular since 1996 and haven't lost much in popularity. However, some people play these games and only want the fight scenes and not play the whole story in the game. This can become an issue because if one doesn't want to follow the game's story, then they might as well drop the game completely. After all, Pokémon is one of those games where you can't exactly go off from the storyline too much. Because every Pokémon game follows a very similar storyline, everyone that does want to play the game but only wants to fight will lose interest in the whole game and then there wouldn't be many other people that would play the whole game. To fix this issue of not having enough battle scenes compared to the amount of story in Pokémon, we created a game that's inspired by the Pokémon game that only has battles, and we call it 'Python Pokémon Battle'. Two players can play this game and the players interact with 'Python Pokémon Battle' by choosing a Pokémon and battling against each other until one player's Pokémon doesn't have any health left; in which they would be considered defeated and the last one standing would win.

The main inspiration for 'Python Pokémon Battle' is from, obviously, the Pokémon games, but we wanted to add a twist to it and make it different and maybe even better. Fighting games always keep people on their toes with the rapid action and vivid attacks being used on the battlefield. Even if the original Pokémon games do have these features in their battles, they still have the storyline outside of the battles which can make the player focus on the story more often in the actual battle. Whenever I, Melea Stafford, would play Pokémon, I always loved going to battle and defeating either a wild encounter or a trainer's Pokémon. However, when the game would go more into the story or even into a side quest, I would immediately get bored and try to go against the story to keep battling because, in my opinion, it's the best part of the game. Since I've always had an undercurrent feeling about the Pokémon games with it not having it being all battles, it was very motivating for me to create a game that only had to do with battles and not with a story so that you can truly focus on strategy and competition.

Before starting the project or even coding at all, Evan and I thought about what exactly should happen when the players interact with the game. Before we concluded about having a fighting game, we originally thought about having the game being an RPG, just like the original Pokémon games. But, with the limited knowledge, skill, and time we had, the idea wouldn't be executed with quality and the game would most likely not work well. So, as an educated compromise, we decided to make 'Python Pokémon Battle' as a fighting game for the sake of time, rather than making a game with fighting scenes and a story.

After concluding the style of the game, the next thing that had to be done was to figure out how to write basic code. The language used in 'Python Pokémon Battle' is Python, a language that's starting to become more and more popular since its release in 1991. We started on our tutorials on the first workday, which was on July 22, and worked on those tutorials until July 26 before getting into 'Python Pokémon Battle'. The tutorials that needed to be completed were assigned in this order: 'Moving Smile', 'DogBark', 'Click in the Circle', and 'Raindrops'. 'Moving Smile' was used to learn how to draw shapes, write code based on what key the user would press, and how to implement moving something on the screen. For 'DogBark', we learned how to scale, which is essentially loading an image, writing code for clicking on the mouse, and loading and playing sound. 'Click in the Circle' is used to learn how to use the distance function to detect where the mouse clicked and using coordinates to check where some logic is true or false; in this case, the logic would be whether the mouse clicked in the circle or not. Lastly, with

‘Raindrops’, we learned classes for objects and stop-motion animation. Classes in Python are used to define the object(s) to sort what code is for which element; and stop-motion was used to change between two images of a person either holding up an umbrella or just standing. The objective was to keep the person from getting wet from the rain by sensing where the rain is, and when the rain would be in its vicinity, the person would put up an umbrella until the rain disappeared.

Once the tutorials were finished and the talking for what needed to be done for ‘Python Pokémon Battle’, we decided to split up the work where Evan would start the code and I would look for the Pokémon images and sound files so that it can be deposited into the repository for Evan to use. It took me about one project session to find a little over a hundred Pokémon images that could be downloadable as a PNG (Portable Graphics Format) since it’s the image format that’s acceptable to use in Python. In the next session, I found all of the sounds much quicker than finding the images and I was able to deposit all of the images and sounds into the repository on the same day. The game’s coding had to be split into two modules: Pokémon selection and the actual battle. Since the players need to choose a Pokémon before getting into battle, a catalog had to be made, which in this case is called a Pokédex, so that the players can search for the specific Pokémon they want. When testing the different modules to see if the code is functional, there have been many bumps in the road. But when it comes to coding, it’s considered normal to not have your software work on the first try. Some bugs had to be fixed throughout the game so that the game wouldn’t malfunction while being used in play.

After debugging a lot and doing some final tests, ‘Python Pokémon Battle’ works very smoothly and rather quickly as well. The rounds go rather quickly because there aren’t any battle, damage, or Pokémon animations; the game’s also quick because of how much damage a Pokémon can do and there’s no dodging or missing attacks like in the original Pokémon games. ‘Python Pokémon Battle’ has gone through a lot of revisions, from the title to the gaming style, it’s gone through many phases and it still somehow works well. But with expectations, it exceeded mine at least because I thought that we would have to go with something completely different and ditch the whole topic, which we essentially did for the concept of having an RPG with the battles. But I think that it was a better thing to do than keep going with it and keep struggling. We adapted well and trekked through rough times, but it was definitely worth it and if I had more time here at Operation Catapult, I would add so much more to the game to make it even more spectacular.

One very important lesson that we learned which can be applied to any type of coding is making sure to debug absolutely everything. Even if the bug may be something simple and avoidable, eventually that same bug will become a larger issue in the future which may become the downfall of the whole project. It’s very good to get rid of any bug as quickly as possible as you’re coding to make sure that there aren’t any major issues when the project is just about to be used in the public. The last lesson, but still very important, is knowing how to code what you want and not going out on a limb. When brainstorming about what you want to do for your project, you must know what you want to do and how you’re coding it so that you’re not going out on a limb when coding. When going on a limb, you won’t have a definite idea of what’s needed to make what you originally wanted because by going with something vague and not specific and understandable, the program could result in a worse state than it was before adding the new code.

Tim Parisi, Clint Robinson, Takia McMorris, Nico Neuweg
Dr. Hossein Alisafaei
Rose-Hulman Operation Catapult
7-30-21

Operation Catapult Project: Laser Selfie

Introduction: Almost everyone has seen a laser at some point in their lives, and lasers normally come with poor connotations. When people think of lasers, people often think of how they could be used for destruction. A high-powered beam of light that can blind individuals is commonly thought of as a weapon that could only be used to cause harm. But what if lasers could be used to create? The long range and accuracy of lasers makes them a tool with massive potential. Our project challenged us to test the potential of lasers and see how accurate they could be.

Problem Statement: Can an image as complex as a photo of a person's face be displayed by a single laser?

Background Information: There are two main concepts that need to be understood in order to understand the project: Fourier Transformation and diffraction. Fourier Transformation is a process by which images can be converted into a series of waves. If light shines through these waves, the shape of the beam of light will display an image. Diffraction is the phenomenon of how light travels through narrow spaces. If light shines through a narrow gap onto a wall, the light will not only shine the shape of the narrow gap as one would expect. Instead, the light that shines on the wall will be repeated both horizontally and vertically, with reflections farther from the center having decreasing levels of intensity. Finally, it is important to know that light travels in waves, each with a random direction.

Design Process: While our final design was simple, multiple iterations of the design led us to our final product. Our first attempt was the simplest, with the sole intention of confirming that the process would be successful. The design began with the laser, which shone light through a single device: the spatial light modulator (SLM). This device works like an external display, but displays graphs of the Fourier Transformations of images instead of the images themselves. This allowed the laser beam to be converted into an image with an external computer. A computer would take the images of our faces, convert them into black and white photos, simplify the details, and send them to the SLM to create Fourier Transformations of the images. While this design was able to display images, it was not successful enough to call a complete project. The laser beam was too unrefined to display any level of detail, and diffraction meant that the image was repeated in a mosaic pattern across the wall. Following this attempt, we focused on refining the image to display detail. This led to the addition of four pieces to the design, placed in between the laser and the SLM. The first piece, the pinhole, was introduced to change the shape of the laser beam to be completely circular. Secondly, the laser beam went through a pair of lenses. These expanded the size of the beam to cover the surface of the SLM. Finally, the laser beam travelled through a polarizer. The polarizer provided order to the random directions in which laser beams travelled, and only allowed light waves travelling vertically to pass through. After multiple iterations of pinholes and lenses of varying size, this design was able to create an exceptionally clean laser beam. However, the laser beam passed through the SLM unaffected. Eventually, we determined that the two lenses were keeping the image from being projected onto a surface. This realization led us to our final design, with three pieces used after the laser. First,

the laser beam passed through a polarizer to provide order and consistency. Second, the beam travelled through the SLM, to display an image. Finally, the laser beam passed through a pinhole, which removed the mosaic effect caused by diffraction. The SLM was then connected to a computer to display the images.

Results: Overall, our design was a success. When provided with images, our laser was able to display images with a satisfactory level of accuracy. Simple shapes, such as circles and squares, were able to be displayed with almost perfect accuracy. Moreover, the laser was able to display complex images successfully, such as the Rose-Hulman 'R.' Finally, when it came to faces, the images did have to be heavily simplified to be displayed. However, when the images were displayed, the laser was able to define facial features such as the eyes, nose, and mouth. With more time, we may have been able to successfully implement the two lenses, and enlarge the image as well as increase quality. However, with almost no knowledge of optical engineering prior to the project, we were able to successfully learn enough optical engineering to display a complex image with a laser.

Reflection: One of the lessons we learned the most from this project was the lesson of Occam's Razor, which states that the simplest solution is often the best solution. After our first iteration, we were so focused on complicating the design to create a clean laser, that we did not notice how close we were to success. Simplifying the design meant that there were fewer pieces that could fail, and that the laser would at the very least display a poor-quality image. Secondly, the project emphasized the importance of teamwork between our group. Doing the entire project as an individual would have been near impossible, especially in our two-week timespan. Listening to other's ideas allowed us to brainstorm more quickly and efficiently, and allowed for clean and fast solutions to problems. Moreover, respect between team members ensured no one would be too afraid to contribute to the project. The project also showed us how connected the sciences are to each other. While our project was mostly focused on optical engineering, many other elements of engineering were required to complete the project. Mechanical engineering was required in assembling the laser setup, computer science was required to create the image, and engineering design was used to create and troubleshoot our project. The project reminded us of the importance of knowing what will be beneficial for your goals, and eliminating things that will only bring you down. When initially given the project, we were given an Arduino board to help with our project. However, we eventually realized that the Arduino board would be of no use to us, and we abandoned it to focus on the laser setup. The project taught us to trust mentors and people who are willing to help you out. While our group knew next to nothing about optical engineering, many people, and resources, such as Dr. Alisafae, graduate students, and textbooks had the information we need to turn our project into a success. Relying on their advice and information, we were able to understand all the information we needed to complete the project, and learn how to troubleshoot and design the laser on our own. Finally, the project showed us how quickly technology is advancing, and the potential of lasers today. If a group of high school students can display an image of a face in two weeks, what could hired professionals do over the span of months? The capability and accuracy of lasers could make them highly valuable in making jobs more efficient. The accuracy of lasers could make them useful in sectors such as the medical field and the manufacturing field.

Introduction:

As a kid did you ever wish you go on a spy mission? Going through lasers, trying not to get caught? Well, with our laser maze, we can make that wish come true. Our laser maze that we have created is going to be sent to the Children's Museum here in Terre Haute. Now, any kid who goes to the museum can live out their spy dreams by going through our maze.

Problem Statement:

Our goal was to create a safe laser maze while working with the constrictions we faced. These constrictions included space, supplies, and time. But the biggest constriction we faced as a team was lack of experience. No one on the team had a strong knowledge of coding, an essential part of the maze. Due to the safety requirements, we needed to be able to create code that would turn the lasers off as soon as a beam was broken so that there would be no danger of being blinded by the lasers. When it came to constructing the maze and everything in it, we faced several miscommunications with the size of the maze, the materials we had, and the estimated wait times to acquire more materials. We originally planned for the maze to be larger than what it is now, but due to some miscommunication about the shelving units, we had to scale back on the size of the maze. This ended up working out though because we didn't need to buy as many supplies, and it made completing the maze easier with the time we had.

Background Information:

In order to establish basic levels of safety for the participants, automatically turning off the lasers when the beams are crossed was crucial. As such, we developed code after many iterations to relate the change in voltage, running back through a loop out of the Arduino, resulting from the effects of a photoresistor or photosensor. These resistors change their voltage throughput depending upon the amount of light that contacts them, perfect for creating a detectable message through changes in voltage, of which the Arduino reads and determines if the participants passed or failed, shutting down the system. In addition, it was important to make sure that the room stayed breathable but not let out fog, since the light particles hitting the fog allows the beams to become visible. These visible beams would travel from a source diode into a set of mirrors and by using mirrors we were able to have more beams inside the maze without requiring as many sensors or laser pointers. As stated earlier, these laser pointers cast beams from one side to a mirror on the other side and back multiple times before reaching their perspective photosensor which is hooked up to the Arduino that runs our code.

Design Process:

When designing the laser maze, we first looked at the shelving units we were going to receive as well as the dimensions of the room. With this information we were able to draw out potential designs of the maze, in which a U-shaped design was chosen to be used in the final product. Once we got everything drawn out, we converted the drawing to Tinker CAD. We adjusted the

CAD design until we all agreed that it should work. After designing the maze, we were able to start on a supply list, including items such as, lasers and photosensors, shelves, mirrors. After much delay, our supplies finally began coming in bit by bit. However, we reached an impediment when we were supplied with half the number of shelves that we were expecting. As a result, we had to redo our design due to the drastic downsizing in structure, while still trying to keep the central beam post design where a central pipe contained all the equipment. We found that the center post would be too intrusive, and not allow enough space for sideways movement when in the maze, so a final design of a simple strait corridor was required. In summary, we altered the design multiple times, with four major iterations: U-shaped corridor, U-shaped corridor with center pole, $\frac{1}{2}$ size U-shaped corridor with center pole, small strait corridor. Once we had our final layout of shelving, we put up black felt and curtains around the outside of the maze to make the room as dark as possible. Along the inside of the shelves, we put up foam padding to make the maze safer and keep our equipment safe. We spray painted the foam black to keep the room dark. With the foam being set up we started putting up the mirrors and lasers, completing the maze.

