

A decorative graphic consisting of a thick black L-shaped bar. The vertical bar is on the left, and the horizontal bar extends to the right from the top of the vertical bar. The text is positioned to the right of the vertical bar and above the horizontal bar.

## **Ongoing Advancement of the Physics Toolbox**

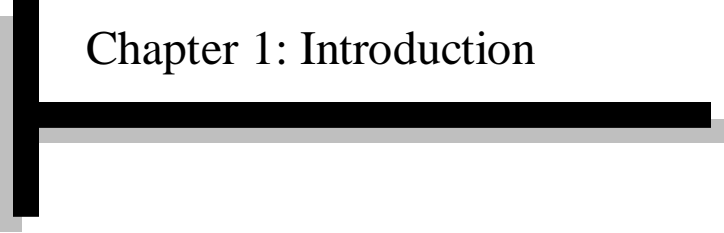
### **Physics IQP**

January 15, 2009

Kyle Pydynkowski  
Bennett Lessard  
Konrad Perry  
John McGinley  
Bennie Jones

## **Table of Contents**

Chapter 1: Introduction	Page 3
Chapter 2: Doppler Experiment	5
Chapter 3: Energy Lab	10
Chapter 4: Hot Wheels™ Video and Still Frames	19
Chapter 5: Applets and Their Instructional Functionality	29
Chapter 6: Organizational Team Management	42
Chapter 7: Conclusion	55
Bibliography	58
Appendix	59
1. Doppler Laboratory	60
2. Energy Laboratory	64
3. Power Point Presentation	71
4. Applet Java Code	72



## Chapter 1: Introduction

The inaugural efforts of the Physics Production Corporation have begun to compile an innovative physics toolbox to help incoming students develop a better understanding of basic physics concepts. The goal of this project was to reshape the first year laboratories of the WPI Physics Department. To achieve our goal we created new “tools” for the Physics toolbox, attempting to develop these new tools in a way that would be innovative and exciting for the students. During its inaugural term, The Physics Production Company developed two experiments, several web applets, and an introductory video for one lab experiment. One of the experiments was a proof of concept for the Doppler Effect. Using a setup of buzzers and microphones the Doppler Effect can be measured using a sound analysis program called Spectrum Lab. The second experiment was an energy lab designed around HotWheels cars. Students will combine the Vernier software with HotWheels equipment to come up with solid evidence of the laws of energy. To enhance this lab, an educational video was created in order to give students a preview of what they will see in their lab session. By seeing the video, students will have a clearer understanding of what they should be doing in lab and be able to use their time more efficiently. In addition to these tools, several assisting and simulation applets were designed for the toolbox. Posting the applets online will allow easy access for the students at any time. These tools should lay the foundation for future advancements of the physics toolbox by other IQP and possibly MQP students, as well as enhance the current state of the Physics Department. All of these new tools are far different than anything that has been seen by freshman taking the introductory Physics classes thus far. These differences should capture the attention of the students who otherwise would not be interested in an introductory level physics course.



## Chapter 2: Doppler Experiment

Kyle Pydynkowski

In the Ongoing Advancement of Physics Toolbox IQP, Kyle Pydynkowski, like Konrad Perry, was responsible for the creation of a Physics lab that could be used in any of the freshman Physics courses. The goal in creating a lab was to make it interesting to the students in order to make them want to learn and come to class. The lab should engage and involve all of the students in the Lab. Every student should have an equal amount of work to do so that all the students have an opportunity to learn. The topic of the lab should be interesting as well as aid the professors in teaching the students in a complimentary way to their lectures. Finally, the lab would be an original idea, in that it would let the students see the topic in a new light instead of seeing the same examples over and over again.

As the IQP began, the topic that was chosen was the Doppler Effect. However, after the original idea for the lab was thought out and tested crudely, a problem arose. The first idea for a lab was to use a frictionless cart, a position sensor, a speaker, and a microphone to demonstrate the Doppler Effect. Using a program called “Data Studio” to emit a sound with a single waveform, a speaker and a position sensor were both placed at the same end of a track. On the other end of the track was a pulley with a hanging mass on one end of the string and the frictionless cart on the other end of the string. On top of the cart was a microphone that was aimed at the speaker at the other end of the track. The setup can be seen in Appendix 3.a. With the sound from the speaker playing a single frequency, and the microphone analyzing it in a program called “Spectrum Lab,” the hanging mass and cart were to be released and consequently displaying the Doppler Effect in the “Spectrum Lab” data. Using this experiment, the students could have used the data that they recorded to find velocities and frequencies of the source (speaker) and

listener (microphone) from the equations that can be found relating to the Doppler Effect.

These equations include:  $f_o = f_s \frac{v \pm V_o}{v \mp V_s}$  where  $f_o$  is the frequency of the observer or listener,  $f_s$  is the frequency of the source, and  $v$  is the speed of sound, approximately 343 m/s at 20°C. The signs in the equation are determined by the directions the objects are moving and can be used in any combination. However, in order to obtain reasonable effects from the Doppler phenomenon, the velocities needed would be too high for the equipment and space that the students have in the lab rooms. Trying to make a cart move at a speed of approximately three meters per second in a small space, without letting it crash or get out of control, seemed to be almost impossible. This lab, although impractical to carry out in the lab space and with the equipment that the students have, would be an excellent way of portraying the Doppler Effect to incoming freshman.

After realizing that the original design would be unrealistic, a new lab needed to be designed. The solution would have to be a variation of the Doppler Effect described above. In creating another Doppler Effect experiment, the same general idea of the Doppler Effect is still apparent but we need to ask ourselves a new question: Is there another way to have motion in opposing directions? After some thought we realized this could be easily achieved with a rotating device. The idea for this lab came from the ideas of Astronomy and the rotation of a galaxy or a star. In astronomy, the light spectrum is used to determine the direction of movement, however in this lab; the sound spectrum will be used.

The new Doppler Effect experiment uses an AC electric motor, two sound sources (in this case buzzers with nine volt batteries), an Auto Transformer, and a microphone.

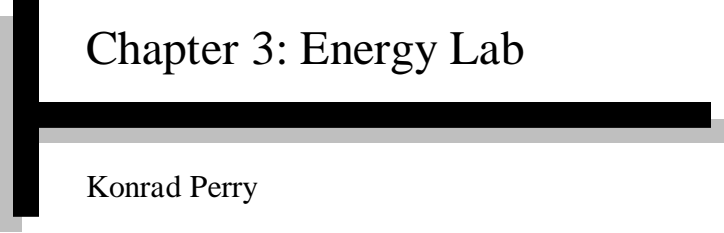
The setup can be seen in Appendix 3.b. In this experiment, the motor is equipped with a wooden blade, similar to a helicopter blade, with a buzzer and nine volt battery mounted on either end. A microphone and a computer with a program called “Spectrum Lab” installed on it were used to collect and analyze the data for this experiment. First, in a quiet room with no distractions, the “Spectrum Lab” program was started on the computer. After waiting for approximately five seconds, one of the buzzers was connected to its nine volt battery. The natural frequency of the buzzer was shown in “Spectrum Lab” as the data was collected. The spectrum that was shown in “Spectrum Lab” during this part of the experiment was a series of harmonic lines that represented the frequency of the buzzer. After sufficient data was collected, the motor was turned on and slowly began to pick up speed as the data continued to be collected in “Spectrum Lab.” During this part of the experiment the Doppler Effect began to be noticeable and take form. The lines in the spectrum from the buzzer began to broaden more and more until the motor reached full speed.

Again, the motor ran for a few seconds so that the data could be collected. The Doppler Effect was clearly visible in this portion of the experiment and was shown as a series of broad harmonic sections in the spectrum. When sufficient data was collected, the motor was slowly tilted toward the microphone to a ninety degree angle. As the motor was tilted, the Doppler Effect lessened and the spectrum became stronger towards a specific frequency. The motor was run for a few more seconds at a ninety degree angle. The broad sections in the spectrum got more narrow and had a more distinct and vibrant frequency in the middle of the sections. In the ideal case the Doppler Effect would be almost unnoticeable when the motor was at a ninety degree angle, because the buzzer’s



distance from the microphone would remain constant. This would be seen once again as a series of harmonic lines in “Spectrum Lab.” Finally, the motor was then shut off and the sections of the spectrum slowly narrowed back to a series of lines as the motor came to a stop. The complete results can be seen in Appendix 3.c and 3.d.

Due to the limited amount of time to finish the IQP, the Doppler Effect experiment was never put into a laboratory format and tested by other students. In the upcoming years of this ongoing IQP, a lab could be made quite easily from the ideas and findings of this experiment. A proof of concept has been established with this experiment and its success. In the future however, a louder buzzer would be an improvement to look into. A louder buzzer would allow the microphone to recognize the sound from further away. With the microphone further away, the change in the Doppler Effect would be more noticeable when the motor was tilted to a ninety degree angle. Also, with a louder buzzer and the microphone further away, the microphone would not collect as much noise from the sound of the motor running, as well as the noise from the wind that the blade creates. Another improvement that might be worth looking into would be a buzzer with a single waveform frequency. Having one distinct peak and no harmonics would result in a spectrum that demonstrated the Doppler Effect more clearly. Ultimately, through a creative lab, the students could learn about the Doppler Effect in a fun and interesting way.



## Chapter 3: Energy Lab

Konrad Perry

## **Background**

This IQP was designed to be the first of an ongoing process to improve the overall quality of the labs in the Physics department. Students who have already gone through the courses were to use their own experiences to come up with improvements for the labs. After much thought and consideration, energy was chosen an area that needed improvement. Every lab that dealt with energy always seemed to be handled with some type of springs. While some people may not have any problems with springs, after enough labs, they become tedious. At this point, the basic direction for the experiment had been chosen, while the details still remained.

The ideas for the experiment did not flow as had been expected, and after a while the search had been broadened to include springs. It seemed that no matter how boring springs were, they were going to be a part of this experiment. However, the more the topic was researched, the more options and creative ideas seemed to appear out of nowhere. There were ways that a spring could be incorporated into a lab that made it exciting and interesting as opposed to the labs that Konrad Perry had experienced in the past. After all the research had come to an end, a lab was finally chosen. It involved attaching a spring to a cart on one end and a frictionless track to the other and measuring the spring constant, force, and acceleration after giving the cart a push down the track. This experiment would be different than previous labs because it would involve both students to ensure that the cart did not crash and it also would involve “high” velocities which would stimulate students’ interest. Although the groundwork had already been set, there were several roadblocks that were encountered almost immediately.