Results and reflection:

Although we went through several different design models and had to make several adjustments due the supply issues and time we had, we were still able to complete our laser maze. Our expectations were originally much higher but as we went through the design process, we realized we would have to lower our expectations due to time constraints. This allowed for us to make sure we would be able to make a good, well-built maze within the time limit and supplies we had.

RGBEAM PROJECT SUMMARY

Have you ever used a laser pointer on a presentation, but instantly regretted your decision to pick a red pointer on a red background? Our project is the perfect solution. With the RGB laser pointer, it doesn't matter what color your background is, because you can adjust the colors on the fly!

For this project, our goal was to create an adjustable laser pointer that can cycle through dozens of colors. In order to do this, we needed three lasers of different wavelengths. We needed red, green, and blue to produce the full spectrum. These colors when combined at different intensities would allow us to replicate almost any color. The challenge, however, comes when figuring out how to combine the beams. Our setup uses dichroic mirrors in an orientation that allows certain wavelengths of light to pass through while reflecting others. Dichroic mirrors reflect only 2 colors of light, and let the third pass on through. That way, the lasers can be coerced into merging with each other, because one color laser can end up in the same position and direction as another color, thus merging them.

So how does a laser emit light? A laser uses a p-n junction diode to emit coherent light in which all the waves are of the same frequency and phase. This process creates the acronym laser (Light Amplification by Stimulated Emission of Radiation). Since the waves are coherent, that is why laser beams are so focused and bright. Inside the diode excited electrons move from a lower-energy orbit to a higher-energy orbit around the atom's nucleus. When they return to their normal or "ground" state, the electrons emit photons (particles of light).

Along the way, we faced many challenges. Our first rgb laser piece that Alisafae found had a green laser that didn't function. We tried to remedy this, but we had a bit of an amazon fail that set us back a few days. When looking online, we tried to get the cheapest laser projectors possible. Unfortunately, it turns out that the cheapest laser projector on the market is an empty travel case. After that setback, we scrambled for a solution. Quickly, we found a laser lawn projector, one with full RGB capability. Amazon one day shipping is a savior. Once we received the projector, --and got around the safety screws-- we extracted all the electronic components.

Within the projector we found a controller board, a stepper motor with a diffraction grating, and finally, the laser assembly. The diffraction grating was used to generate the patterns that the laser point scatters into. Once we removed it, we realized that almost all the basic components for a sophisticated laser pointer were inside. There was the laser of course, but the controller board would also allow for advanced modes and settings. Thus, we decided that the controller board inside would be better than the arduino based system we planned on using.

However, this lawn projector was designed to be plugged into a wall, and to convert the 120 AC voltage from the outlet into 6 volt DC required for all the electrical components. While this may seem disastrous from a first look, it turns out that this is actually ideal. This is because the AC to DC converter is actually connected to the cord, and that the only energy inside the projector is that six volt DC. Even more fortunately, six volts is the exact amount of electricity that four AA batteries put out. So, we were easily able to add a battery pack inside.

Combining the lasers was easily the most difficult part of the setup. It's incredibly precise and needs to stay stable for long periods of time while being aimed and pointed at various things.

The setup we use requires almost perfect alignment in order to make the beams line up correctly. A degree adjustment for one of the mirrors can offset one of the lasers by a lot.

All of this effort led to our final product: the most powerful laser pointer that you could ever want.

Utilizing a class 3a laser from a lawn projector, this laser is supposed to be spread over several dozen feet projected onto a house's facade. However, since we are focusing in on a single point, this laser is much more powerful than the run-of-the-mill laser pointer. With this power, comes a large size. Our laser pointer is only 'hand-held' through a loose definition, owing to the heatsinks required to cool down the laser diodes.

Ultimately, all the successes and failures we faced taught us some important lessons. Not just about lasers, we had to learn to listen to everyone's ideas, even if we think that our own idea is better. We learned to make best use out of all the group members and play to each of our strengths. Another thing we learned is to be resourceful, making use of a non conventional way to obtain our lasers.

Works Cited

- Clarke, Roy. Techniques for Laser Combining. LumOptica,
lumoptica.com/file_download/6/Laser+beam+Combining.pdf.
- “Beam Combining.” StackPath, www.laserfocusworld.com/lasers-sources/article/16549530/photonic-frontiers-beam-combining-beam-combining-cranks-up-the-power.
- Rubio, Antonio. “Wavelength Beam Combining for Power and Brightness Scaling of Laser Systems.”
Lincoln Laboratory, MIT, 2014.
- Administrator. “What Is a Laser DIODE? Its Working, Construction, Types and Uses.” Electronics Hub,
22 Jan. 2018, www.electronicshub.org/laser-diode-working-structure-types-uses/.

The Introduction:

“We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard, because that goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one which we intend to win.” On September 12, 1962, John F Kennedy delivered his famous moon speech, in which he boldly promised to land men on the moon, the once-thought unattainable celestial in the night sky. Following in Kennedy’s footsteps our group decided that we didn’t want to do something easy, we wanted to do something hard--something that has never been done before. Many people scoffed at our tenacious attitude and our outlandish proposal to extract juice from bananas, but in the end, we proved them wrong. For our project we decided to produce a sour banana flavored gummy purely because no else has attempted to do it. Since the majority of banana flavors in candies nowadays come strictly from chemicals and artificial sweeteners, we decided to use natural flavoring directly extracted from bananas. However, this path was not an easy one to walk. If you have ever seen or eaten a banana, you will know that they are not exactly the wettest of fruits, and so we struggled for days. At some points, we considered throwing in the towel, switching our project for something more mainstream, something more attainable. However, we persevered, and after much trial and error, we ended up with two distinct methods to produce juice: a boiling method and a straining method. The straining method involves letting a banana puree sit over layers of cheesecloth to let the juice drip into a cup, but there were drawbacks. This method required manual reloading of the puree into the cup, and took about an hour of executing to yield a decent quantity of juice. The boiling method was simpler, as it is only boiling water and adding chopped banana, but its product was lower in quality, offering less of a potent fore-flavor and a stronger aftertaste, and similarly took hours of simmering to achieve. In the end, however, we decided to mix the two together for our recipe, a decision which paid off. We were on our way to making our banana gummies.

The Problem:

Every project comes from a problem. After all, a project attempts to address a problem and to find a solution. However, not all problems are created equal. While some problems exist on a global scale, or affect many people, the solving of other problems is less of a need and more of a want. This type of problem was the one we faced. Our problem was that there are practically no banana candies that use real flavoring; every single one is artificial or manufactured. Thus, we needed to find a solution in which we could produce a real banana flavor.

The Background Information:

We began with the theory “anything can be juiced” made by an anonymous user online. A silly idea at first but we couldn’t laugh it off, we would provide more evidence to prove this to be true. So we set out to juice bananas, a fruit that’s incredibly hard, and some would think impossible, to juice. To prove this theory though, we would have to look at the specific word juice, what exactly does it mean to juice something? We found the answer in splendid spoon’s blog comparing juice and smoothies, in it she said “Juicing is a process that extracts water and nutrients from fruits and vegetables, but discards the indigestible fiber naturally found in the produce.” This would mean it would have a watery consistency and essentially no pulp or meat of the banana in it. So we set out to juice a banana properly to help prove that anything can be juiced, even seemingly juiceless fruits.

The Design Process:

In the process of making banana candy, the struggle to produce a functional banana candy led to the development of many different methods for the extraction of banana juice. At the beginning of our project, we began attempting to milk the banana with a rag to juice it. This sadly didn't work out for us and it caused us to have to find a new method to get banana juice out of the bananas. After that first failed attempt, we thought that maybe banana juice was simply unattainable, and that we would be better off using a banana syrup. So, we tried to boil a banana puree in water to the point where it was a syrupy consistency. However, we couldn't reach such a consistency, and instead we just ended up making a cooked banana sludge, which was just a heated version of the puree we started with. We were disappointed, but our previous try gave us an idea. Instead of boiling the banana down to a sludge, we then tried to boil the banana puree to where it was still liquidy to get a sort of banana flavor infused water. However, this still contained lots of impurities and banana chunks, so we strained it out. Our process was as follows: we first put water in a pot and let it sit on the stove for 10-20 minutes so we could get it to a nice boil. Then we put in chopped up bananas and let it sit there for around 30-40 minutes. Afterwards the banana had turned into sort of a banana patty because of how condensed with juice and water it had become. Finally we had taken the banana patty and strained it with a strainer to collect the juice. The juice was not very concentrated but it was still banana juice, and while it didn't have a punchy flavor, it had a great aftertaste. This caused us to try and find another way of getting banana juice. Afterwards we had started trying to use our \$60 budget and we purchased cheese cloth, because after we had tried to use the rag to get juice out of the banana we thought that we needed a thinner cloth to get the juice. So we decided to blend the banana with hot water in the food processor and then put it through a straining type system with the cheese cloth. We poured the banana puree over top of around eight layers of cheesecloth which were rubber-banded to a cup underneath. Slowly, the juice dripped from the puree and into the cup. Once most of the liquid was gone from above the cheesecloth, we added another scoop to keep the process going. Occasionally, we had to scrape the residue off the top layer so it didn't clog the cloth and so it would allow the juice to drip out once again. This produced very concentrated banana juice but it also took the longest time which made it the least efficient way that we had attempted, so we decided to avoid it if possible. Finally the last method that we tried to do was the double boil method. Since the regular boil method's taste was so mild, we wanted to see if we could strengthen the banana flavor by using a double boil method. For this, we used the juice we obtained from the boil method, and boiled even more bananas inside of the juice. When we removed this juice, we found that it was indeed more potent, and the aftertaste was still there. So, for our final juice, we decided to use a half-and-half mix of the strained juice, with the strong fore-taste, and the double boil juice, to give our final juice a well rounded flavor.

The Results:

After all our testing, we finally came up with a resultant gummy recipe. We based our recipe off one we found online, however we changed quite a few things after our extensive testing in order to create the perfect gummy candy. For the final recipe we did a double boil method half strained mix of banana juice. We started off with a packet of gelatin in a bowl and then we added our banana water. Afterwards we added a quarter cup of corn syrup then added two teaspoons of sugar with a half teaspoon of citric acid and finally yellow food coloring to make it more appetizing. Then we microwaved it on and off for a minute and a half. After it

cooled down we poured it into our molds and refrigerated it for an hour. While refrigerating we mixed together our 1:5 ratio of citric acid to sugar in a bag. After the gummies are done setting in the fridge we coat the top of the gummies in the sour sugar mix. This recipe turned out to be delicious and even Timi, our team member who HATES bananas liked the final product. We hope you enjoy our sample too!

The Reflection:

At first we were all skeptical of if this would even work, we even came up with backup plans such as changing the fruit or making mints, but we stuck with it. Making the juice was a struggle and everytime we failed we were never quite sure where to start up with a new idea but we kept going and it paid off. This whole experience has taught everyone patience, perseverance, and trust in ourselves and our teammates. We as a team were able to jump past the hurdle of juicing a banana and can't express the pride that comes from it. I'm sure if we had more time we would find new ways to refine the product but we did the best anyone could possible do in the given time. Hopefully we can all carry the knowledge we gained through this experience into the future.



Hydrogel Drug Delivery

07/30/2021

Group 19

Ashley Zeman

Owen Adler

Zerna Akamah

Anisha Vuligonda

Introduction

The 3D polymer network of hydrogels has tremendous potential for drug delivery usage. This project examines sodium alginate-derived hydrogels. Our team has examined the effects of multiple variables on the speed, length of the release, and concentration of the released substance.

Problem

Our goal is to create a hydrogel with a maximized concentration of released medicine with as sustained release as possible. To accomplish this, the team must test different variables to see what heightens the potential of hydrogels for sustained medicine release.

Goal

Our goal is to create both an efficient and effective capsule casing for the most concentrated release of medicine, which is modeled by a yellow dye.

Background

This experiment examines the properties of hydrogels and how to maximize their effectiveness. A hydrogel is a flexible three-dimensional network of hydrophilic polymers. The hydrogel absorbs water-soluble substances and can later release them. In this project, the hydrogel is made up of Sodium Alginate and Calcium Chloride. The hydrogel contains Tartrazine dye which serves as a model for medicine.

The effectiveness of the hydrogels is examined using a Spectrophotometer. A Spectrophotometer is used to measure the absorbance of a liquid sample. The absorbance is a measurement of the amount of light absorbed by a substance. Thus, the spectrophotometer can measure the amount of dye present in the solution as more dye will absorb more light. A calibration curve can be produced using known concentrations of dye in order to construct a linear equation that can calculate the dye concentration from a measured absorbance.

Apparatus

- Beakers
- Spatula
- Graduated cylinder
- Electronic balance
- Magnetic stirrer and stir rod
- ProPette single channel pipettor
- Vacuum pump
- Filter paper

- Buchner flask
- Weigh boats
- Cuvettes
- Syringe
- Spatula
- Spectrophotometer

Resources

- Calcium Chloride
- Tartrazine
- Distilled water
- Sodium Alginate
- Vinegar (5% acidity)

Procedure

I. Stock Solution Preparation

- A. Prepare 30mL of 6.0% wt Calcium Chloride solution.
 1. The concentrations of 10% and 2% were also prepared and tested
- B. Prepare 0.5g/L tartrazine solution
- C. Add 0.2 g alginate for every 10 mL of tartrazine solution. Use a magnetic stirrer overnight.
 1. This alginate solution was varied with 0.05 and 0.4 g per 10mL in the following trials.

II. Spectrophotometer Calibration

- A. The original stock solution containing 0.5 g/L of Tartrazine is diluted with deionized water to form lower concentrations.
 1. The dilutions were performed with a graduated cylinder and consisted of half dilutions and usage of the $\text{Concentration}_1 \times \text{Volume}_1 = \text{Concentration}_2 \times \text{Volume}_2$ dilution formula.
- B. The various diluted concentrations of Tartrazine were sampled and analyzed with the Spectrophotometer to find the absorbance for the different concentrations.
- C. The absorbance and Dye concentrations were graphed to produce a concentration curve that can be used to analyze the dye concentrations of the beads at a later time.

III. Bead Production

- A. Add filter paper into the Büchner funnel attached to the flask. Connect to the vacuum pump and wet the paper
- B. Draw >4mL of alginate solution into the syringe
- C. Drop 3mL of solution into calcium chloride bath in beads. Spread them out as beads tend to combine
- D. Wait for the desired time for the beads to crosslink. Rinse beads with deionized water and remove them from the funnel to dry.
 1. This time was 2,5, or 10 minutes

IV. Dye Release

- A. Fill a beaker with 100 mL of distilled water
- B. Transfer dried beads to the beaker using a spatula.
- C. Immediately take a sample from the beaker with a pipette. Measure the sample's absorbance in the spectrophotometer.
- D. Add stir rod and begin stirring at 150 rpm.
- E. Take absorbance readings every five minutes until values stabilize (in most cases ~60 minutes).

V. Supplementary Trials

- A. The procedure mostly remained the same for the supplementary trials
- B. Before dye release, beads were removed to create samples at a standard 2.5 grams to dissolve.
- C. For the acidic environment trials, 80ml of deionized water was added to 20 ml of 5% acidity vinegar.

Design Process

Design Iterations

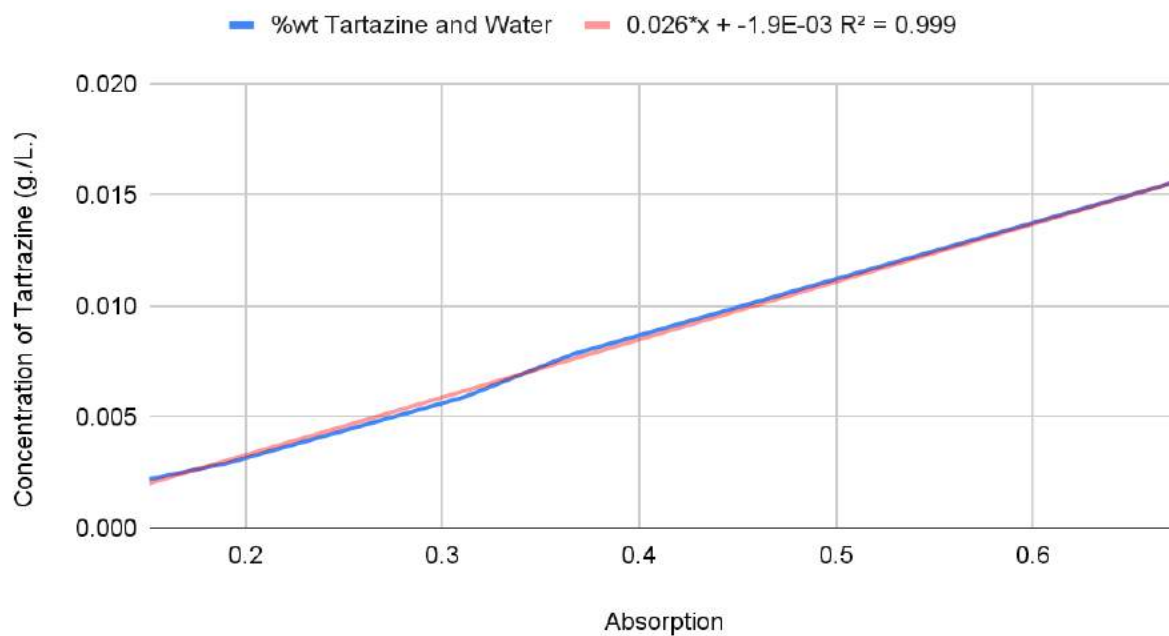
- With the intent of maximizing the total release and sustain of the dye, variables were adjusted to determine the best ratio of resources and the most advantageous procedure.
- The main procedure of the experiment was based on the Alginate Bead Drug Delivery Standard Operating Procedure.

- The amounts of Calcium Chloride and Sodium Alginate were varied with both less and more concentrated solutions than the solutions detailed in the SOP.
- The crosslinking time was varied with more and less time than done in the first attempt.
- Our faculty mentor gave our group a few ideas of how to continue testing to obtain more data.
 - The suggestions included standardizing the mass of the beads to be dissolved and to dissolve in an acidic solution.
 - These secondary trials were performed using the two most successful methods from the previous trials.

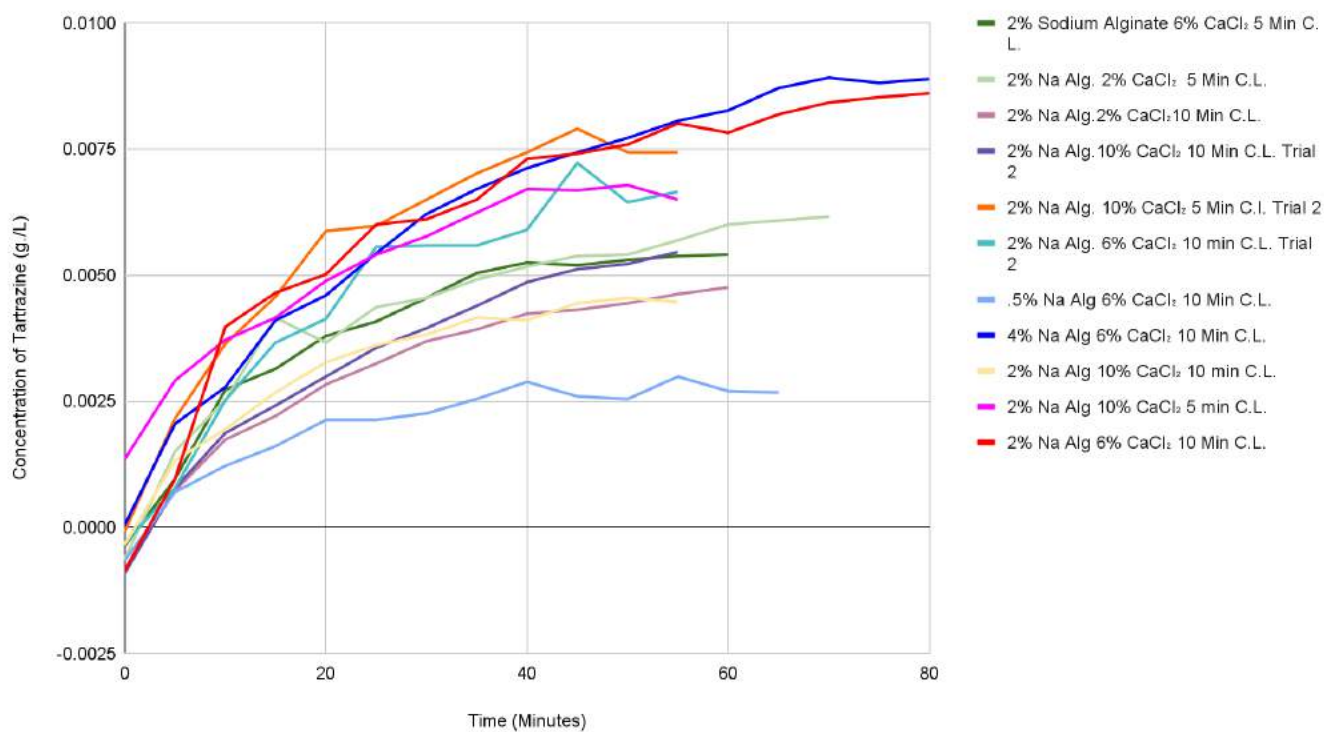
Calculation

- To perform dilutions, the equation $\text{Concentration 1} * \text{Volume 1} = \text{Concentration 2} * \text{Volume 2}$ was used to calculate the amount of dye to be diluted
 - Example with 5ml of 0.25 g/L of Tartrazine solution.
 - $5\text{ml of } 0.25 \text{ g/L} * 5 \text{ ml} = 10 \text{ ml} * ? \text{ g/l}$
 - $= 0.125 \text{ g/L}$
- Calculations with Concentration curve. The linear equation $0.026 * \text{Absorbance} + -1.9\text{E-}03 = \text{Concentration of Tartrazine (g/L)}$
 - This equation can be used with the found absorbance values from the experimental data to determine the concentration of tartrazine in the solution released by the alginate beads
 - $\text{Absorbance} = 0.415 \quad 0.026 * 0.415 + -1.9\text{E-}03 = 0.00889 \text{ g/L Tartrazine}$

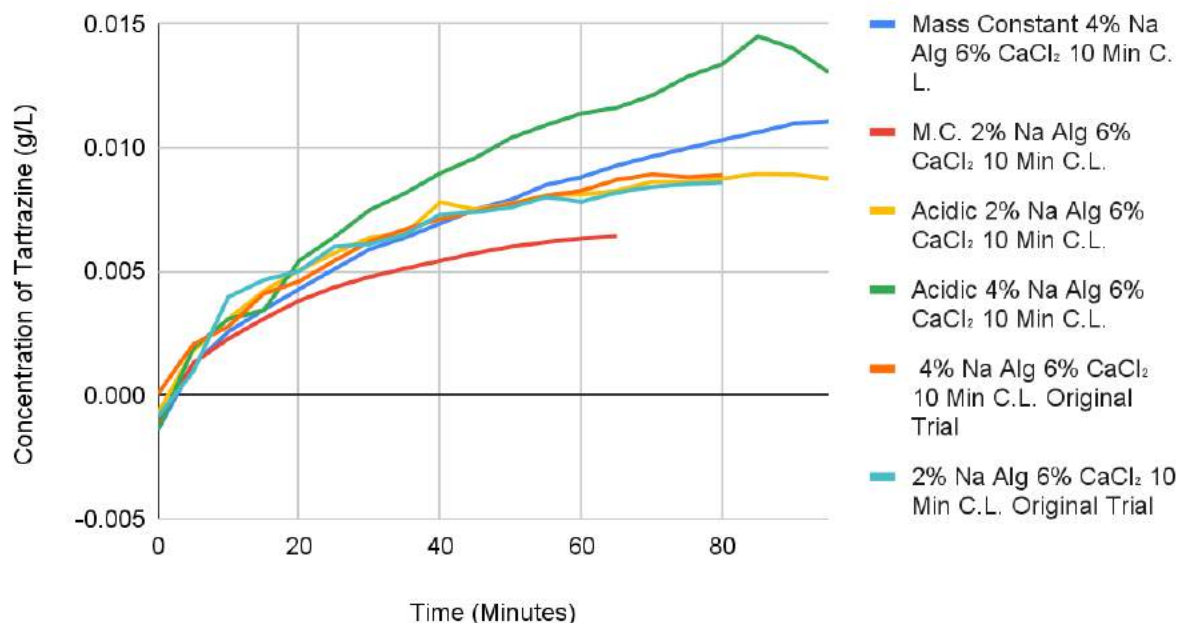
Calibration Curve



Tartrazine Concentration over time



Secondary Trials



Results

We changed variables one at a time and tested the new samples. We discovered that the hydrogel with the highest amount of Sodium alginate was the most successful while the hydrogel with a lower amount of Sodium alginate was the least successful. Our trials with only 2 minutes of crosslinking time failed to produce fully formed beads and thus could not produce dye release data.

Upon our investigations, we found that a sample made with 4% sodium alginate solution, 6 wt% of calcium chloride, and a cross-linkage time of 10 minutes proved to be the best sample for maximizing dye concentration release for strength and sustainability. The concentration reached the highest value before reaching a stabilizing point later than other trials. This shows the formulation contains the most dye and releases for the longest amount of time. The second best sample contained 2% sodium alginate, 6 wt% calcium chloride, and 10 minutes crosslink. These trials have shown that a greater crosslink time paired with more sodium alginate is best for increasing dye concentration, while a change in calcium chloride fails to produce any substantial advantages.

Secondary trials were done to examine the effect of balancing the mass of beads dissolved and the impact of dissolving in an acidic environment. The results from these secondary trials are unusual. To balance the masses, the amount of 4% sodium alginate had to be greatly reduced yet it achieved a higher dye concentration than the trials without a mass being controlled, while the 2% sodium alginate had significantly reduced dye concentration. The acidic environment with a pH of approximately 3 increased the concentration of dye and speed of dye release for the 4% sodium alginate trial yet it led to a negligible change in concentration for the 2% sodium alginate solution.

Potential Errors

One main issue is that the amount of mass was not taken into account during the original trials. While the volume of solutions used and the number of beads were approximately the same for all trials, the mass was allowed to vary due to the different densities of hydrogels formed by the trials.

Another potential source of error was the stock solutions. This project consisted of multiple stock solutions which were all produced at the beginning of testing. Therefore tests were made with the same stock solutions different times after their production, which could have potentially altered concentrations. Since the same ProPette pipette was used to take samples from three simultaneously running dissolving tests, it could have potentially not been cleaned out properly and caused contamination. The cuvettes used in the spectrophotometer would have led to inaccurate results if they had any smudges or irregularities.