## **Spring and Cart Lab**

The idea of the spring and cart lab had included all of the necessary aspects that were required for this project, it was all there in theory, but the bugs had yet to be worked out. The first major problem encountered was the positioning of the sensors. Because the cart would be attached to the track via the spring, it was unrealistic to place any type of sensor in between the two simply because if the student happened to miss, or forget to grab the cart before it hit the sensor the school could be out a great deal of money. The next issue was that it seemed imprudent to let the car crash into the end of the track without any type of interference. After much consideration, solutions were brainstormed; however none of them quite satisfied all of the early expectations of the lab. As the writing of the lab procedure commenced, and was tested by the guinea pig in the lab, it became more and more evident that this lab was not what had been envisioned. It was starting to look like any other spring lab that had been attempted in the past. So, when shortly after the testing an idea about a non-spring based energy lab surfaced, it seemed much more appealing than what was happening with the spring lab. Once again, the student experimenter was questioned and asked what his thoughts were on both of the labs. The spring lab was deemed not quite as interesting and exciting as this new idea, and so the energy lab started heading in a new direction.

## **Hot Wheels Experiment**

Due to the fact that the spring lab seemed to become duller with each day that passed, there was a dire need for something new and different. In one brainstorming session with Germano Iannacchione, an idea came up to use flexible tracks and cars to model Kinetic and Potential Energy. This new lab design would mean moving on to something far more different than what we have done in the past. The physics department had never used anything like this before, which is one of the reasons that it was so appealing. However that also meant that none of the materials were readily available. After a few trips to the local toy shop though, this problem was fixed, and work soon began on the new project.

Two experiments were chosen for the final lab. The first of these experiments involves the use of photogates to measure velocity. The second experiment utilizes a force sensor to measure force and impulse. The track is set up in a gradual downward slope with its base 100 cm from where it peaks at 90 cm. There is one photogate set up where the track flattens out and another 40 cm further down the track. The student will choose 3 heights from 30-60 cm use their knowledge of energy conservation to calculate what the maximum velocity of their car should be. They will then drop their car from each of those heights three separate times. The photogates in combination with the Vernier software will measure the velocity of the car as it passes underneath them, giving the students a velocity under each photogate. The student will then calculate the percent error of the velocity to see how well the energy conservation laws are upheld.

In the testing and analysis of this section of the lab, the cars consistently have a velocity that matches the expected velocity within 6% at the first photogate. At the

photogate that is 40 cm further down the track, that error only jumps slightly and has a maximum of 7% error. Another important note is that at no point was there any major issue or problem in the photogate part of the lab. It ran smoothly from conception to execution, which is more than can be said for the second part of the lab.

For the second part of the lab, the student removes both of the photogates, and clamps a force sensor to the table at the end of the track. The student will calculate the momentum of the car and relate that to the impulse that the force sensor should measure. The student will then release the car three times from each of their three heights and have the car collide with the force sensor. The Vernier software will create a nice force vs. time graph on the screen which the students can easily integrate to obtain a value for the impulse.

However, in the testing and analysis of this section of the lab it seemed like as soon as one problem was fixed, another would appear. The first issue that occurred in the force sensor experiment almost had it replaced with another experiment. Every time that the car would hit the force sensor, the Vernier software would record an impulse that was far too low. After several different configurations of the force sensor including a bumper, a tape barrier on the front, taping the sensor to the table, and finally clamping the sensor to the table, a solution was found. The sensitivity setting on the sensor had been set to  $\pm 10\text{N}$  because of the light weighted cars. However they were exerting much more force on the sensor than  $10\text{N}$  and with the setting changes to  $\pm 50\text{N}$  things got better for a short while. The next problem was that the graph of the impulse was consistently showing a skewing error to the right. This was interpreted that the force sensor was somehow moving as the car was in contact with it. In order to fix this, a camera was borrowed from

the ATC and the collision between the force sensor and the car was filmed numerous times. The problem turned out to be not that the force sensor was moving, but that the track would move slightly as the car passed over it and would come into contact with the force sensor for a brief moment. This problem was easily fixed by taping the track down to the table more securely. Another issue was encountered when a photogate was attached to try and determine the exact speed of the car before it hit the force sensor. Vernier had constructed their equipment to operate at different sampling rates. In order to obtain an accurate result from the force sensor, the software needs to be sampling at its maximum rate of 1000 samples per second. In comparison, the photogates will only function when that software is sampling at a rate of 250 samples per second or less. As there was no way to reprogram the software, the students will be forced to use one of their measurements from the photogate portion of the lab. In testing to make sure that this value would be valid for the force sensor experiment the velocities were found to have an average error of less than 1%. Therefore there was no issue with simply using the measurement from part one. In actuality it simplifies the lab for the student because they will no longer have to analyze data taken from two different instruments on the same screen. In the analysis of all of the preceding errors quite a bit of structure was added to the lab. However the final two problems only caused massive amounts of wasted time.

As mentioned before, the force sensor would only work when the software was measuring at 1000 samples per second. This would have been all well and good if the software itself had functioned properly at that rate. At one point, every single time that the force sensor experiment was run, the Vernier software would record its data and then promptly close. All that it gave was a cryptic error message. After corresponding with a

technician from Vernier a newer version of the software was installed on the computer and it was assured that the sampling error would not occur again.

While the sampling error never resurfaced, an even bigger problem came to light. Every time that the experiment was run the software would return a different impulse value. While sometimes the expected value would show up on the screen, the vast majority of the time the data would be wrong, and not a repeatable wrong either. The data was different every time. Seeing as this problem was unheard of, the force sensor was deemed to be broken and a replacement was found. However, shortly thereafter the exact same problem showed up again. There was nothing left to do but to delete the Vernier software completely and try again. A new computer was chosen for the trials and the old software was reloaded back onto it. After the first three runs it was clear that there was something wrong with the new software. The old program was running flawlessly at 1000 samples per second and it was returning the expected values. For the time being, all of the problems had been removed and the lab now had two functioning sections.

In addition to the lab procedure, another form of media was created to help the peak the students' interest. While the problems were being worked on, John McGinley was busy creating a short video that students could watch before, during, or after a lab session. The point of the video is to help prepare the students for the lab as well as give them something to look forward to before they get there. The video is posted on the physics web site and will be available to anyone who should need it

Once again it was time to bring in the guinea pig to test what had been put together and to time how long the lab would take the average person to complete. The first trial was a struggle at best. There were many questions that had to be fielded, and



much of the wording of the lab procedure had confusing parts to it. In time though the lab was streamlined and the questions became fewer and fewer. The time taken to complete the entire procedure dropped from 87 minutes to 35 minutes. Throughout the testing the student was asked several times if he had learned anything from the lab. Each and every time he was asked, it seemed as though he had learned something new. This was great news because the lab was designed to be exciting for the student while also teaching them a great deal about physics. The final copy of the lab was then attempted by other students who had previously taken the physics course and therefore were not completely new to physics. Completion was accomplished in under 30 minutes from both parties, and when asked what they thought about the lab both agreed that they had learned something new while doing a fun and interesting lab. The final copy can be found in Appendix 2.a along with data tables taken from various users in Appendix 2.b

### **Future Work**

While the Hot Wheels lab was completed, there are several directions that the physics department can take in the upcoming years. The spring and cart lab was never finalized, but there was a good deal of work that was put into it. It would not take much time at all to build that into a fully functional lab. Also, the usage of Hot Wheels cars and track should not simply stop at this lab. There are an unlimited number of configurations that the track can be put into and each one of those could be used as its own individual lab.

One example would be to configure the track into a gradual sloped “U” shape. At the bottom of the “U” would be a force sensor. The car would be released from the top of one of the sides and begin a oscillating back and forth. The force sensor would measure the component of the force in the vertical direction and the students could compare this to another oscillation or even some set of expected results. The possibilities with the cars and track are endless. And when paired with video and applets as has been the case with this lab, they should be that much more beneficial to the students.



## Chapter 4: Hot Wheels™ Video and Still Frames

John McGinley

## **Overview**

A picture is worth a thousand words, therefore, a video is worth a million. The use of still frames as well as video to highlight laboratory experiments is an extremely powerful tool. Using a video to demonstrate what is to be done during the laboratory session before the students arrive is not only beneficial for the students, but the laboratory instructors as well. If a video is well designed and the information is presented in a way that sparks the interest of several students before they arrive at their laboratory session, it is almost guaranteed that the learning process will be more enjoyable and informative.

## **Problems/Solutions**

Several problems arose during the creation process of making a video for the Hot Wheels™ laboratory. The problems were not only hardware, but software related as well. First and foremost, the camera that was used for the video was a Sony hard drive camera. This camera was the most convenient choice because it is provided by the Academic Technology Center (ATC) and is costless. Unfortunately, Sony's choice of an "MPEG-2" format is not readily supported by all systems. There were several options in resolving this problem. To accommodate Mac, one has to upgrade to QuickTime Pro. With the purchase of QuickTime Pro the MPEG-2 format is able to be used with iMovie. The cheaper solution was to download the correct codec packages off of the Internet to make Windows Media Player display MPEG-2 files. Once the video could be played on the computer, another problem arose. Even though Windows Media Player would display

the video clips, Windows Movie Maker (the video editing software that Windows provides) would not display the videos. To solve this problem a codec package by the name “K-Lite” was downloaded. The K-Lite codec package contained the correct codec for video playback for Windows Movie Maker.

Another problem that arose during the creation process of the video was lighting in the laboratory room. During the video, several still frames are pieced together to assemble the track without having a person in the shot. While doing this, the camera was accidentally turned off causing the final pictures of the setup process to be a wider-angle shot as well as have a different light exposure. To ensure that this would not happen again, all still frames were taken at the same time.

The backdrop for the video was also a problem during production. The laboratory room has windows, air conditioners, books and so forth that could possibly lead to a lack of focus on what is actually being presented by the footage. Any item that was not pertinent for the video that was captured during the shoot serves as a distraction. The first take of the video included such items as an air conditioner and school books in the background. Though this does not seem to be huge problem, it has a big effect on people that are less interested in the material that is being presented. For any video created in the future, a neutral background such as a sheet could be a simple solution.