Reflection

This project improved the team's ability to enhance a design based on trial results. Our project heavily relied on performing different experiments as well as interpreting their results. We gained experience working in a lab environment and using new lab equipment, such as the spectrophotometer.

We also researched modern medicine, which gave us an insight into why pills take time to work and how they are engineered to maximize speed, effectiveness, or time of release. Our team gained knowledge on the properties of hydrogels and their potential applications for medicine.

Overall, this project gave us as a team the opportunity to have an in-depth encounter with the chemical engineering department and laboratory.

Works Cited

"Alginate Bead Drug Delivery Standard Operating Procedure." Rose-Hulman Institute of Technology Chemical Engineering Department. Updated 26 November 2016.

Bahram, Morteza, et al. "An Introduction to Hydrogels and Some Recent Applications." *Emerging Concepts in Analysis and Applications of Hydrogels*. InTech Open. 24 August 2016.
[https://www.intechopen.com/chapters/51535#:~:text=A%20hydrogel%20is%20a%20three,\(1960\)%20%5B1%5D](https://www.intechopen.com/chapters/51535#:~:text=A%20hydrogel%20is%20a%20three,(1960)%20%5B1%5D)

Mandal, Sanchita, et al. "Sustained release of a water-soluble drug from alginate matrix tablets prepared by wet granulation method." *AAPS PharmSciTech*. National Library of Medicine. 13 November 2009. <https://pubmed.ncbi.nlm.nih.gov/19911286/>

Trebuchet

Intro

The problem that was encountered was not being able to throw a tennis ball more than 100 ft. The way this problem could be fixed is through a rolling whipper trebuchet. This type of trebuchet involves two separate arms that fall in a circular motion to build up a lot of energy and then, like the name suggests, whip the ball off into the distance.

Background Information

The project is mainly focused on the principles of physics. The main concepts that were used in the trebuchet were conservation of energy, potential energy, kinetic energy, kinematics, angular momentum, and rotational kinematics. Newton's Law of Conservation of energy explains how the energy created from the trebuchet's weight is converted into energy that can be used to propel a projectile out of the trebuchet at a high velocity. Potential energy is energy that is associated with height and mass. When the counterweight is at the start of the launching process, it is at the highest point giving it the most potential energy to release into kinetic energy. The equation $(\text{mass}) * (\text{gravity}) * (\text{height}) = (\text{Potential Energy})$ shows this relationship since they are being multiplied together. In order to use the potential energy that is built up by increasing these variables it needs to be converted into kinetic energy.

Kinetic energy is the energy associated with motion. With more velocity, comes more kinetic energy. Potential energy converted to kinetic energy is what drives the tennis ball past the 100 foot goal. Another concept used in this project was angular momentum. Like regular linear momentum, the equation uses $(\text{mass}) * (\text{velocity})$ but then uses $* (\text{radius})$ as well. If we use a counterweight of higher mass or have a longer string to increase the velocity of the tennis ball, it will build up more momentum and be a lot harder to stop. Overall, the process or system that we are using for the trebuchet is kinematics and rotational kinematics. Kinematics is the study that describes the mechanics of the movement of an object without worrying about the forces that cause the movement.

Not all of the potential energy stored in the counterweight initially will be converted into kinetic energy useful for launching the projectile. Some potential energy will remain as potential energy at the final position of the counterweight since it will not be resting on the ground. Some of it will be lost in the form of thermal energy due to friction within the system of the trebuchet and some will be lost in the vibrations of the system and environment, also known as sound. The optimal strategy to allow the projectile to travel the furthest is to launch it at a 45 degree angle with minimal energy lost. Structural stress could also add issues, so strong, sturdy materials were chosen to construct the trebuchet. If too much normal stress or shear stress is given to any part of the system, then it will have a failure. Every part was chosen for its ability to withstand that stress force. There are many popular trebuchet designs throughout history and the design that was chosen for this project was the rolling whipper design. The principal behind this design is for it to whip the ball super fast by using the high potential energy of the counterweights.

Design Process

For the design process, discussions alongside sketching out the trebuchet were used. From sketching and conversation the dimensions and parts that were necessary were determined. Overall, the trebuchet uses 2x4s and plywood, bearings, wheels, nuts, washers, bolts, weights, rope, a sling, and metal bars to hold the arm and the weights. After discussing how the team would go about building the trebuchet, it started to come together. Overall, keeping the design process simple and to the point allowed for an effective building process that couldn't have been accomplished otherwise.

Results

After completing the trebuchet, many trial runs were done to test how well the trebuchet works. In order to have a more accurate reading of the tests, a measuring wheel was used for each launch, the same type of tennis ball was used, and the arm was dropped from the same height each time. After going through all the trial runs the highest launch was 208 ft. This distance was far below the estimated 281.2m. This is because the equation didn't factor in air drag, friction of the system, human errors, and inconsistent launch angles. The one thing that is the easiest to improve upon in the trebuchet's flaws would be its inconsistent launching angle and release mechanism. During the trials, the trebuchet would sometimes release too early or late and cause the tennis ball to launch shorter distances. This would be the focus of the trebuchet had there been more time to work on it. Also, it would be good to have a consistent 25 pounds on each side of the counterweight arm. Right now the arm is unbalanced which may be causing it to launch irregularly. The limit of weight for this project was 50 pounds for a counterweight. However, only 45lbs were available to use so it is uneven. Overall, the trebuchet works very well, but could also work way better with some modifications.

Reflection

Building the trebuchet was a great way to connect a bit more with the engineering world, and apply what they use daily into the project. There were many great things to take away from this project like the design process, planning out what and how to do the task, and the many astounding principles in physics that apply to the trebuchet. Throughout the project, these lessons were used as best as they possibly could in order to build a great trebuchet. Alongside the hardware aspect of the trebuchet there were other great lessons used to propel the project forward. One of these lessons is learning from mistakes. Throughout the project there were many mistakes made. After they were made, those issues never seemed to come up again. One example of this is in the building stage of the trebuchet, when the base was being built. The base's 2 x 4s were connected by 2 inch screws (due to a miscommunication). Since a 2 x 4 is only 1.5 inches wide the screws didn't support the structure well. After this problem, 3 inch screws were used in the frame from there on out. This problem never really came up again. Another very important lesson that was used throughout this project was teamwork. By using teamwork, things that would have taken forever alone, got finished very quickly. In conclusion, building a trebuchet not only launched a tennis ball 208 ft, it also taught and used many great lessons.

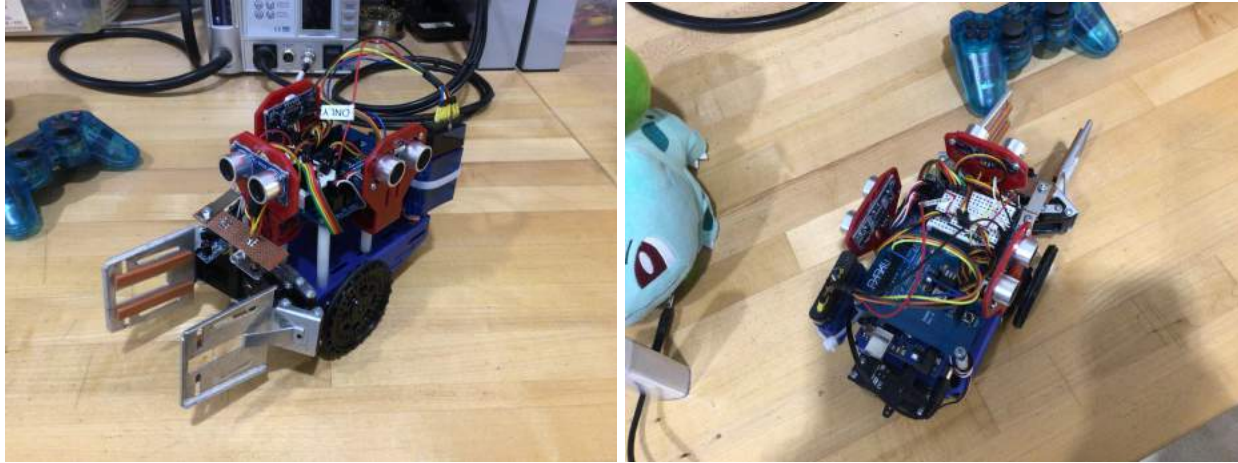
Group 20 Aluminum Potatoes: Jamie G, Brett, Quentin, Evan,

Team #21

Bianca Ravesloot and Ben Egelseer

Our team, part of the Microcontroller Arduino groups, were tasked with taking a preset arduino build and programming it to perform various functions. As mentioned, the small “robot” was already put together before we started the project, with everything being wired and soldered in its entirety. We had two main functions, one to program the robot to respond to a PlayStation 2 controller, and a second to give the robot autonomous movement when searching for a specific infrared signal. This signal was given off by a mini arduino board within a Pokémon plush, and had its own separate set of necessary coding.

Before jumping headfirst into the project, each team went through a series of labs that explored an unique concept part of the project as a whole. In total, there were 6 main labs, with a few extra after the fact to explain the necessary programming. Starting out, our instructor did a lecture on both DC and AC circuits. We were given materials to put together a few different layouts, as well as a digital multimeter in order to properly calculate the circuit’s resistor and voltage values. AC circuits in particular needed an oscilloscope, since the voltage of said circuits contains an important relationship with the relative frequencies. From there, we moved on to the infrared section of our labs, where we analyzed how signals could be transmitted to the IR Phototransistors and back to the LED it originated from. We used our gathered information to create a mock-up barcode reader, and later on to determine how our programming could properly emit the signals we needed. The rest of our labs were used as an introduction to our arduino boards, and a surface level look at how to code them to perform our end-goal intentions.



As for our design, the only variable we controlled was everything having to do with the programming. Our arduino and sensors were put together ahead of time as mentioned before, so our main goal became adapting the code we were given to best fit our robot and the tasks it needed to accomplish. On paper, there wasn't much of a process or design phase as other groups, but we performed quite a few tests to see if our code was correct. These tests included: running the robot through the mazes set up as a competition for the robots, checking the sensors, and altering the code to record certain bits of information in a separate window. After each test we changed the code accordingly, then went back to test again until we found our solution.

All in all, our project met all of our main goals. Although it may not have completed its tasks flawlessly, our robot was still able to do well in each competition it faced. Our fastest time for the controller portion of the tests was four minutes and twenty seconds, beating a couple different teams in the process. Addressing the project as a whole, we as a team believe that it would've gone a lot smoother if we had previous experience with arduinos or it's programs. We also believe that the code shouldn't have been handed to us as it was, considering that it made the project a lot less beneficial to our learning, and didn't give us as much of an experience with taking a shot at the programming ourselves. However it still was a great project, and fun to experiment with the different sets of files and sensors.

Arduino Bots

Introduction

For the past 2 weeks, we have been programming, designing, and learning about Arduino boards. But what is an Arduino board?

Arduino is an embedded microcontroller. Its sole purpose is to follow instructions, using any or all devices hooked up to it. Our goal was to use this device to control a small robot equipped with a claw, three servo motors, and three ultrasonic sensors to locate and flag five Pokemon. We created two programs, one PS2 remote-controlled and the other autonomously controlled.

Problem Statement

Complete the challenge and flag all five pokemon in autonomous and remote controller mode.

Background Information

We began this project with several lectures and labs on various topics. First and foremost, we learned about two types of electrical currents: alternating and direct currents. We did multiple labs regarding this and learned about various circuitry including resistors and transistors. Next, we continued with lectures and labs concerned with the various components of the Arduino robot that we would need later on. These components included servo motors, lights, and ultrasonic sensors. We coded these components using Arduino. As mentioned above it is an embedded microcontroller. Arduino has a specific program for it. It uses a lot of the same code words and setup as c++. After learning the basics of electrical engineering, we were ready to begin our challenge. The challenge is a maze with an assortment of pokemon inside, but the maze is inconsistent. We would need to maneuver through multiple mazes as well. There are two challenges: autonomous and remote-controlled. For the autonomous challenge, the robot will have to locate and flag all of the pokemon in a maze. For the remote control challenge, the robot will be controlled via PS2 controller and have to flag ten Pokemon in the classroom. The robot will need to send a signal to the pokemon, which contains an infrared led. When the infrared lights light up, the pokemon will send a signal back. This lets the robot know what type of pokemon it is. The remote control challenge will be timed and points will be awarded for the group's ranking. The autonomous challenge has a different set of rules, more focused on deductions. For example, on the autonomous challenge, if a robot runs into a wall for more than three seconds, three points will be deducted. Another deduction for the autonomous challenge is getting stuck, which results in a five-point deduction. Once each challenge is completed, the robot must return to the start position.