### **About the Video**

The Hot Wheels™ lab was randomly chosen to be the inaugural video made. Video has been found to be a powerful tool. Having a video that can be watched before,

during, and after the students spend time in the laboratory could help to provide a better learning process. When the students watch the video before entering the laboratory, it should provide a quick overview of what is going to happen. Giving the students an exposure to the materials that they will be dealing with before they get into the laboratory will allow for the students to readily recognize if they are missing something important such as equipment or how to actually do the lab correctly. Providing the video for the students during the laboratory period also has a potential to improve learning. Being able to watch the video while the students are in the laboratory setting, they will be able to make sure that their final setup is correct before they begin to take data that would be severely affected with an incorrect setup. Finally, being able to watch the video after the laboratory period is over, the students will be able to use the video as a means of remembering what was done during the laboratory. Having a video will serve as a great reminder of what was done and how it was done. This will provide an opportunity for better answers to the laboratory questions that are not finished within the laboratory period.

An important underlying detail of the video for this laboratory experiment is the background music. The music that was chosen for this laboratory video is an instrumental version of “Fireman” by an artist named Lil’ Wayne. Using the instrumental version of a popular rap song will hopefully gain the interest of a variety of student. The student that this video is geared toward is the student that lacks motivation to want to go to their laboratory period and do the work. If a less motivated student can associate with a song that they have heard before, it will hopefully cause them to watch the entire video and become more interested in doing the laboratory experiment. If the

video is watched the entire way through before the student arrives for their laboratory period it will make the setup process more seamless.

The video begins with four still frames of one of the cars used in the laboratory. The next portion of the video scans from one side to the other of a different car. The purpose of this is to present material that would not be typical in an educational environment. Using footage in this manner provides a little humor because it's as if a very expensive automobile is being shown off. Beginning the video this way will help catch the attention of students that are not as excited about physics. If the student can relate with the video before any informational material about the laboratory is presented, it is more likely that their focus will increase and the necessary information will be remembered.

During the second portion of the video more of the important information is presented. This portion of the video shows the materials that are going to be used in the laboratory experiment. Every item that is used during the laboratory has its own section of the video. A still frame was taken and runs for a few seconds along with its name so the students will become familiar with the materials that are used for the laboratory. Making the students aware of what is used before they arrive will allow for an easier laboratory experience for student and instructor alike.

The next portion of the video shows the setup from start to finish for the photogates. Several still frames were taken and then pieced together to show the building process of the track. Each time a new section of track is pieced together or the photogates are added, a new still frame was taken. This was the best way to show the setup of the laboratory without having to show people in the video. Eliminating people

from the video could help to avoid from a distraction from the laboratory material. Using the still frames pieced together allows for a video showing the setup without involving other factors that could ruin the student's experience. Section four of the video is quite similar to the previous section. This laboratory experiment uses two setups, however, a minor change is made in transitioning from photogates to force sensor. The video again shows the setup of the track (in case a student does not finish the laboratory in the assigned time and has to go back to finish their work) from start to finish only this time putting the force sensor in place of the photogates. Again, the video is created using still frames put back to back to remove any distracting factors.

The concluding section of the video provides footage that will leave the students excited about going to their laboratory period. Similar to the beginning of the video where footage is taken in a fun way, the end of the video provides the same feeling. A close up is taken for each section of the track. The video concludes with footage that makes the audience feel as though they are actually riding down the track. The camera was placed at the top of the track and then brought down the track to give the appearance that it was attached to the top of one of the cars. Ending the video in this manner will leave the students with an excitement similar to the feeling they had at the beginning. It is important to spark interest not only in the beginning of the video, but to end with something that will provide fun as well. In doing this, more students will come to their laboratory time with more energy and excitement which will in turn cause for a better learning environment for the students.

Several different reactions were achieved when the video was presented to students. First of all, when this video was shown to a student that had never taken a



physics course during their time at WPI a little grin was on their face from the first note of the song playing in the background. As the video continued to run the grin turned into a smile and eventually laughter came from the student. When the video concluded the student was asked for feedback. The student said that they wished they would have taken a physics course because the laboratory experiments seem like they are much more interesting than any other that they had taken. The student also said that the video was a breath of fresh air because it was a new style of presenting educational information instead of reading a piece of paper or looking through a PowerPoint. This reaction was great to see because it means that the video was a success. Causing a non-physics student to become excited about the subject was the goal, and it was achieved.

The video was filmed in a fairly unique way. The major goal during production was to present the material without having to use people in the footage. To eliminate this problem still frames were used and pieced together. Using this method to create the video gives the appearance that the laboratory equipment was setting itself up and it also made sure that no extra distractions were being presented to the student. Mixing still frames with typical video sequences provides the student with two different sources to keep them interested. When the video freezes to a still frame, it is a simple way to show the student the importance of the material being presented and will cause them to pay more attention. Finally, the video footage was filmed from several different angles so that it was not a bland production. To keep the students interest, far away shots are used to show the setup process and emphasize how the laboratory is supposed to look. The video also uses extreme close-ups to give a better understand of how the track is connected and how the final product should look.

## Still Frames

Still frames are a useful tool for students when taking part in this laboratory experiment. At times, the laboratory sheet requests the students drop the car from a certain height. A simple picture was taken of the final track setup and has been manipulated to show from point A to point B is what is meant by height. Another still frame that was added to the laboratory write-up is when length is discussed. Similar to the height photograph, length is presented the same way. If the height and lengths are not done correctly it will adversely affect the accuracy of the data being collected and will make the laboratory experiment not worthwhile. Another part of the lab requests that the force sensor to be set at  $\pm 50\text{N}$ . A picture was taken close up of the force sensor to show exactly what is meant. The force sensor effectively has two settings, and if the incorrect setting is applied it is another way to ruin the data and affect the learning process of the student. Another still frame that was created shows how far apart the photogates are supposed to be from one another. It is a simple addition to the laboratory; however, it will ensure that a student who follows instructions through pictures rather than words will have the correct setup as well. The final still frame that is used for this laboratory experiment is of the photogates. On the inside of the photogate is a switch that has an up and a down position. The still frame shows two photogates side-by-side, one being the correct setup (with the switch in its up position) and one incorrect (the switch in the down position). This still frame is necessary just like the others because it will not allow the laboratory to be done correctly.

## **The Video as a Teaching Tool**

Using video as another way to present material could prove to be useful for the future. Giving the students another means of looking at material will help stimulate eagerness in a different variety of students. When a student is required to watch a video showing them what they are going to be doing in their upcoming laboratory periods will familiarize the student with pertinent information. Allowing the student to see all the materials and setups that are necessary will provide an easier laboratory experience giving the student an easier opportunity to obtain more educational information.

## **Future Work**

In the future of this ongoing physics IQP, the sky is the limit. Any lab that is being used by the physics department should also have a video to go along with it. Providing students with several different ways of learning and seeing material will allow the department to reach more students than it has before.

There are several things that should be done differently when creating videos in the future. First of all, the area that is being filmed needs to have a solid color used for the backdrop. An easy way to accomplish this would be to use a big sheet of some soothing color (possibly taupe or something that is not distracting). By using one continuous color for the entire background it will make sure that a student is not distracted by something that is not vital to the laboratory experiment. As it is now, the

video that was made for the Hot Wheels™ laboratory experiment has several different shades in the background. With this distraction removed, the video would be more suitable for a student that is not as enthusiastic with the subject.

The second improvement for future videos is a simple modification. As it stands now, the video that was created has a popular rap song for the background instrumental. If several videos were made providing different soundtracks it would allow for all varieties of students to enjoy the video. Using several different genres of music is a simple change. By muting the rap song instrumental and applying different genres such as rock, classical, country, etc. would allow the students to choose what they want to listen to as their background music. If the student can relate with the music playing in the background they will be able to become more excited about participating in the laboratory experiment.

A final recommendation for future videos would be to increase the amount of special effects used. Even though this is supposed to be a teaching tool, special effects could surprisingly have a positive influence for learning. For example, during an important part of the laboratory video if some kind of a special effect was used, it will instill the image into the students mind. By associating an important part of the laboratory with a special effect the student will have another tool to remember what is being emphasized.

To view the video, go to the following link:

<http://www.wpi.edu/Academics/Depts/Physics/Courses/Labs/Video-Folder.html>



## Chapter 5: Applets and Their Instructional Functionality

Bennett Lessard

## **Introduction**

As an alternative dimension to the laboratories created in the Ongoing Advancement of the Physics Toolbox, applets have been designed to assist freshman students with some of the difficulties of physics in a technological and interactive way. The goals for the creation and implementation of these applets coincide with my own interests concerning the Java Language. The main objective for this IQP pertaining to the applets was to create interactive applets and post them online. Explanatory text surrounding the applets is provided to clarify the use of the tool to provide the best functionality for the student. Accomplishments towards these goals include the creation of applets in two categories: assisting applets and simulation applets. The assisting applets created consist of three power of ten applets, two Greek letter applets, and a statistical analysis tool. Simulation applets created are the collision simulation applet, the projectile motion applet, and the Hooke's law applet. All the code for these applets is available in appendix 4.a through 4.i. Below is a description of how each of the applets works and a small portion about how the code was used to create the final result.

### **Assisting Applets: Powers of Ten**

The powers of ten applets are a matching style game where the user is presented with a stimulus in each of the three applets, either a power of ten, a prefix of a power of ten, or an abbreviation of a power of ten. Once this stimulus is shown, the user then has to match the equivalent values in the other two categories. For example, once presented

with a power of ten, the user has to match the corresponding prefix and abbreviation which are presented as a grid of buttons. If the match is correct, the buttons will be highlighted in green for about three seconds, and then return to their original color. Then the next power of ten to be matched will be displayed. If the match is incorrect, the buttons pressed will be highlighted in red, and then returned to their original color. Once all ten of the powers have been matched, the applet displays the amount of time it took for the user to complete all ten matches and the number of incorrect matches made.

To make this applet more useful, and so that the stimulus isn't in the same order every time, a random value generator is used. Java uses a function called `generator`. This allows the programmer to have the code generate a random integer value between two specified values. Based on the integer generated, the code determines a power of ten that has been designated to the integer value. This does not mean that the integer value generated is the power of ten, only that each power of ten has an integer value pre-assigned to it, and when that pre-assigned value is generated, the corresponding power of ten is displayed. This applies for the other two versions of the power of ten applets, where the user is prompted with the abbreviation of a power of ten, and has to match the corresponding power of ten and prefix, and where the user is prompted with the prefix and has to match the corresponding power of ten and abbreviation.