Design Process

Our process was structured with the layout of the learning process. We followed labs and lectures to create the basis of our design. Many elements of our program were taken from the already made testing labs. We used this to see how the robot was coded and controlled, such as "servoLeft.writeMicroseconds(1500 + speedLeft);". This piece sounds complex, but when you break it down it's not, servoLeft is directly talking about the left servo. WriteMicroseconds, explain the units for the section in the parentheses and so the (1500 + speedLeft) is the speed of the left servo, in microseconds. We used this exact method to integrate these ideas into our programs and learned about the different components of Arduino programming. Our autonomous program contains parts from 3 different programs: ultrasonic sensor and sonar distance, start and

Team #22
Allison Kirk
Dane Blackburn

stop with ramping, and IR scanner and pokemon tester. The ultrasonic sensor and sonar distance program are used to control the 3 ultrasonic sensors. This allows the robot to use ultrasonic detection or otherwise known as echolocation. It sends sound waves to an object that's in front of it and uses that information to decipher how far that item is away. We used such codes to control what ultrasonic sensor is being used and how. For example, we set it up like this, "if((SonarDistanceFront <= TooCloseCMFront) && (SonarDistanceFront > 0))". The SonarDistanceFront is referring to the front sensor. TooCloseCMFront is a variable we assigned a number to, in this case, 20 cm. So when the front sensor "sees" something 20 cms away, it does the next action, such as maneuver or move. That brings us to our next program, Start and Stop with Ramping. This is a pretty easy program and one of the first ones we look at. It focuses on the servos and controlling the speed, and time set. We added a new function called void maneuver, to set our servos, this allows for better flow. The initial program was integrated into our new function, as follows:

```
"void maneuver(int speedLeft, int speedRight, int msTime)
  servoLeft.writeMicroseconds(1500 + speedLeft);
  servoRight.writeMicroseconds(1500 - speedRight);"
```

We structured this code using a series of if and else statements to set the parameters for when to enable certain lines of codes. Last, but not least, we used the IR scanner and pokemon tester mainly for sensing the infrared lights on each pokemon. This was used exactly as found in the program then we blended it into our program.

Results

When testing our autonomous program we found that it could decently maneuver the maze. Walls caused some grief but we coded an autocorrect feature to put the robot back on track. If a value from the front sensor is repeated long enough then the robot backs up or does a 180° turn. But corners were a problem, especially convex corners. The sensors only have a certain view and actually can't see a convex corner, which makes programming them impossible.

When testing our remote control robot we ended up getting a time of 4:29. In this challenge, we had to maneuver around a classroom and flag ten infrared Pokemon circuits. When doing this the remote control and infrared portion worked well. We however had a slight setback with one of the infrared Pokemon circuits.

Reflection

Our challenge brought forth many learning opportunities and obvious problems. Time management is an important one. The full extent of the project was not explained to us, leaving much room for confusion and distractions. We had a general idea of the work required and should have divided and organized a schedule. But we were missing key components and did not understand the amount of work that needed to be done. The same is true for the presentation of our poster and project summary. But overall, the challenge was fun and made for some cool results. When we actually got the robot to work, we felt a great sense of pride in our accomplishments. We programmed a functioning robot! This project allowed us to gain a greater understanding of electrical and computer engineering, and Rose Hulman's unique teaching styles.

Team #22

Allison Kirk

Dane Blackburn

Embedded microcontrollers are very important to many different aspects of engineering and our society. From Mars rovers to automated toilets, they help us achieve anything. We can take this knowledge and understanding to new heights as we reach for the stars.

Arduino Bots

Introduction

For the past 2 weeks, we have been programming, designing, and learning about Arduino boards. But what is an Arduino board?

Arduino is an embedded microcontroller. Its sole purpose is to follow instructions, using any or all devices hooked up to it. Our goal was to use this device to control a small robot equipped with a claw, three servo motors, and three ultrasonic sensors to locate and flag five Pokemon. We created two programs, one PS2 remote-controlled and the other autonomously controlled.

Problem Statement

Complete the challenge and flag all five pokemon in autonomous and remote controller mode.

Background Information

We began this project with several lectures and labs on various topics. First and foremost, we learned about two types of electrical currents: alternating and direct currents. We did multiple labs regarding this and learned about various circuitry including resistors and transistors. Next, we continued with lectures and labs concerned with the various components of the Arduino robot that we would need later on. These components included servo motors, lights, and ultrasonic sensors. We coded these components using Arduino. As mentioned above it is an embedded microcontroller. Arduino has a specific program for it. It uses a lot of the same code words and setup as c++. After learning the basics of electrical engineering, we were ready to begin our challenge. The challenge is a maze with an assortment of pokemon inside, but the maze is inconsistent. We would need to maneuver through multiple mazes as well. There are two challenges: autonomous and remote-controlled. For the autonomous challenge, the robot will have to locate and flag all of the pokemon in a maze. For the remote control challenge, the robot will be controlled via PS2 controller and have to flag ten Pokemon in the classroom. The robot will need to send a signal to the pokemon, which contains an infrared led. When the infrared lights light up, the pokemon will send a signal back. This lets the robot know what type of pokemon it is. The remote control challenge will be timed and points will be awarded for the group's ranking. The autonomous challenge has a different set of rules, more focused on deductions. For example, on the autonomous challenge, if a robot runs into a wall for more than three seconds, three points will be deducted. Another deduction for the autonomous challenge is getting stuck, which results in a five-point deduction. Once each challenge is completed, the robot must return to the start position.

Design Process

Our process was structured with the layout of the learning process. We followed labs and lectures to create the basis of our design. Many elements of our program were taken from the already made testing labs. We used this to see how the robot was coded and controlled, such as "servoLeft.writeMicroseconds(1500 + speedLeft);". This piece sounds complex, but when you break it down it's not, servoLeft is directly talking about the left servo. WriteMicroseconds, explain the units for the section in the parentheses and so the (1500 + speedLeft) is the speed of the left servo, in microseconds. We used this exact method to integrate these ideas into our programs and learned about the different components of Arduino programming. Our autonomous program contains parts from 3 different programs: ultrasonic sensor and sonar distance, start and

Team #22
Allison Kirk
Dane Blackburn

stop with ramping, and IR scanner and pokemon tester. The ultrasonic sensor and sonar distance program are used to control the 3 ultrasonic sensors. This allows the robot to use ultrasonic detection or otherwise known as echolocation. It sends sound waves to an object that's in front of it and uses that information to decipher how far that item is away. We used such codes to control what ultrasonic sensor is being used and how. For example, we set it up like this, "if((SonarDistanceFront <= TooCloseCMFront) && (SonarDistanceFront > 0))". The SonarDistanceFront is referring to the front sensor. TooCloseCMFront is a variable we assigned a number to, in this case, 20 cm. So when the front sensor "sees" something 20 cms away, it does the next action, such as maneuver or move. That brings us to our next program, Start and Stop with Ramping. This is a pretty easy program and one of the first ones we look at. It focuses on the servos and controlling the speed, and time set. We added a new function called void maneuver, to set our servos, this allows for better flow. The initial program was integrated into our new function, as follows:

```
"void maneuver(int speedLeft, int speedRight, int msTime)
  servoLeft.writeMicroseconds(1500 + speedLeft);
  servoRight.writeMicroseconds(1500 - speedRight);"
```

We structured this code using a series of if and else statements to set the parameters for when to enable certain lines of codes. Last, but not least, we used the IR scanner and pokemon tester mainly for sensing the infrared lights on each pokemon. This was used exactly as found in the program then we blended it into our program.

Results

When testing our autonomous program we found that it could decently maneuver the maze. Walls caused some grief but we coded an autocorrect feature to put the robot back on track. If a value from the front sensor is repeated long enough then the robot backs up or does a 180° turn. But corners were a problem, especially convex corners. The sensors only have a certain view and actually can't see a convex corner, which makes programming them impossible.

When testing our remote control robot we ended up getting a time of 4:29. In this challenge, we had to maneuver around a classroom and flag ten infrared Pokemon circuits. When doing this the remote control and infrared portion worked well. We however had a slight setback with one of the infrared Pokemon circuits.

Reflection

Our challenge brought forth many learning opportunities and obvious problems. Time management is an important one. The full extent of the project was not explained to us, leaving much room for confusion and distractions. We had a general idea of the work required and should have divided and organized a schedule. But we were missing key components and did not understand the amount of work that needed to be done. The same is true for the presentation of our poster and project summary. But overall, the challenge was fun and made for some cool results. When we actually got the robot to work, we felt a great sense of pride in our accomplishments. We programmed a functioning robot! This project allowed us to gain a greater understanding of electrical and computer engineering, and Rose Hulman's unique teaching styles.

Team #22

Allison Kirk

Dane Blackburn

Embedded microcontrollers are very important to many different aspects of engineering and our society. From Mars rovers to automated toilets, they help us achieve anything. We can take this knowledge and understanding to new heights as we reach for the stars.



GOTTA TAG 'EM ALL

It's crazy to think how a little green board can control a robot twice its size. In this case, it's the Arduino board controlling the servo motors, ultrasonic sensors, and IR sensors. Arduino boards are able to read inputs such as light on a sensor or a finger on a button. It also can turn it into an output such as activating a motor and turning on an LED. The servo motors control the wheels and the grabbers. Servo motors are motors that turn to a certain angle. The motors that control the wheels turn at a 360 angle and the motor that controls the grabbers turn at a 180 angle. Ultrasonic sensors are sensors that measure the distance of a target object by emitting ultrasonic sound waves. We use the ultrasonic sensors in the maze so the robot knows when it's about to hit a wall. After the robot knows its distance from the wall and it's too close the robot will turn either right or left depending on the distance of the walls to the sides of the robot. IR sensors work when an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver. In the maze, the Pokemon have IR LED lights on them. When the receiver detects the LED the robot turns around and then moves on to find the other Pokemon.

For our project, we have to make a robot that successfully makes it through 2 mazes and tags all the pokemon.

In the days leading up to the actual robot project, we had to do 10 labs. These labs included working with breadboards and math. In the first lab, we learned about Voltmeters, Ohmmeters, and DC Power Supplies. We also learned about resistors and how to read their values. For lab 2 we learned how to work with Oscilloscopes and Function Generators. In lab 3 we started working with breadboards, LED lights, and power supplies. We powered a red LED light with a push of a button. We also had to measure the voltage that went across the LED while the button was pushed. At the end of lab 3, we made a barcode scanner with the use of an IR phototransistor. Lab 4 is when we started working with the servo motors. We learned how to work with the Arduino boards in lab 5 by programming it to turn off and on a red LED with the use of a button. In lab 6 our goal was to read whether a value is on and off using analog pins. In labs 7-10 we mainly worked with a breadboard and the Arduino circuit board. We learned more about how to program the Arduino board. The equations we used in these labs were Ohm's law, Series resistance, Parallel resistance, and Voltage divider. The labs helped make the final project easier to do. For our project, we had a score sheet. Getting first place in the PS2 controlled part gets the highest score. In the autonomous part, making it to the end and back to the beginning gets you the highest score. The deductions of the maze are if your robot goes against a wall for 3 seconds you lose 3 points. If the robot died during the maze then you would lose 20 points. You lose 10 points if the robot needs manual assistance. For the bonus points, you can get +10 if the robot turns at a perfect right angle. If the robot flags the special pokemon then that's an extra 30 points. For



adding cool features to your robot and the program it can be up to 15+ points. Whoever has the top score on the maze receives an extra 20+ points.

Because our bots were already built and wired for us, our design process was more focused on the various labs previously outlined and the actual process of coding the robot. After we went through these various labs introducing us to the different command types and syntax of the Arduino program, we started working through how to create our own unique code to have the robot do the various challenges we were tasked with. We were given different base codes that we could use and change, fitting them to what we were trying to do. There were also various subroutines that the robots had to run to function properly with the codes written. In the Arduino program, there are various libraries at our disposal that we utilized in our code. These libraries granted us access to new command codes and syntax for the various components of our bot. These libraries included a servo library, an IR remote library, and a PS2 controller library. For the most part, programming the code for our bots was orchestrated by the students in their respective groups. We were, however, allowed and encouraged to ask for help if we got stuck and could not figure out how to fix a problem in our code. There were many resources for us to ask for help, including our professor, Dr. Chris Miller, and the GAs, Luca, Casey, and Kaden. We used the process of trial and error to see how the robot functions with different coding, observing how it behaved and what functioned best. We tested the robot many times in the various mazes to find the right values for turns and moving forwards or backward. We did this through things like delays and timing.

At first, our design did not function how we wanted it to or how we programmed it to. This was quite frustrating because it would work one day then stop working the next. After working through various issues and asking for help, we did end up creating a code that allowed the robot to function properly and do some extra challenges and features. Because of the extra time we had to play around with the code, we added a celebratory dance to our robot after it found the pokemon. We also added an RGB LED that lights up to indicate that a Pokemon was found. This LED is programmed for both the autonomous and remote-controlled portions of the robot. The dance, however, is not in the coding of the remote-controlled portion. We also figured out how to find the special pokemon that is harder to get to. It was very satisfying and we were triumphant when we finally got all of the coding to function to our desires and have the extra features.

While learning how to code and interact with our robots, we ran into multiple issues, all of varying degrees and relations. Some of these problems were related to a simple syntax error, like having an extra curly bracket `}`, while others were more complicated. One of these problems included having our code read and executed in the wrong order. Some of these problems we encountered were simple enough to fix; changing a line of code or deleting the unneeded bracket. Some, however, needed the help



BMO TECH













Group #23

Jylena Blanding & Abigail Bouravnev

of the GAs. For one of our problems, we had to recode an entire section because there was a syntax error that could not be rewritten correctly. For this problem, Casey helped us to use the proper coding and commands so that the code would run properly. We also had help with other things, like the IR sensors and the clashing of two libraries. This problem was solved by singling out the different lines of programming and moving them to a new sketch and seeing where the problem began then fixing it, with the help of another GA, Kaden. We also ended up replacing some of the parts of our robot because they were not functioning properly. In the end, we ended with four to five different remotes tried three remote receivers, a different bot entirely, and new grips on the wheels. We also had to have Kaden adjust our gripper servo so that it did not scrape along the ground. After working through the various problems, the bot ran pretty smoothly, allowing us time to add extra features, like an RGB LED. This also posed a few problems however because we had to figure out how to set the LED to an analog output signal and we had to move various other wires on our board for the proper functioning of the LED. In the end, because of different problems we ran into, we had four to five different remotes, three new remote readers, new grips on our wheels, and an adjusted grabber. This process of trial and error taught us that it is good to ask for help when you need it. It also taught us that coding is very sensitive and precise.