This applet provides a useful tool which can be used in a lab environment or as a study tool for a student. In a lab, this tool can be used to not only help the students learn the powers of ten that will be used throughout their physics classes, but also provides a fun way that students can compete against one another based on the time to complete the

matching. Because the time is recorded, partners can compete against each other in a fun way while providing a score that can be recorded as part of a lab report.

### **Assisting Applets: Greek Letters**

Similar to the power of ten applets is the Greek letter applets. There are two different Greek letter applets: Upper case Greek letters and lower case Greek letters. These applets are also matching games where the user is presented with the name of a Greek letter, and the user has to click on the corresponding Greek letter displayed in a picture.

The Greek letter applets use the same idea of random value generation to make the order of the Greek letters appear different each time. Java code determines where the user has clicked using a functionality called the mouse listener. When the user clicks on the screen, the code receives the x and y coordinates of the click. Based on these coordinates, the code executes a series of if statements to determine if the click was on the appropriate Greek letter. If the click coordinates are within the set parameters in the code, then the if statement is true, and a correct match is recognized. If the user makes five incorrect clicks in a row, then the code moves on to the next Greek letter. This eliminates the possibility of the user just clicking quickly on each letter if they don't know which Greek letter is the match. Once the user has matched all 24 Greek letters, the applet displays how long the user took to finish matching all the letters. Since Greek letters are common place as symbols in not only physics equations, but in all types of



mathematics, sciences and engineering fields, it is important to know what the name of each letter is. This is why this applet is particularly useful for engineering students.

### **Assisting Applets: The Statistical Analysis Tool**

An applet that has endless possibilities for future work is the statistical analysis tool. This tool is used to evaluate data entered by the user. The tool plots the data in a scatter-plot as the data is entered. The code auto-scales the scatter plot as the data is entered based on an algorithm that uses the maximum and minimum values of the entered data. The maximum and minimum values are constantly updated each time a new value is entered by the user to ensure correct scaling. Once the user has finished entering the data, the 'analyze data' button is pressed, and the applet displays the standard deviation, average, minimum and maximum values, as well as the number of terms. Another option for graphical analysis besides a scatter plot is a frequency histogram of the data. This is done by hitting the 'histogram' button. This plots the histogram where the scatter plot is displayed. Not only does this tool have applications for physics analysis, but for general data analysis as well. The code can be changed to add whatever additional analysis desired.

### **Simulation Applets: Hooke's Law**

A simple applet that can be used with a spring constant lab is the Hooke's Law applet. This applet requires ten pieces of data; five forces and the five corresponding

distances from equilibrium that the spring stretched based on the applied force to the spring. Based on these ten data values, the spring constant is calculated and displayed for the user. This applet can be used to go along with a calculation done by a student in the lab environment.

### **Simulation Applets: Projectile Motion**

One of the two physics applets that deal with object movement is the projectile motion applet. This applet bases movement on inputs by the user. These inputs include an initial velocity in the x direction, initial velocity in the y direction, and the initial position on the y-axis. After entering these values, the run button is pressed, and an object is launched with the parameters entered. The object leaves its path of flight behind as it moves through the air. The applet also displays the current position of the ball while it is in flight. Along with this, it displays the velocity in the y-direction in the form of a bar graph, the time in the air, as well as the maximum height reached. After the first run of the object, the user can enter new parameters, and run the flight simulation again. The next time the code is run, the first path will be left on the graph, and the second object will be launched with the new parameters, and leave behind a second path as well. This will continue for four flights. After four launches, the first flight path will be erased, and the next flight path will be stored into the first path again, while leaving flight path two, three and four still displayed.

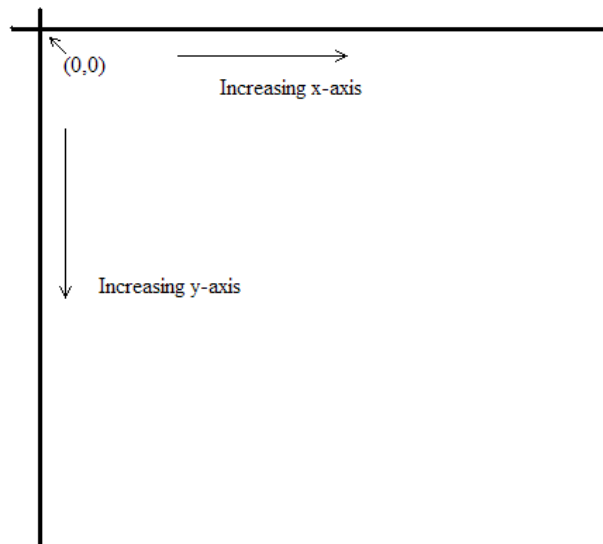
This applet has many ways that it could be changed to add additional functionality. These include adding air resistance, projectile motion with the parameters of angle and velocity in that direction, as well as additional flight path data storage.

### **Simulation Applets: Collisions**

The last applet is the collision applet. This applet simulates an elastic collision of two objects. This applet requires user inputs to determine movement of the two objects. The inputs consist of two masses and two velocities. The input masses are values between 500 and 2000 grams, and the initial velocities range from 0 to 25 m/s. The entered masses determine the display size of each of the two objects that are going to collide through an algorithm. The objects travel at each other with the input velocities, collide elastically, and then travel in the directions according to the collision laws of physics. The initial velocities and final velocities of each object are displayed below the objects in motion, as well as a plot of the velocities over time for both objects. This applet can be used as a supplement to a collisions lab, as well as an addition learning tool to enhance knowledge of collisions. Future work on this applet could include allowing the user to decide whether the collision is elastic or inelastic, as well as angular collisions and both positive and negative initial velocities. If the limitations on the masses and velocities turn out to be inappropriate for the use of the applet, they can be easily changed in the code to fit new values that work better for the application.

## Difficulties in Applet Design

Creating a useable collision applet that functioned according to the laws of physics was not a one step process. There were road bumps throughout the code writing process that often left the programmer stumped. At one point during the design process, the objects that were supposed to collide were reacting to a collision before they actually collided on the screen. Not only did the collision applet prove to have hidden difficulties, but all the applets provided stumbling blocks throughout the creation process. One of the most difficult issues that was found in the collision and projectile motion applets were the various axes used when programming simulation applets. Java has an axis for applets where the upper left corner of the applet is the point (0,0). The x values increase as the applet screen moves to the right, and the y values increase as the applet moves down as shown below. In an x-y plot, usually the y-axis increases in the opposite direction.



**Figure 1.0-Java Coordinate System**

The issue with the projectile motion applet is that the axis provided by java does not coincide with the axis that the programmer would want to use. The origin used by the

plot on the projectile motion applet is at java's coordinate (40,719). This makes the programmer's job slightly more difficult, because the physics equations that model the flight of the ball cannot be entered exactly as the equations work in every-day physics. This problem was solved by setting the origin of the acceleration plot equal to (40,719), and instead using the position equation:

$$position = x_{initial} + v_{initial} * time + \frac{1}{2} a * time^2$$

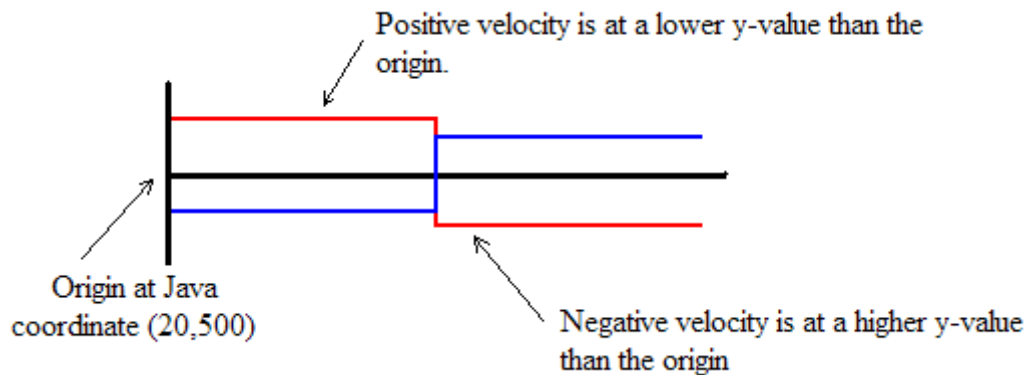
the code uses:

$$position = x_{initial} + v_{initial} * time - \frac{1}{2} a * time^2$$

The code subtracts the value of  $\frac{1}{2} a * time^2$  because the y-axis is inverted according to java.

The second axis issue that proved to be a stumbling block is found in the collision applet. This applet is especially problematic because there are four different coordinate systems in use at every moment. The first is the java coordinate system, where (0,0) is at the upper left corner. Both objects in the collision applet have their own coordinate system. The first object, object A, has positive movement to the right on the applet screen, while object B has positive movement to the left on the applet screen. This issue gave the programmer problems during the first stages of development. The equations for collisions in physics assume that if two objects are traveling at each other one has positive velocity and the other has negative velocity. The programmer neglected to realize that both objects velocities were being designated as positive in the code, when in reality, one was negative and the other was positive. Once this issue was resolved, the equations began to work as expected, and the objects were colliding elastically.

Along with these three coordinate systems, the velocity-time graph must also be taken into consideration. When time equals zero, the java coordinate value for the velocity-time plot is (20,500). A positive velocity will be shown above this point, while a negative velocity will be shown below this point.



**Figure 2.0-Velocity vs. Time Plot in Java Coordinate System**

Since java increases as the value moves farther down the applet screen, a negative velocity must be subtracted from the point (20,500) to appear as a negative velocity. For example, if object A is traveling at negative ten meters per second, then  $500 - (-10) = 510$  gives the point where the line will be drawn. This goes against conventional thinking where a negative value should have a lower y value, instead of higher value. Similarly, a positive velocity must be subtracted from this value to show the velocity as being above the origin.

Another issue that needed to be addressed is the difference in how Java is run on PCs and on Macs. A major issue found during the creation process dealt with the powers of ten applets. Macs show Java buttons in a different manner than PCs do. Macs don't show the background color of the button changing, which defeats the major functionality of the applet. This problem was fixed through making an alternative version of the code that works in a similar fashion to the original version. The Mac version, instead of

changing the background color of the button, colors the text inside the button. This color changes in the same manner that the buttons did in the original version, turning green when a correct match is made, and red when an incorrect match is made. In addition to making the color of the text change, asterisks are added to the text on the button to make the click more visible for the user.

The Greek letter applets and the Hooke's law applet both use pictures that are loaded onto the applet screen. When inserting a picture into an applet screen, the code creates an image that is the picture, and a media tracker which is another java function that pertains to image displays. Java is particular when loading images into applets. In order to load a picture into an applet, the picture file has to be in the same location as the source folder that contains the java .class files.

There are two different file types that a programmer in Java needs to work with when creating applets for online use. Programming is done in files with the .java extension. These files are where the code is written by the user. For this IQP, the programming was written in Eclipse, a program that can be downloaded for free of the internet. Eclipse takes the .java files which the programmer writes, and compiles them into the .class files which are used when inserting applets into the HTML files. The .java files are useless for this application, but are critical for the programmer and anyone who will be using the code for future applications. All the java code in the appendix is a copy of the .java files.

Posting the applets online proved to be a small issue that required a minimal amount of research. To post an applet in an HTML file requires a few lines of simple code. An example is shown below:

```
<Applet Code="Power10.class" width=450 Height=400>  
</Applet>
```

This lets the HTML file know which java class file it needs to run and the width and height of the applet screen to be displayed. Microsoft Publisher was the program that was ultimately determined to be the best way to create the HTML files. The lines of code for displaying the applets have to be inserted into the HTML file as an HTML code fragment under the insert tab on the tool bar. To make the java code run on the HTML the class files needs to be in the same file location as the HTML file in the computers directory. By using Publisher, it is easy to not only provide the applet on the screen, but also provide the necessary descriptive pictures and text on the web page.

All these applets can be used in different ways to provide learning for students. The assisting applets such as the powers of ten and Greek letter applets can be used by students on their own to learn the Greek letters or the powers of ten, or these applets can be added to an introductory lab for students to complete during a lab session.

### **Future Work**

Future work in the area of physics applets is unlimited. Applets can be designed to mimic the labs that are currently in use by the physics department and the future labs to be produce by the Physics Production Corporation. There are many other types of tools that could be created to help students with difficult areas of the basics of physics. Ideas for applets include vector math, significant digits, unit converters, and an applet that allows the user to pick an symbol that can be found in physics, and the applet shows



which area of physics the symbol is used and the equations where the symbol is used. As mentioned before, each applet that was created could also be altered to enhance their functionality.

## **Conclusion**

Applets provide a way to engage students in a technological manner. Simulation applets provide examples of physics concepts that support teaching in the classroom, and mirror physics topics that are covered in laboratories. Assisting applets provide students a way to learn an area that is a foundation of physics in a way besides just studying a sheet of paper. Since the assisting applets provide the time to completion, the applets can inspire friendly competition between lab partners and friends within a lab session or on their own, while still teaching students important aspects of physics. The Physics Production Corporation's goals concerning applets for online applications were met through the creation of the aforementioned applets. The process of creating applets and then posting them online for students to use and advance their personal knowledge of physics has been defined so that future members of this corporation will be able to further the effectiveness of the physics toolbox through assisting and simulation applets.



## Chapter 6: Organizational Team Management

Bennie Jones

Growing up a student hears all about biology, chemistry and physics; hears all about the lab work and the experiments, yet once the time comes to take these classes they no longer seem as appetizing as before. The older students become the less appealing these sciences became. It sometimes even gets to the point where they no longer enjoy learning about them. It was around that time that this IQP was formed. The Physics Department recognized a need for change. A few team members had been involved in physics studies in high school and furthered their knowledge during the first year at Worcester Polytechnic Institute (WPI). These experiences ignited the motivation to help change the Physics Department. At that time the group confronted Fred Hutson and Germano Iannacchione who also shared their passion for change and the ball started rolling.

The team needed to figure out what aspect of the freshmen experience needed changing. In dissecting what they knew about physics the team realized their knowledge was not as vast as they would have liked. At this point a light bulb triggered and everyone knew exactly what it was they could work with: the freshmen physics experience. Freshmen physics at WPI is as dry as it could possibly be. The team needed something that would motivate their efforts to create a product that would grab the attention and interest of their audience. This is when they decided to create an ongoing, interactive physics IQP that would deal with and concentrate on freshmen physics labs and experiments.

The task of designing these new labs and applets was then split into four different departments: lab creation, writing Java applets, video-enhancing stimulation, and organizational team management. Two members were assigned to lab creation and the

others divided the remaining three tasks. Konrad Perry and Kyle Pydynkowski decided that, though semi-effective, the current labs needed some changing. They remembered the lab work they did in their first year at WPI and felt as if things needed to be tweaked. Using such physics subjects as the Doppler Effect (a new direction) and Energy (revamping), enticing experiments were created that would hopefully bring life back to the department and grab students' attentions once again.

The next group was the applet department. Bennett Lessards' main concern was to create user-friendly applets that help enhance and reiterate the information in a lab. The applets help students understand different aspects of physics, from exponential growth to aspects of collision. With easy-to-use, interesting, eye-popping applets, the group hoped to teach the material and grab the attention of the younger students more than past applets have.

Next, John McGinley, with help from the team, created a video enhancing stimulator that students could watch prior to arriving at the lab. This video would hopefully grab their attention as well as get them enthused about arriving to class. This section approached this part of the project with an open-mind and many different ideas. If the video could somehow relate to the students by including things they found interesting, yet also be relative to the upcoming experiment it could get them excited before they even stepped foot into the laboratory. The student would anticipate a fun and educational experience which would then result in higher participation and overall higher moral within the classroom.

Finally was the organizational team management group. Bennie Jones was the main manager in charge of the group. He was put into place to help the other three groups

make everything user friendly and easy to understand. With no prior physics experience he provides an insight into what the everyday student understands and finds interesting. He also serves as both an editor-in-chief and helps to tie in all experiments and applets into the final product. That is where this section begins: the organizational team management stage.

The main concern and goal for this part was to develop an easy to understand lab that could not only be comprehensible to students with prior knowledge, but also the physics new timer. This section had its own influences, its own motivations, and its own concerns. Management needed to tie the other three sections together and then, after it is all said and done, reflect on the entire project with an unbiased and open-minded point of view.

Managements' part started off slowly. Ideas and basic concepts were out there, but without an experiment the experimenter could not carry out his duties. A week or so went by without any experimental procedures commencing. At the beginning of week two he was approached by the energy lab with an idea about a spring's constant project. This energy lab was the first experiment to show some promise. It was designed to show the spring constant by stretching out a cart to a certain length, calculating the amount of force that the spring would undergo, then putting these numbers into equations and calculating the spring constant. The applet that coincided with this experiment would then accept or reject this recently calculated number. It was decided that this lab just was not exciting enough; it did not have that interesting aspect that the group wanted so it was back to the drawing board.

The next experiment created was much more interesting. With some brainstorming and careful consideration the idea of a HotWheels™ lab dealing with toy cars and tracks could very well catch the students' attention. After a week of trial-and-error in the lab a rough draft of the experiment was created and numerous trial runs were conducted.

Though semi-successful it was evident that there were many changes that needed to be made. First of all, there was no concrete lab written out. The first run had the instructor telling the experimenter what needed to be done. This would not be what happened in the actual lab when students were conducting the experiment so a hard copy of the experiment needed to be created. The two individuals sat down and tried to collaborate on what needed to be said in the write-up and what information needed to be present in each section. Another week of experimenting went by with much success and by the third or fourth week it seemed as if everything had been put into place. The experiment was being run in the allotted time with no real concerns or problems arising. It was at this point that management turned its attention towards the applets. There was a good toolbox of applets constructed and it was time experimenting with them commenced.

The applets covered a variety of different aspects of physics from teaching and resource applets to everyday physics equation applets. Hundreds of lines of Java code were written per applet to try and enhance the material covered in the developing experiments. The applets themselves were creatively made and, with a little instruction, easy to use.

To start, the applets were simple, yet interesting. In physics you deal with Greek letters, as well as different powers of ten. Scientists use those symbols and powers to represent certain equations and aspects of the science. The team felt as if game-type “Memory” applets could be beneficial. These applets told the user the name of the Greek letter/power of ten, and then had them click on the matching Greek symbol/prefix/abbreviation that matched it. This would help the students familiarize the symbols with the names resulting in an association with the physics terminology. After the game was completed, the applets would show the total number of correct answers the student received and the time it took them to do so. These applets were very simple, but after being repeatedly played it became easy to associate the symbols with the words.

Next, the Statistical Analysis applet was approached. Yet again the applet was simple: the student enters a bunch of different data points, presses “analyze data” and the standard deviation, average, max/min and number of terms analyzed all show up. The applet was far from interesting. It was more of a helping tool than a re-enforcing applet. After the student calculated all the information by hand they could either confirm or reject if their numbers were correct by pressing a couple keys. It also plotted the data on a scattered plot and histogram showing exactly what the data analyzed.

The Acceleration applet showed a lot of potential. The possibilities were endless for this applet and, had an appropriate lab/experiment been designed to help it out, could have really enforced the material covered in the lab. In this applet the user types in a velocity and height. The parabola is then animated across the screen with a couple of other data points listed also. This was felt to have been a lost applet, given no real chance to thrive. An experiment could have been created where the user first sets the velocity of

an object at a certain point, drops/throws that object at a particular height, and then through careful calculation determines where the object would land. After numerous repetitions the user could place the data he calculated by hand into the applet and figure out if he was correct in his reasoning and calculations. Even then the applet itself could have somehow been programmed with a randomized basket or goal placed somewhere on the axis. The student would then, knowing the height of the basket/goal, calculate the velocity the ball would need to travel so it would land comfortably in its designated area. Those are just some thoughts for the future. The applet itself showed no real connection to any of the experiments that were created and therefore sat on the back burner of this toolbox.