The first thing that we needed to learn is the basics of robotics such as current, signals, circuits, resistors, and servo motors. For the first few days, we conducted experiments with simple circuits to power a LED light, by either hooking up a power supply or by wiring in a switch or button to turn on the light. We used a device called a breadboard to easily create the circuit. Upon creating the circuits we would then try to determine how much current is flowing

Diagram illustrating the connection between an Arduino Uno and a Microchip ATmega256P. The Arduino is connected to the Microchip via an I2C interface. The Microchip is also connected to a 2x10 pin header, a 2x10 pin header, and a 2x10 pin header.

Image	Schematic	Description
		DC (Direct Current) Voltage Source Provides a constant voltage to power your circuit. Batteries and the DC power supply are both examples of DC voltage sources.
		AC (Alternating Current) Voltage Source Produces a changing voltage. The function generator can act as an AC source. While we won't be connecting anything to the outlets, the power in the outlets is AC as well.
		Resistor Resistors reduce current flow in a circuit and also act to reduce voltage levels. Resistors are in very common components with many uses. Resistors are required on LEDs to limit the voltage across them!
		Capacitor Capacitors store charge in a circuit, and can be used to smooth an electrical signal. There are two different types of capacitors pictured to the left: electrolytic and ceramic. They look different, but are the same type of component.
		Diode A diode allows electric current to flow only in one direction. In the schematic symbol there is an arrow pointing in the direction the diode will allow current to flow.
		LED (Light Emitting Diode) A diode that lights up when a current is passed through it. There come in various colors, sizes, and brightness levels. If you want something to light up on your project, especially in different colors, you'll be using LEDs.

1

all sides? Or what if it was getting too close to a wall but didn't need a 90-degree turn to adjust it? Through a lot of trial and error, we slowly took each problem as it came, adding more and more to our code(through mostly if-else statements) until we finally had a robot that could mostly run through the entire maze without getting stuck. The only problem was that the robot could not make a turn into a hallway on its left or right without anything in front of it, as all of our turn conditions required an object to be in front of it, which worked for every contingency in the maze except the aforementioned. Why could we not do the aforementioned? If we had it always take a right at any right hallway, or left at any left hallway, it would never take a forward hallway, and thus not cover the entire maze. But, as it was, the robot was not able to cover hallways that branched off to the left or the right and had no wall in the front. To solve this issue, we initially tried to put a random variable that would randomly choose a number, one or two, and depending on that, go forward or go left or right. However, the robot would run this command every time the wall conditions were met, meaning that sometimes, it would choose randomly a direction to turn when there was no open hallway, as in a corner, for instance, the way in front and to the right was, in fact, open, even though there wasn't a hallway, so it would turn 90 degrees right randomly and then get stuck. In the end, we decided to make a non-variable code to solve this issue, meaning that we predetermined movement for the robot that didn't rely on the sensors. We know the configuration of the maze in the last couple of days of the project, so we made a setup code that would find the first pokemon(which was in the special scenario where there was a right turn but no walls in front), and then move to loop the sensor code we had to find the last pokemon.

Once the movement was done, we moved onto the IR sensors. We had to make a program that would send IR rays from our robot out in front of it. When those rays were intercepted by a pokemon, it had to communicate with the pokemon in order to identify which one it was and correctly “tag” it, causing the pokemon to light up. TO do this, we made a complex script that set values to each pokemon's IR and made them variables in analog to use in our code. Once we had the variables, we made a code that would identify the pokemon, send a certain signal corresponding to the pokemon to it, and then turn around and move on. We made this part without too many issues, however, we ran into some issues with getting the IR sensors on the pokemon to be lined up with the robot, as the robot would approach the pokemon at any random angle. Also, since it relied on its sensors and couldn't “know” the pokemon was not a wall, we had to make sure it wouldn't just turn around once it got close to the pokemon before it could “tag” it. However, through trial and error of measuring the correct distances needed to tag the pokemon, we finished the program and it worked very well, albeit with the help of us pointing the pokemon at the bot when it approached the robot, which was allowed since all the robots had this similar issue.

With our microcontroller project, the goal was clearly laid out for us; get a robot, in this specific case a pre-built BoeBot, to navigate two mazes of opposing difficulty to detect Pokemon, then return back to the start of the maze. While the goal was simple on paper, the process to complete it had a couple of road bumps along the way. In order for our robot to detect the Pokemon in the first place, we needed to use IR sensors on our robot to send and receive a signal to the IR sensor attached to their respective Pokemon. We also had to implement two different driving modes for our robot; an autonomous mode, where the robot navigates the maze without any human control, and a remote control mode, where we use a PS2 controller to navigate the robot through the mazes. Finally, we necessitated optimization of our robot for getting the highest score possible through the maze.

While this particular project does not have a true real world application, the robot itself, as well as the technologies used to power the robot can be used to solve several real world problems. Generally, these types of robots are used to traverse locations deemed either too dangerous or difficult for humans to navigate. One of the clearest examples is for bomb defusal; robots can remotely defuse bombs without any need for humans to risk their lives. There are also places within the Earth, such as the deep sea, that humans cannot easily enter that robots can. This allows robots similar to the one we designed to aid in deep sea exploration, enriching our knowledge of the Earth's ocean floor. We can even explore beyond the Earth thanks to semi-autonomous robots such as rovers, collecting information on other nearby planets and moons.

In order to be able to start coding the robot properly to get autonomous and manual control working, we first needed to understand the core components of our BoeBot. As aforementioned, we used IR sensors to connect to the Pokemon; the robot constantly sends an IR signal for one of the IR receivers on the Pokemon to detect. Each Pokemon has a unique IR receiver, so when the Pokemon's attached IR receivers receive the signal, it will send back a unique signal that the BoeBot can recognize and identify as the specific Pokemon. In order to even get the robot to move, however, we need to give the robot instructions via code. Our specific microcontroller on our BoeBot is an Arduino Uno; the Arduino IDE (the Uno's primary coding platform) uses C++ to code the board. Both members of this team have experience in Java, but not C++. Fortunately, we just had to understand a little bit of differences between the two languages; after that, we were able to use our knowledge to be able to tell the BoeBot to turn, align, and continuously check for Pokemon, alongside other advanced functions covered in the design process. Finally, there wouldn't be any robot to program without some electronics to make it run. There are three servos attached to the board; two of them have complete 360° motion, and are used to turn the two wheels on the robot and provide it with movement. The servo with 180° motion is attached to the grabber located on the front of the BoeBot; we never ended up using this component for actually grabbing anything. Each electrical component was connected to each other via wires and a breadboard, which is a thin plastic board used to connect electrical circuits to each other.

Once we understood all of our components and their use cases, it was finally time to design our robot to navigate the mazes. Our utmost priority was getting our autonomous driving mode as smooth as possible; to do this, we needed to ensure that our turning was clean and made a 90° angle. Along with several situational movements such as aligning away from walls. After a few days of work, we eventually succeeded, and got a solid framework for implementing features from our base autonomous driving. After this was done, we were able to quickly create several functions to use for our BoeBot. We ended up spanning from meticulously refining basic functions like turning and aligning, to creating functions with more complex functionality such as getting unstuck or checking open pathways. However, a minor challenge was that our ultrasonic sensors would occasionally give out readings that were incorrect, so we made a function that would take three values from all three ultrasonic sensors and would return the median of these values. That way, the incorrect readings we were getting would be filtered out, which fixed quite a bit of our issues involving the robot turning unintentionally. We also ended up making a function to allow the robot to detect if it's stuck or not, and if so, it would back out and turn into an open direction.

After this was finally completed, we needed to be able to detect Pokemon. Here, we simply edited a pre-existing version of code used to detect these Pokemon to be compatible with our Arduino Uno. We ran into a bit of an issue thanks to us importing the wrong header file for the IR Sensors to work properly. This caused several bugs within our code, but luckily after it was caught, the IR sensors worked smoothly from there. Now, at this point our autonomous code worked well enough to run through both maze and detect Pokemon, so we were finally able to turn our attention to the manual portion of the project. Thanks to our strong base with our autonomous code, our manual code ended up being incredibly simple to implement. Our code for enabling the PS2 controller was already provided, so we only needed to add in the ability to detect Pokemon for it, and then it was fully functional. There was one change we wanted to make; how we were able to tell the robot detected the Pokemon. On our autonomous control, we wanted to make the robot turn completely away from the Pokemon, so it would continue the course on its own facing the correct way back to the start. On our manual code however, we still needed complete control of the robot, so we preferred having the robot open and close its grabber to signify to us that the Pokemon was detected. After that, the rest of our project time was spent further testing and debugging our autonomous code.

We ended up learning quite a bit from this experience. For starters, both of us have never worked with C++ before, so understanding another programming language allows us to further broaden our practical programming knowledge and possible applications. We also learned a bit about playing more to our strengths as part of a team; we communicated about what we preferred to do on the project, which allowed us to work more efficiently and better than we could have as one person. These learning experiences will allow us to become much more effective engineers in whatever field we end up choosing to do, and so we consider this project a complete success.

The RC Car is a product of 20 hours of engineering. It was pondered and stressed over. It caused overwhelming pride and pure hatred as it was cursed for its successes and failures. Overall, however, the RC Car was a valuable learning experience as we grew as scientists through our investigation, application, and perseverance through obstacles. Having a background in separate fields, Luke and I were able to provide alternate viewpoints of the projects as we divided and conquered through our labs, finishing them all in a day and a half. This cooperation was then brought to the main event as we brainstormed from differing opinions and found ways to agree, slowly furthering the robot's ability. As an electrical apprentice, I provided a prior understanding of Ohm's Law and wiring. Opposite of this, Luke had significant experience in programming in Java and transferred this vast knowledge into coding the robot. Thus, while I floundered in the identity of float commands, he confidently and accurately told both the robot and I how it was going to move. In this way, I was able to learn and contribute to a project far beyond what I had considered within my abilities.

In a world fast advancing into a technological utopia, the self-driving car is a flagship of the developing world. In this project, we were presented with a task, "Program a robot able to maneuver through a maze and respond to the world around it." I took this as a scaled-down representation of the mentioned self-driving car and therefore, was greatly intimidated by such a massive feat. However, my initial awe and intimidation grew into surprise as the RC Car came into being. As with all projects, the journey started with the robot simply moving forward, but as time passed, I saw as our combined efforts gave the pile of circuitry and machinery the ability to determine how far away its surroundings were, how to react to its environment, and how at each *turn* to have an instruction, so it was never without a solution.

As I had previously stated, I had some background in electrical as a whole but was completely blown away by the incorporation of coding and the sheer amount of information that I didn't know and needed to attempt a project of this caliber. From basic commands that the RC Car needed to understand to the decryption of the incoming data in a way it could be repurposed for direction, fascinated me as programming truly is another language. Through my public school experience, I was required temporarily to take a class on a foreign language. For reasons I don't even understand, I took french. Throughout this project, I found myself constantly reminded of this class because coding well and truly is an entire language. In the same way, I had to translate french sentences and phrases. This project pushed my mental faculties, and I had to quickly gain an understanding of the language of computers. For me, this was the most amazing part. This project of course was not all about coding though. We were given the opportunity to further understand sonar sensors, infrared light and photoreceptors, bar codes, volume dials, the composition of binary, and many other concepts that revolutionized my understanding of electronics. This introduction to electrical engineering gave me more direction than I have ever had, and now I can't see myself in any other field.

The design of this robot began by applying our obtained understanding of coding servos in the language of "C" and implementing this knowledge by instructing the robot to drive in various directions. This was further developed by using the mounted sensors on the robot to have the robot react to obstacles in front of it and adjust its movement code accordingly. The first aspect of environmental reaction was programming the

sensors to detect obstacles approaching from ahead. Once it had determined there was a wall ahead of it, it measured its distance to the walls on each side. Using this measurement, the robot was able to turn in the direction of the open corridor. Once this potential event had been coded, we moved onto keeping the robot in the center of the path as it traveled down the hallway. Due to the fact that the robot prior to this had only been instructed to be aware of forward obstacles, we had to make it aware of dangers to the side. Thus, in order to prevent it from slowly running into the wall at a diagonal in a long hallway, we had it slow down one wheel and speed of the other to compensate for the original trajectory, causing the robot to veer away from the incoming wall and directing it back to the center. In more numerical terms, we had it turn when the front sensor detected the wall was less than 20 cm and the side sensors had room of at least 30 cm. For the directional correction, it began adjusting its path when the side sensors detected a wall less than 8 cm to each side. Following this we realized that no matter what we did, the robot often would take a turn and because of the protruding prongs, would get caught on even the smallest cracks in the wall. To solve this, we had the robot understand if it read the same value (if the value remained the same for a significant amount of time it meant it wasn't moving and based on the code, the only reason it wouldn't be moving would be if it were stuck) from the front sensor that it should reverse for 1 sec, so when it drove forward again the sensors would detect something and correct its direction. Unfortunately, this is where everything fell apart. For some reason as soon as we began implementing this code, the rest of the code began to malfunction. On this first day, we were able to make the back up function work a few times, but for the few instances this code helped with, it created many more issues. Furthermore, when we came back the following day (the day had ended just as we wrote the code for the back up function) the sensors had seemingly stopped reacting to obstacles and would constantly run into walls or not correct its trajectory as it rubbed diagonally into walls. This left us baffled for some time as we went on a long and frustrating troubleshooting process. We initially checked the sensor values through the live feed and ensured the sensors were reading properly. Following this, we ruled out the order of the code as the issue by moving around various lines of code, so they wouldn't interfere. With no prominent solution presenting itself after hours of confused, angry problem-solving, we cursed our new addition and removed the backup command. To our dismay, however, this solved nothing, and we encountered many of the same issues. For the next day or two, we asked for consultations, proof reads, and lots of help. We made little progress in this time as we changed different values, added and removed delays, and even used leds to signal when certain codes were running in order to eliminate variables and possible errors. By Monday (7/26), we had had enough. We wiped our lines of code and started rewriting. This time, we went step by step, testing each piece to remove errors and with experience, the process went much faster. We still had a few issues with the correction factors but it went far better than the previous attempts.