The collision applet was quite interesting to the organizational team management department. The user types in two objects masses and their initial velocities and shows what would happen if said objects were to crash into one another. A lot of thought and rework was put into creating this applet. It started with a very rough draft and, after some reworking from the team members, kept being perfected until the end product was created. It was amazing to see how, after collision, what an object would do. Not truly understanding the laws of physics it became clear to the management department how much mass and velocity play into the grand scheme of things. These were just two different colored circles traveling towards one another; imagine much larger, real life objects.

Finally, the team looked at the Hooke's law applet. This tool told you what your spring constant or 'k' value was in respect to the calculation of the force and distance the spring stretched in meters. This applet looked useful when the spring experiment had first



started, but soon became evident it would not apply or help perfect our overall end product. This is where the “ongoing” part of this IQP really comes into consideration because future projects could pick up on the experiment and use this applet as reinforcement.

Playing with all these applets really did not seem like work at all. Writing the code was the hardest part. If a problem was noticed or if certain aspects of each applet could be enhanced it was the applet departments’ problem to figure out what line needed to go where; what back-slash or semi-colon needed tweaking. While these adjustments were being done, the Doppler Experiment was given hierarchy.

The Doppler Effect experiment had many ups and downs throughout the term. Whether it was ordering the parts, constructing the machine, or figuring out what kind of data needed to be collected for an efficient lab, this experiment proved to be a challenge. Once an experiment was established that demonstrated this effect, an informational session commenced to further the management departments’ knowledge of the Doppler Effect itself. The program Spectrum Lab shows how, when an object goes from close to far away from any given point, you hear the effects of the Doppler Effect, yet when you turn the entire device to a 90 degree angle there is no effect seen. This is because the sound source is not moving away from the given point; it is staying at a constant distance only rotating 360 degrees.

Working with this department proved to be beneficial. Had the two departments not sat down and discussed this experiment, they would have just seen a light-spectrum of colors moving in unconventional ways rather than the Doppler Effect in full effect. Unlike the energy lab, this part of the project is still in the experimental stage. It does

show extreme potential and all the right factors are in place, but an actual lab idea and write-up need to be created. This is a physics toolbox though, so raw materials needed to be present.

Finally, the organizational team management department had the opportunity to work with the video enhancing stimulation department. One full lab had been created in this IQP so the video really needed to hit home with that lab in mind. The video was created to grab the attentions of the students before they stepped foot in the lab. Ideas were bounced back and forth for what seemed like weeks before it was decided to take an exciting, hip-hop approach. In this day and age the majority of today's youth can somehow relate to the hip-hop era. If the video started off with a popular rap songs instrumental playing in the background the student would hopefully be hooked from the beginning. Then, after watching the video in its entirety, the student would understand the task at hand to be completed on the ensuing day. The two tools together (video and HotWheels™ experiment) would then get the students excited about physics. The group believed that relating to students in this way would really achieve that goal.

Working with the video enhancing stimulation department was a very knowledgeable and exciting time. One of the groups' main goals is to grab the attention of the younger students and this was the final cap to that. An experiment had been created that would accomplish that goal, but the video was a preview to it all. The students, not knowing what tomorrow's lab would bring, log onto the physics site where the video is uploaded and get a great feel for what is to come. It was the ultimate cap to getting the students enthused about physics.

With any project come problems, fortunately the management department expected these problems. The problems arrived in different aspects of the other departments. Technical difficulties prevented the video-enhancing department from recording as planned, unforeseen sensor issues and program malfunctions prevented the energy lab from being as accurate as expected, and the lack of Java knowledge had made it complicating and extremely aggravating whilst creating the applets. The team knew such problems would arise. There was nothing that could be done to prevent these problems; they just needed to be approached carefully and, through insightful thinking, fixed. If anything, the management department created a few of these problems to help the completion of the final product. If management did not understand the way something worked, or why the team needed a certain applet, the rest of the group would have to change their approach accordingly. Everything needed to make sense to even the least comprehensible of individuals.

After the problems were identified the solutions were then put into the works. Why is the video software not working? The transfer from a Mac to a PC was scrambling the code and causing the video not to play. With some online converting software the problem was fixed. Why is the Vernier software not working? What is it about the sensors that are causing them to calculate different results every trial? The new Vernier program was not as compatible with the older sensors. After some uninstalling and reinstalling the problem was gone and the numbers seemed to come out with as little as a 0.8 percent margin of error. Why were the equations in the applets coming out with obscure numbers and sequences? Through re-reading the different Java scripts and doing a little online investigation it became known that Java had a backwards way of design.

With a couple strokes of the keyboard things seemed to be back on track. Identifying these aforesaid problems took days, but fixing them (for the most part) took only hours.

A good professional management engineer does things on instinct first, then afterwards may sit down and reflect on what exactly they had done afterwards. The organizational team management department had sat down and dissected each aspect of the group: helping to rework, reword, sharpen and shine each individual department as well as the project as a whole. The department then started to think about what aspects of Management Engineering had been used to accomplish those tasks.

The majority of business related firms deal with a certain aspect of Management Engineering called Operations management. This term can be defined as creating, operating, and reworking a system so that a company's product/service is delivered in an appropriate manner. In this specific instance the firm was the Physics Production Corporation and the product or service was creating a physics toolbox that could be modified, enhanced, and used for years to come. The organizational team management department used the "creating, operating, and reworking" aspect of operations management to really pull everything together as a whole. Being the manager of this project if things turned south it was his head on the chopping block.

The department also realized that many aspects of the four competitive advantages were present in their current project. The four dimensions of competitive advantage are as follows: Cost/Price, Quality, Flexibility, and Reliability, all of which the department felt the group used to achieve their goals.

At the beginning of the term the team did not know exactly what was to be created in this project. They wanted to create something new and exciting that covered

the material, but usually that would mean dishing out a lot of money to do so. That being said money became a quick issue. The group needed to create a couple experiments that would not be too difficult or expensive to re-create in a lab environment. Basically, the team needed the experiments to be probable and affordable. If the physics department had to dish out hundreds of dollars per experiment, per student then the whole IQP would not be practical. That is where the idea for HotWheels™ cars came from. The group knew these cars and tracks could be bought in bulk for a cheap price and hoped the students would have fun using them.

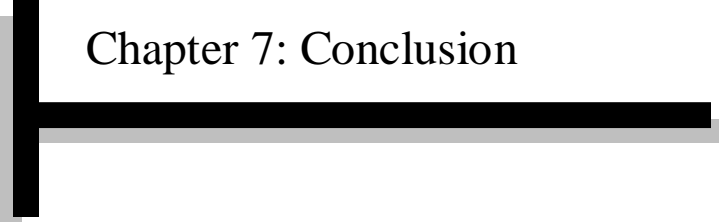
Next they tackled the quality aspect of competitive advantage. Quality is one of the most important parts of the competitive process. Just because your prices are low does not mean you are going to sell more of your product or service. Physics Production Corporation now knew their project would be cheap, but would it be educational as well? The group felt as if each tool in this toolbox had both the potential to be enticing as well as potential to grow. As far as they were concerned the quality aspect seemed concrete.

The third advantage was flexibility. Flexibility from an operational management stand point means that a firm offers a wide variety of products to their consumers. It has been mentioned numerous times that this IQP was in fact the inaugural process of creating a physics toolbox. By toolbox the group really means “wide variety” of physics applets, experiments, videos, etc. that will increase ones knowledge of the science. The organizational team management department saw the toolbox as highly flexible in this aspect.

This brings us to the final advantage: reliability. Reliability is a firm’s ability to offer a wide variety of products to its customers in a certain designated time either

established by the firm or by the consumer. In this instance the department felt as if the reliability could have been a little better. This is the first step of many in creating this physics toolbox but, some of the goals and expectations of the group were not met because of different circumstances. In a business like environment those departments would have been terminated or replaced by individuals that could have delivered on time and taken on that reliable role.

In conclusion the organizational team management departments' feelings towards physics have changed dramatically. At the onset of the term he thought physics was supposed to be the more exciting science of the big three (the other two being chemistry and biology), but after sitting down with the rest of the group and looking over past labs he quickly lost interest. If these older labs could not grab the attention of the management department then how would they grab the attention of incoming freshmen? The group had a big task they were presented with. Through many rough drafts, many ideas bounced off one another, and many late nights sitting in front of a computer screen wondering why this code did not work or why that program did not give them the data that was expected, the Physics Production Corporation finally arrived at a final product that they could be proud of and share with the Physics Department.



## Chapter 7: Conclusion

The inaugural Physics Production Company has successfully begun creating a toolbox of new material for the Physics department. The toolbox that has been developed has a solid foundation for expansion for the future. The first priority of the toolbox was to set in motion a change for laboratory experiments. This has been done first with the creation of a machine to produce the Doppler Effect. Having this machine is a terrific starting point to implement a laboratory experiment that has never been done before here at WPI. With the use of the Doppler machine and the knowledge that was gained by this inaugural Physics Production Company an educational laboratory experiment is within reach with only a little more effort. The next addition to the laboratory portion of the toolbox comes in the form of HotWheels™ cars and track. The laboratory experiment that has been developed is to emphasize energy in a more fun way than has been previously done by the Physics Department. The use of the cars and track rather than springs or other sorts of un-original physics laboratory equipment provides a unique and exciting addition to the toolbox. The versatility of HotWheels™ materials is basically endless and IQP groups in the future will be able to expand on this idea with ease. Another addition to the toolbox is the applet section. Two basic categories of applets were created, assisting applets, and simulation applets. The assisting applets section is a compilation of applets such as Greek letters and powers of ten whereas the simulation applets have collisions and projectile motion. These two broad categories support expansion in the future of the toolbox. Further web development to complement the creation of the applets will provide future groups with a way to continue to make their tools available to physics students through the web. The final tool that was introduced to the toolbox by the inaugural team of the Physics Production Company is video and still



frames for laboratory experiments. The videos and still frames are a great way to re-emphasize and support the laboratory experiments. The video that was made is specific for the HotWheels™ energy laboratory experiment, however, using this video as a template videos can be made for the rest of the laboratory experiments the department already has, as well as experiments the department will have once another Physics Production Company begins to work. All in all, the inaugural Physics Production Company has laid the groundwork to help expand the teaching capabilities of the Physics Department through this innovative toolbox.