Overall, the RC Car performed above our highest expectations although personally mine, at least, were fairly low given my lack of experience with fully automated vehicles. It performed very well as it was able to finish the maze and complete all the objectives we set out to accomplish. I'm sure with more time and greater understanding, the robot could be developed much farther and gain more

precision and a plethora of responses to a wider range of possible outcomes, but as far as exposure to the field is considered, the robot was a great way to learn about programming, and electrical engineering as it showed me how much is possible with robotics and gave me a sense of pride over what we had managed to create. In this way, our creation not only met but significantly exceeded my expectations.

Watching and participating in this development gave me a new appreciation and admiration for the field of electrical engineering and engineering as a whole. The resources provided to me by Rose-Hulman University gave me the tools I needed to explore such a fascinating field and forever change my career outlook. What started as a reluctant arrival as my parents forced me to participate in the program, became an eagerness and joy for college life. I feel like I'm beginning to understand why the students and faculty I have spoken to love it so much here. It is a place of great education and gave me the opportunity to learn so much about my goals and my future.

Introduction

Humans have always been obsessed with doing the impossible. Especially when it comes to transportation. Driving, flying, space travel, drone surfing, you name it, we've tried it. However, we have yet to create a device that allows a person to walk across water using only their own two feet. Our team has taken on that challenge, despite the many problems that must be overcome.

Problem Statement

Our team was challenged with finding the fastest and most efficient way of crossing a lake using only the power of our feet.

Background

Our project relied heavily on the basic principle of buoyancy, an object's ability to float in water. Before creating our product, we had to make sure that it would generate enough upward force to push our user out of the water. We calculated whether our design would float by using Archimedes' principle. Archimedes' principle states that every object has a buoyancy force that is equal to the weight of the displaced liquid. By measuring the volume of the buckets, we were using, we were able to approximate the total upward force of both our shoes and our walker. Our walker should be able to hold 160 pounds, while the shoes should be able to hold another 300 pounds between them. This principle validated our design, demonstrating that our boat would continue to float even with our 160-pound user on top of it. We then proceeded to prototype our idea, and quickly learned that when something works theoretically it rarely works in practice.

Design Process and Results

Our design process began with a long brainstorming session. We tried to generate as many ideas as possible, regardless of how unusual or impractical they sounded. Our first idea was to build a contraption that required the user to be in a plank position on the surface of the water, propelling themselves across the lake by performing mountain climbers. As we started creating a more detailed design of this model, we realized that it would be too physically demanding for the user to remain in this position while keeping their balance. For these reasons, our team decided to create a design that would utilize the stability of a walker and allow the user to remain upright.

We chose to create the walker out of $\frac{3}{4}$ inch diameter PVC pipes. PVC, or polyvinyl chloride, is a type of plastic that is both durable and flexible, which allows it to easily support the weight of any potential user. Many materials would rust or crack after extended contact with water, which would ruin the structure of the walker and cause it to sink. However, PVC is specifically designed to work as piping for sprinklers, plumbing, and laboratories. It can remain in water indefinitely without losing any structural integrity. To increase the buoyancy, we decided to add multiple watertight 5-gallon buckets to the bottom of the walker. Our heaviest group member is 160 pounds, so we figured that 4 buckets would be enough to carry that weight. In addition to the walker, the design included two 18-gallon plastic storage containers strapped to the feet of the user to increase buoyancy and stability by evenly distributing the weight.

After finalizing our design, we moved on to constructing a prototype of the walker and the shoes. We had to decide on the dimensions for the walker before we could buy the PVC pipes. PVC pipes are ten feet long, so we originally decided to make the walker five feet wide to maximize the PVC usage. This was also wide enough for the shoes to have full mobility without running into the walker. We also planned for the height of the walker to be around four feet tall so that the user would not have to bend over to support themselves. However, when this frame was constructed, it was too large to easily balance on so the dimensions were modified.

When we began testing our prototypes, we immediately began to run into problems. During the first test, our user lost his balance and fell into the water. This was caused by several factors, but the one that caused the most damage was the uneven distribution of weight on the walker. We added buckets to the walker to achieve an even weight distribution given that the user would be perfectly in the middle of the walker. However, during the trial, the user leaned forward while walking. This caused the front of the walker to sink significantly deeper into the water than the back, which resulted in the walker rocking back. The user immediately lost his balance.

During the second test, we learned that the plastic storage bins were not as sturdy as we initially thought. As soon as the user went deep enough into the lake to no longer be touching the bottom, one of the plastic storage bins cracked, causing the storage bin to become completely submerged in water. Our team had accounted for the total downward force on the plastic storage bins, but we did not account for the inward force caused by the pressure of the surrounding water. This pressure was much greater than we had anticipated and caused the catastrophe to occur. Our solution to this problem was to find storage bins with thicker walls. These newer, sturdier bins would stand up to the pressure of the water without cracking, even if they were heavier, bulkier, and more difficult to walk with.

However, as we tested for a third time, our team discovered that the reinforced storage bins were too big to fit inside of our walker. We had designed the walker for the smaller plastic storage bins, so these new bins were not compatible with the walker. Our team realized that none of the storage bins we had would work with the walker, so we decided to try making the shoes out of foam. The foam shoes needed to be able to support the weight of the user, so we glued three layers of foam together. Our team taped the same fasteners we had used for the reinforced bin trial to the pieces of layered foam and tested for a fourth time. The fasteners were not secured to the foam well, which caused them to pop off immediately. While it was difficult to balance without a way to secure the shoe to the user's foot, we were able to determine that the foam was not buoyant enough. We decided to add more foam layers to the shoes and to screw the fasteners into the foam. The extra layers will increase the buoyancy and the screws will help the fasteners stay attached to the shoes during the test. We then tested for a fifth time. The new layered shoes were able to keep the user afloat, something the previous designs had failed to do. However, the fasteners were not secured perfectly in the middle, so the shoes rotated in the water and dumped the user into the lake. Securing the fasteners perfectly in the middle will increase the rotational stability and hopefully keep the user out of the water.

Reflection

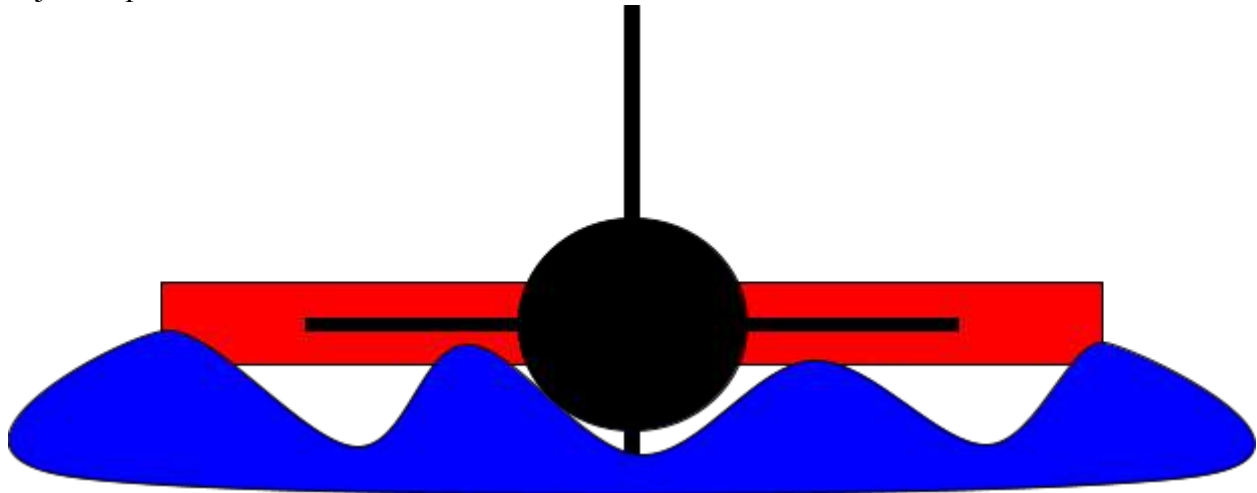
During this project, we spent a lot of time brainstorming. This shortened the time we had to build the project and test it, which made it difficult to fix the problems we discovered in our design. We should have put more thought into the materials we used to build our project and made precise calculations to determine how much weight the materials would hold. Making precise calculations would have allowed us to perform less failed trials, as we would know exactly how much buoyant force would be present. While we did an excellent job working together and collaborating, our team struggle to remain organized. Our workspace was never tidy, which made it difficult to find the materials we needed when we required them. In addition to not having an organized workspace, we forgot to record all our original design ideas on paper, making it difficult to follow our design process and see how much our current model changed.

Citations

Potter, Merle C. et al. *Mechanics of Fluids, Fifth Edition*. Cengage Learning, 2005.

Crowe, Clayton T. et al. *Engineering Fluid Mechanics, Eighth Edition*. John Wiley & Sons, Inc, 2005.

Team Name- IFBB
Group #29
Jacob McCuaig
Bailey Zimpleman
Jonathon Stadler
Arjun Gupta



Introduction

For this project, our group decided we wanted to implement mechanics into our design for the project “walk on water”. Other groups strayed away from this and wanted to design their flotation by connecting it to their feet. Our group wanted to challenge ourselves by using bicycle parts to act as the power for our raft. The process of designing and building was not very thought out but we did a very good job by “eyeballing” the project most of the time and we would not be where we would be if we did not have the assistance of Dr. Mueller and Kevin. Even though our group had our fair share of hardships, we made it out alright.

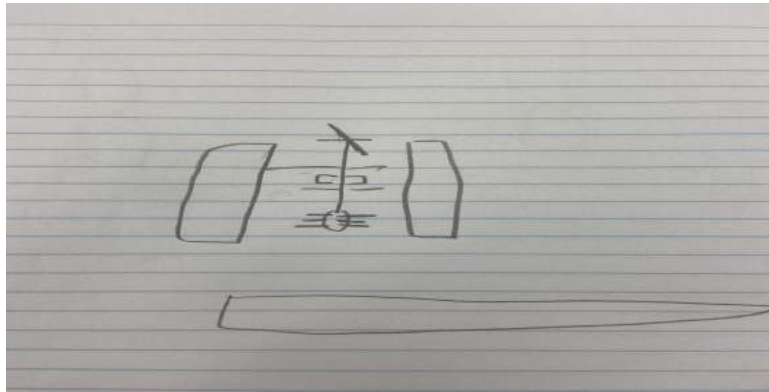
Problem Statement

While building our boat, we were making massive strides in completing our build until we had to build our source of power. This set us back because we had to design how we would connect the pedals to the axel by use of a chain. We grabbed massive amounts of chain and decided which size to use. While we were connecting the chain we realized that we had no master links to use. Our last ditch effort in connecting the chain was using a chain separator tool to connect the links together which is not what it was designed for. By the time that we were done, our tool had broken.

Background Information

Our background information is rather short because our team decided to build as we go instead of going through an extensive design process using theories and math to determine if our raft would work or not. The most important part of our calculations, however, were determining if our jugs would float or not. We originally intended to use five gallon jugs as floatation but we came to the conclusion that four gallon jugs could sustain a huge amount of buoyancy for a slightly cheaper price. For our sprockets, we intended to have a gear ratio that would create a lot of torque to go to the paddles since we did not want the driver to stop half way in the middle of Speed lake from being tired.

Design Process



For our design process, our original design was to attach pontoons to a bicycle to get across Speed lake. This design was then found out to be very hard because we didn't want to waste our time and resources with the pontoons. Arjun then made the idea of using water jugs as the flotation and making a paddle boat. Everyone agreed to the idea and began the official design of the paddle boat. Because we knew that the raft would be very heavy, we did the math and found that the average four gallon water jug can sustain forty pounds of force in the water before sinking. Next the frame was built and the water jugs were installed. Our original design was made for our lightest team member, Arjun, to pilot the vessel. Unfortunately, Arjun had some family troubles and had to leave which left our second lightest team member, Jonathon, to be the next captain. When we finished the frame portion of the vessel, we wanted to test out how well it would float in the water because the boat weighed around sixty pounds. When we put it into the water for the first time, it was a success. Even though it floated, we thought there was room for improvement since the boat did not float as high as we wanted it to. We soon added more jugs to the bottom and began to design the mechanical portion of the boat. First to be installed was the pedals and then soon followed by the seat and axle. As we continued building, the challenges kept getting progressively harder. The rear sprocket was a huge pain since we had to attach it to the dowel, but with the help of Kevin, we were able to make it work and were able to achieve great results.

Results

Our team did not intend for our raft to be as successful as we intended. Every time we tested our raft, we observed that we made massive improvements than the previous test. Our second and third tests could be considered failures to some people because our chain did not work each time, but to us it was proof that we were one step closer to success. When we finally fixed everything, it allowed us to actually move freely in the water. It exceeded our expectations in many areas except how fast it would go. We wanted the raft to go at similar speeds as a normal paddle boat would go but we could achieve that due to our dowel not strong enough to sustain great forces. If we were to upgrade our dowel to a metal one, we would definitely go way faster than what we are going now.

Reflection

Our team has come to the conclusion that this project tested our problem-solving skills and our ability to work as a team in a very positive way. The project brought our team closer together and allowed us to have fun while building it. Though we did have instances of injury and sadness from the loss of a teammate, we were able to bring each other as a team. This experience is something that we will never forget and would recommend anyone to experience.

Group 30

Emily Kroettinger

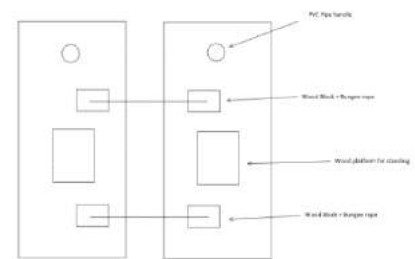
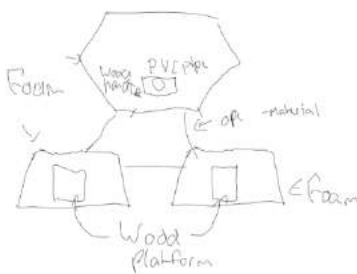
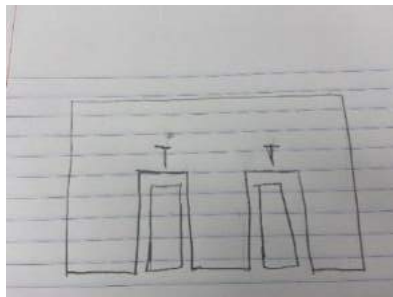
Alex Dietrich

Tewodros Holmes

The Water Strider

Long has walking on water been a fascination of humanity. As far back as the writing of the Bible, people have thought of such an act as mythical. Now, we, like many others before us, are taking on the task of creating a craft capable of taking us across a lake, allowing us to walk on water. We were tasked to build a device that would allow us to “walk on water.” The device had to be human-powered and ridden by one person.

Some of the concepts that we needed to use for this project were buoyancy, construction, water mechanics, and the design process. We used buoyancy to confirm that the type and amount of material would hold the weight of a person. We utilized construction because we needed to figure out the best way to put together all the materials we used along with the best materials to use. Additionally, we needed to consider water mechanics when building our skis as it was necessary to ensure that the way the skis were shaped would allow us to move forward. Finally, we needed to consider the design because we needed to make sure that the skis were designed in the most efficient way possible for us to complete our goal. Originally, we were not thinking much about how comfortable and easy it would be to walk on our device. After learning about designing with empathy we changed a few pieces of the design to better facilitate comfort in operation.



Our design process began with brainstorming. As a large group of 11 people, we threw out our ideas and discussed the basic foundations of different solutions we wanted to build. The three categories of ideas we created were a pedaling platform, a board someone would lay on horizontally and push themselves across the water, and somehow someone could walk vertically across the water; groups separated based on who wanted to build which design, and our group chose to build the one which would allow someone to walk vertically across the water.

The sketch we started with was a w-shaped wood board with a place for our feet that would be used to walk across the water (shown on left). In our group, we tried our best to figure out how to use this design for our project by creating sketches and rudimentary potential designs

for it. After a while, we realized we were constraining ourselves and our ideas to the first concept we had seen, and subsequently, we tried to see if something else would work better.

While we were not entirely certain what to build, we went to the supply room and looked for something that we could use. We found a large, canoe-shaped piece of foam. We figured that we could cut the foam in half and use it as two skis to walk with. At first, we were still planning to have a “walker” as a separate foam piece in front of the two skis. However, when we got into more detail with the planning, we realized that it would be difficult to keep the skis and the walker close to each other while walking.

We then changed our design plan and decided to attach PVC pipes to the skis themselves to hold onto while walking rather than using a separate foam piece as a walker. We also connected the two skis together with bungee cords and hooked them onto a wooden block we cut which was stuck to the foam with adhesive spray. The last step was using the same adhesive spray to put a thin wood plank on both skis where our feet would be placed to help with stability and reduce damage to the foam.

Our first test on Speed Lake did not go as expected. We had to take our skis through the building and out to the lake. On the walk, the poles were unstable, and we put a lot of effort into keeping them steady, which was the first warning sign. When we finally reached the lake, we had to figure out how to get it into the water and then stand on it. We slowly lowered it from the dock onto the water so a teammate could carefully lower himself onto the wooden panels on the skis. He was immediately unsteady even when he was sitting and could only maintain standing for short periods without falling. The skis kept turning around and tilting outwards, causing the bungee cords to break off. Eventually, our teammate lost his balance and fell into the lake. The project upturned, we lost control of it, and the PVC pipes fell into the water. We would have lost them, but our teammate thankfully managed to retrieve the fallen pipes. We picked up our broken, soaking project with the help of some other campers, and brought it back to start working on a way to improve our design. This test was not successful, but it was only the first one, and after this we improved our design and tested it again.

The second test of our skis went much better than the first one. On the second test, our teammate was able to stand on the skis and keep his balance. This balance stemmed from the pieces of foam that we added onto the sides of the skis to prevent them from leaning outward as they had done during the previous test. The main problem this time was that we could not move our skis forward because they did not have anything underwater to push the water so that we could walk across the lake. However, this test was an improvement from the last, which we were very relieved to see. After this test, to improve this design, we added more foam on the bottom of the skis to help us move forward and to add a little bit more buoyancy on the bottom so our teammate could get on top of the water a little bit more. We also attached the PVC pipes differently by gluing caps in the skis and putting the PVC pipes in the caps, so they were tighter. Another improvement that we made was to glue but also screw in the foam and wood on our skis so it is more guaranteed that they will not fall off.

Our third and final test of our skis also did not go as planned. The flotation was a little better than the first and second tests, but there were also a lot of problems with the skis. We were still not able to move forward to walk across the lake. Also, the PVC pipe caps did not hold the PVC pipes in well like we thought they would, so the PVC pipes loosened and eventually came out. The foam on the bottom of the skis helped with the flotation, but they too eventually came off; the screws did not hold and ripped through the foam. The improvements we decided that we needed to make to make our skis work were that we needed to add more foam to the bottom, shaped to help us move forward across the water. We also decided that we needed to somehow strap our teammates foot onto the skis so that he would be able to lift his feet up the water, which would hopefully help him move forward. Although the project did not accomplish the goal, we still learned a lot through the process.

While working on our water strider project, we gained many valuable insights that will help us in the future. At first, we were very conservative with our ideas and sketches, and constrained ourselves to the first concept we produced. After some struggle, we realized that it would be more fun and enriching to try to produce an out-of-the-box design. This realization was also helpful in developing our own methodologies that we will use in our future engineering pursuits. We feel we are more comfortable with producing inventive and avant-garde designs, especially since it allowed us to create a more effective design. We learned how to work better as a team because we all had different pieces of the whole that each of us had to make to complete our final project. We also learned how to go through the design process to create one big final project. We gained experience in breaking down mistakes to improve upon them, as our many trials gave us much to work with. Finally, we began to work better as a team through this project by combining ideas to let us create our ideal product.

Introduction

Speed lake is a contaminated, man-made, algae filled lake in the middle of the Rose-Hulman campus. Our project was to analyze the water quality, and construct a filter to best filtrate the filthy water. Our secondary objective was to construct that filter to be easily replicated, with limited materials, in third-world countries.

Problem Statement

In order to process the water from Speed Lake, we needed to create a filter to help remove the impurities and undesirables from the water.

Background Info

Through our experimentation, we determined that the turbidity, or clarity, of the water was one of the main components we would need to improve the condition and quality of Speed Lake's water. We tested for nutrients such as ammonia, phosphate, and nitrate, dissolved oxygen, turbidity, pH, and specific conductance. Testing for the nutrients allowed us to know what specific nutrients make up the water and the reason for that such as large amounts of phosphate which is often the result of goose poop. The dissolved oxygen told us how oxygenated the water is and the ability for plant life to grow in the lake such as algae. Turbidity told us how cloudy the water was and the measure of relative clarity in the water. The pH allowed us to test how acidic or how basic Speed Lake's water is. The tests we did to test the specific conductance of the water also allows us to report the electric conductivity of the water in units of microsiemens per centimeter. Once we conducted these tests on Speed Lake, we collected our data and began brainstorming ideas for our filter. Through our brainstorming, we discovered that it would be easiest to test just the turbidity and pH of the filtered water since those factors are most prominent in the water treatment process. We worked on a coagulation and flocculation lab that helped us learn more about the water treatment process along with the benefits of testing turbidity and pH. In the lab, we took a sample of the lake water and separated it among six containers, testing the turbidity and pH of each container. In five out of the six containers we added a coagulant in the form of alum, a hydrated double sulfate salt of aluminium, which destabilized colloidal particles in order to coagulate the particles into larger, more cohesive particles. The flocculation process then included particle aggregation and eventually particle removal. This particle removal leads to the removal of turbidity which is often correlated with the removal of pathogenic organisms such as Giardia and Cryptosporidium which can be dangerous to humans if found in drinking water. Once we added the alum to five of the six containers in the jar tester, we spun the stirrers in all six for about 30 minutes and let the water sit overnight before we tested the turbidity and pH of the new water. Overall, the turbidity of all six containers dropped an average of 5.88 NTU with the fifth container that contained 70 mL of alum dropping the highest at 10.83 NTU. Therefore, at the completion of the lab, the turbidity levels had dropped to a safe drinking level and we determined that the turbidity level of the water was essential to its water treatment. Our filter is only capable of removing impurities physically rather than biologically such as the nutrients of phosphates, nitrates, and ammonia. Therefore, the cloudiness of the water or the turbidity is most vital in determining whether the water is drinkable or not after the process of our filter.

Design Process

Our design process was a system by which we conceptualized, and tested each idea through trial and error. We consistently started with a structure of two empty water jugs, and a smaller large water bottle acting as the filter in between them. We then tested the filtering

process, trying out different combinations of materials. For each combination we tested the turbidity, and pH, which are the most important factors in determining whether the water is drinkable. For our first filter was a mixture of sand, activated carbon (charcoal), and larger pieces of gravel to keep the mixture down. While it did lower the turbidity of the water, it ultimately failed to filter the water to acceptable levels. Our second filter used an increased amount of gravel rocks, sand, and charcoal. This filter also included our first use of the ground-up corn. At the top another large piece of rock held the materials in place. This filter was an initial success as the rock kept the rest of the materials from floating away in the water. However, the filter did ultimately fail as the corn made the turbidity worse. Our third filter was made from the same materials of the second filter except for the cheesecloth along with a coffee filter. The filter also had slightly less of the materials than last time. This filter extremely lowered our turbidity, proving this filter to be a success, but we saw the corn as a limiter in future filters. For our last filter we would take that into consideration and remove corn completely. Therefore, we used our successful materials of charcoal, sand, and gravel to construct our filter. We also did not use a cheesecloth but instead used two coffee filters to drain any fine particles from the water. Ultimately, our fourth prototype was most successful using these few materials.

Results

At the completion of our project, according to our data we concluded that our fourth filter was our most ideal filter through the use of charcoal, larger pieces of gravel, and 2 coffee filters. According to all the prototypes' data, our fourth filter had the lowest value for turbidity at an average of 2.82 NTU which was the closest to drinking water's required turbidity value, which is below 1 NTU. With this information, we constructed a larger version of our fourth filter that was about double its original size. This new finalized filter had a larger surface area due to the use of a larger sized water bottle. The new filter's layer of minerals (gravel, sand, & charcoal) were also larger in depth and surface area because of the larger bottle's diameter. For our new filter, through the completion of several trial runs, we determined that separating each layer of minerals by a cheesecloth was a strong factor in maintaining a low turbidity for clean drinking. We also determined that it was best to use two coffee filters to hold all the minerals up in the filter and to act as a large safety net in catching any loose fibers from the filter that may remain in the water. Our final filter design also included large rocks at the very bottom and very top of the filter. The larger rocks largely serve to block any small minerals from escaping the filter and serve as weights on top to keep any minerals from floating to the top of the filter. Before we filtered the water through our final filter, we needed to make the lake water a worst case scenario and raise the turbidity to a higher level. Therefore, we added bits of clay and brought the lake water to an initial turbidity level of 26.43 NTU. The water was then ready to run through the filter. After having filtered the water through this new filter, we ended up with a final turbidity level of 1.91 NTU which was the closest turbidity value to the requirement for safe drinking water. The filtered water also had a final pH of 8.18 which was expected since nothing was being added to the unfiltered water to make it either acidic or basic. Therefore, the water kept about the same pH at around 8.

Reflection

All in all, despite being unable to reach the safe drinking turbidity levels of below 1, we were able to drop the original turbidity of the lake by about 23 NTU. This result was huge to our filter's success since our filter was primarily for filtering out the physical components of the lake water rather than the biological components. The research, the experimentation, and the construction of our filter all led to its ultimate success in the end. However, our mistakes and our

failures also led to the filter's success. We learned through testing that using corn in our filter was not successful at all and in fact made the turbidity of our water worse. After filtering the water through the corn, we ended up with a turbidity much worse than the original turbidity of the lake water. The corn was too fine to be filtered and tended to pass through the coffee filter and end up in the water making it very cloudy. We also learned that the cheesecloth was best used before the coffee filter because the loose fibers of the cheesecloth tended to be filtered into the water but if it was before the coffee filter, the coffee filter was able to catch any of the loose fibers before they fell into the water. Although we documented our filters decently and kept our research and data organized, we failed to watch the order of our materials more closely and eventually learned that it was better for our filter to be in the order of somewhat fine to the finest as it neared the bottom. We also failed to measure and keep track of the depth and size of each layer of minerals and our prototypes were never at a constant depth, giving us confusing results for our final filter. Despite our several mistakes, we did learn from them, and ultimately reached our goal of drinkable water with a huge drop in turbidity from our filter.