## Bibliography

Absolute Java with Student Resource Disk (3rd Edition). Walter Savitch. Addison-Wesley Longman Publishing Co., Inc. 2007

Data Studio. Available through Pasco Co. at <http://www.pasco.com/datastudio/>

Eclipse: Java Development Tool. Available for download on the World Wide Web at <http://www.eclipse.org/downloads/>

Format Factory. Available for download on the World Wide Web at [http://www.download.com/FormatFactory/3000-2194\\_4-10819418.html?tag=mncol](http://www.download.com/FormatFactory/3000-2194_4-10819418.html?tag=mncol)

K-Lite Codec Pack. Available for download on the World Wide Web at [http://www.download.com/K-Lite-Mega-Codec-Pack/3000-13632\\_4-10794603.html?tag=mncol](http://www.download.com/K-Lite-Mega-Codec-Pack/3000-13632_4-10794603.html?tag=mncol)

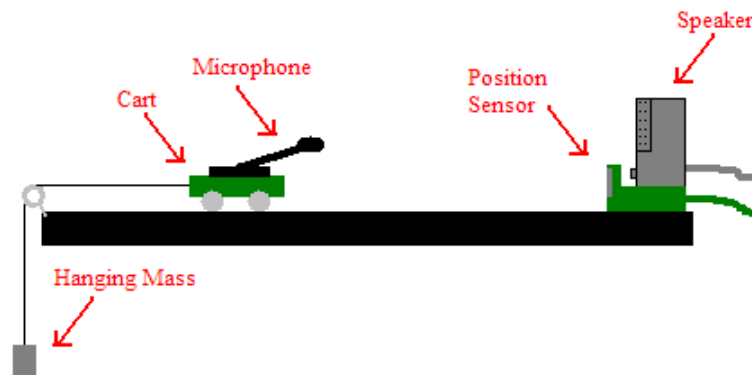
Sears and Zemansky's University Physics (12<sup>th</sup> Edition). Hugh D. Young and Roger A. Freedman. Pearson Education Inc. 2008

Spectrum Lab. Available for download on the World Wide Web at <http://freenet-homepage.de/dl4yhf/spectral1.html>

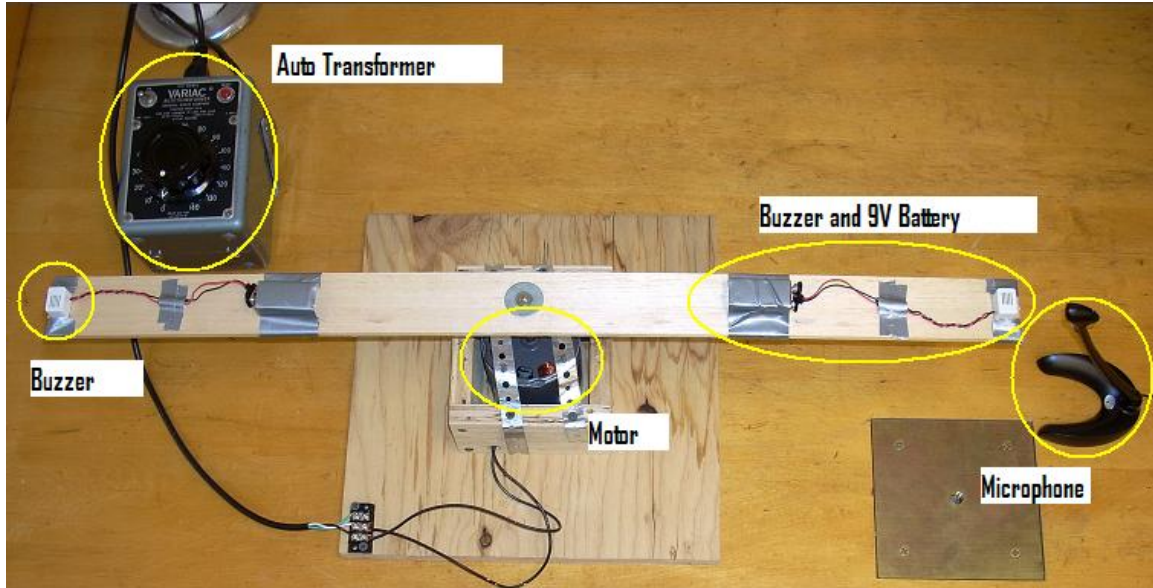
# Appendix

1. Doppler Laboratory
  - a. Linear Doppler Effect Schematic
  - b. Doppler Setup
  - c. Doppler Results
  - d. Doppler Results II
  
2. Energy Laboratory
  - a. Lab Write up
  - b. Data Sheet
  
3. PowerPoint Presentation
  
4. Applet Java Code
  - a. Acceleration
  - b. Collision
  - c. Greek Letters (Lower Case)
  - d. Greek Letters (Upper Case)
  - e. Hooke's Law
  - f. Powers of 10
  - g. Powers of 10 Abbreviations
  - h. Powers of 10 Prefixes
  - i. Statistical Analysis

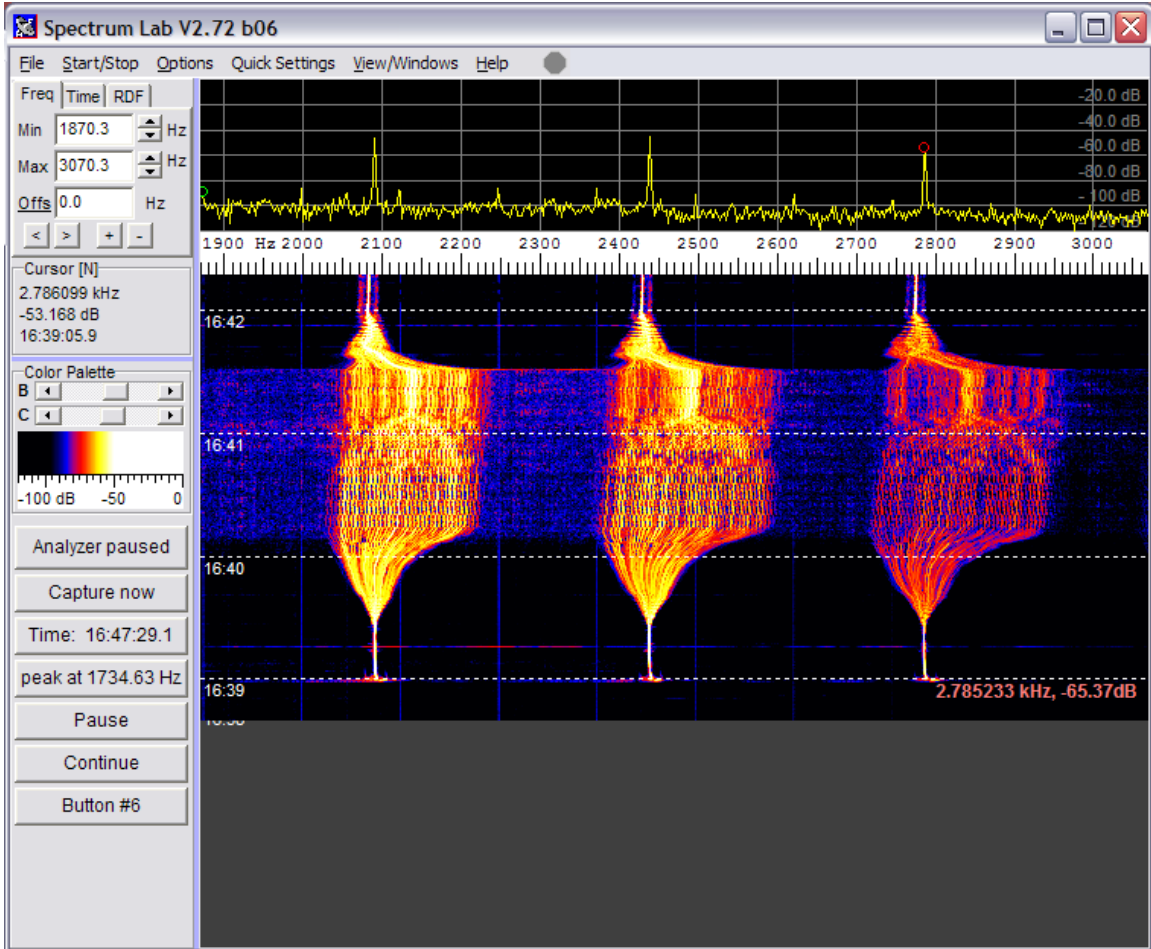
Appendix 1.a



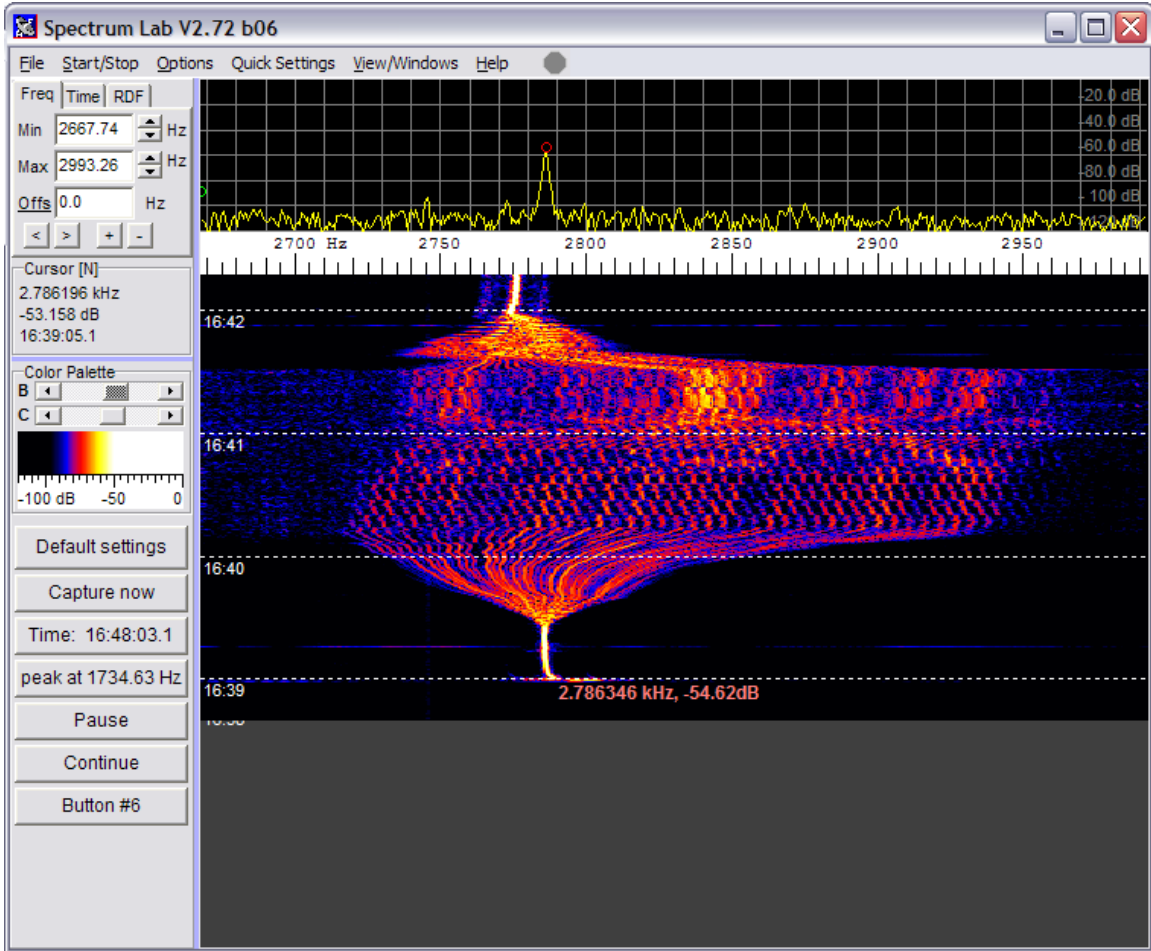
Appendix 1.b



Appendix 1.c



Appendix 1.d



## Energy Experiment



**Figure 1**



**Figure 2**

### Setup:

As shown in Figure 1, begin lab construction. When finished, the track should be comprised of four pieces. Once complete, adjust the clamps as necessary so that the top of the track reaches no more than 90cm. The track should have a gradual slope for more accurate results so it should flatten out at 120cm down the table from the pole it is clamped to. Secure the track to the table with tape in several places to make sure it is sturdy. Once the track is complete, set up both photogates using the gram masses as shown in Figure 2 (this should require a 50g and a 2g mass for each side). The first should be placed where the track becomes level, and the second should be placed approximately 40cm from the first. This should put it near the end of the track. Plug both photogates into the Logger Pro reader, and make sure that the 1<sup>st</sup> photogate that the car will pass through is plugged into Dig/Sonic 1, and verify that the beam will hover right over the sides of the track. If not, add or remove masses as necessary. Run the car through the photogates to verify that they are in a good position before you continue (the red light should turn on when the car is blocking the beam).

**IMPORTANT:** Verify that when the photogates are set up and plugged into the Logger Pro reader that the shutter on the inside of each photogate is in the “up” position. If it is down, the sensor will not read anything and you will not be able to take any data.

Measure the length of the car that you will be using. Open Energy Lab.cmb1 and click on Data and choose user parameters. Under Value for both Photogate Distance one and two, enter the length of your car. Once you have completed this you are ready to begin the experiment.



## **Procedure**

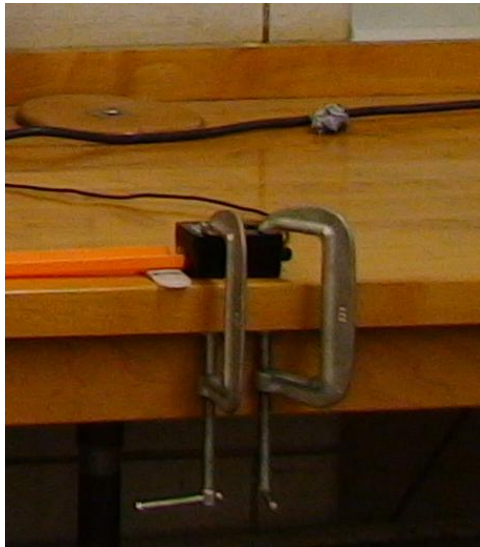
### I) Dual Photogates

Use the meter stick to choose a moderate dropping point for your first trial. (Somewhere between 30cm and 40cm should work well.)

Run the logger pro software, and release the car from your chosen height. Write down your results and repeat twice more for a total of three trials. Use the data that you record to calculate an average velocity through both the 1<sup>st</sup> and 2<sup>nd</sup> photogate.

Repeat for a height between 40cm and 50cm

Repeat for a height between 50 and 60



**Figure 3**

### II) Force Sensor

Remove the photogates, and then move the force sensor so that it is as close as you can get it to the track without actually touching it. Use the clamps to secure the sensor in place as shown in Figure 3 (make sure that both of the photogates are now unplugged). Open Energy Lab II.cmb1, choose the +/- 50 setting on the, and zero it (ctrl+0) for best results. Measure the mass of your car and record it on the data sheet. Release the car from the first height you chose in part 1 and observe the force reading. Repeat twice more and log your data in the data sheet.

Repeat again for the two other heights that you have data for.

## Worksheet

- 1) With the variables that you know after part I, are you able to say anything about the maximum potential energy, maximum kinetic energy, or the maximum velocity? (If no, why not? If yes, what is it?)
- 2) Using any data you found in part II solve for the remaining values from question 1.
- 3) How does the velocity you measured through the 1<sup>st</sup> photogate compare with the maximum velocity you calculated above? What are some possible sources that could account for any error that you ran into? (Hint: Compare the velocities through both photogates) How much work is being done on the car by gravity?
- 4) Using your velocity measurement through the 2<sup>nd</sup> photogate from Part 1, calculate the momentum of the car when it reaches the end of the track through the equation  $FT=MV$ . How well does this match the impulse that the force sensor recorded?
- 5) Calculate the percent error, and list possible reasons for error. Is your error value reasonable? Why or why not?

## DATA SHEET

1<sup>st</sup> Height:

Expected Velocity (m/s):

Photogate	Trial 1 Velocity (m/s)	Trial 2 Velocity (m/s)	Trial 3 Velocity (m/s)	Average Velocity (m/s)
1				
2				

2<sup>nd</sup> Height:

Expected Velocity (m/s):

Photogate	Trial 1 Velocity (m/s)	Trial 2 Velocity (m/s)	Trial 3 Velocity (m/s)	Average Velocity (m/s)
1				
2				

3<sup>rd</sup> Height:

Expected Velocity (m/s):

Photogate	Trial 1 Velocity (m/s)	Trial 2 Velocity (m/s)	Trial 3 Velocity (m/s)	Average Velocity (m/s)
1				
2				

Mass of Car:

Length of Car:

1<sup>st</sup> Height

	Trial 1	Trial 2	Trial 3	Average
Max Force (N)				
Impulse ( $N*s$ )				

Expected Impulse ( $N*s$ ):

2<sup>nd</sup> Height

	Trial 1	Trial 2	Trial 3	Average
Max Force (N)				
Impulse ( $N*s$ )				

Expected Impulse ( $N*s$ ):

3<sup>rd</sup> Height

	Trial 1	Trial 2	Trial 3	Average
Max Force (N)				
Impulse ( $N*s$ )				

Expected Impulse ( $N*s$ ):

Appendix 2.b

**DATA SHEET**

1<sup>st</sup> Height: 35 cm

Expected Velocity (m/s): 2.619

Photogate	Trial 1 Velocity (m/s)	Trial 2 Velocity (m/s)	Trial 3 Velocity (m/s)	Average Velocity (m/s)
1	2.544	2.532	2.524	2.533
2	2.500	2.469	2.483	2.485

2<sup>nd</sup> Height: 45 cm

Expected Velocity (m/s): 2.968

Photogate	Trial 1 Velocity (m/s)	Trial 2 Velocity (m/s)	Trial 3 Velocity (m/s)	Average Velocity (m/s)
1	2.788	2.845	2.816	2.816
2	2.751	2.796	2.686	2.744

3<sup>rd</sup> Height: 55 cm

Expected Velocity (m/s): 3.283

Photogate	Trial 1 Velocity (m/s)	Trial 2 Velocity (m/s)	Trial 3 Velocity (m/s)	Average Velocity (m/s)
1	3.172	3.168	3.162	3.167
2	3.106	3.100	3.095	3.100

Mass of Car (kg): .0343

Length of Car: .075m

1<sup>st</sup> Height

	Trial 1	Trial 2	Trial 3	Average
Max Force (N)	12.40	12.36	12.73	12.50
Impulse (N*s)	.0925	.0924	.0929	.0926

Expected Impulse (N\*s): .0898

2<sup>nd</sup> Height

	Trial 1	Trial 2	Trial 3	Average
Max Force (N)	15.43	16.16	15.67	15.75
Impulse (N*s)	.1043	.1074	.1053	.1056

Expected Impulse (N\*s): .1019

3<sup>rd</sup> Height

	Trial 1	Trial 2	Trial 3	Average
Max Force (N)	20.98	20.56	20.19	20.58
Impulse (N*s)	.1199	.1187	.1209	.1198

Expected Impulse (N\*s): .1126

Physics IQP

# Ongoing Advancement of the Physics Toolbox

Bennie Jones  
Bennett Lessard  
John McGinley  
Konrad Perry  
Kyle Pydynkowski

## Appendix 4.a Acceleration Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Label;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

@SuppressWarnings("serial")
public class Acceleration extends Applet{

    private Panel panel;
    private TextField tf,tf1,tf2;
    private Label label;
    private Button button;
    private int
ballx,ballxi,xpos,place=0,maxval,yvelocity,bally,ballyi,ypos,holder=0,t
imex;
    private double vox,xox,voy,ay,time;
    private boolean running = false, caughtException =
false,paused=false,hitend=false,restarting=false;
    @SuppressWarnings("unused")
    private Thread thread;
    private ArrayList<Integer>
myArrx,myArry,myArrx1,myArry1,myArrx2,myArry2,myArrx3,myArry3;

    @Override
    public void init(){
        //Initialize values used later in code
        thread = new Thread();
        time =0;
        ballx = 35; ballxi = 35;
        bally = 715; ballyi = 715;
        ay = 9.8;
        xpos = 0; ypos =0;
        //Setup the holders for the ball path depending on
place, which holds the current
//array number for the ball path
        switch(place){
            case 0:
                myArrx = new ArrayList<Integer>(1);
                myArry = new ArrayList<Integer>(1);
                break;
            case 1:
                myArrx1 = new ArrayList<Integer>(1);
                myArry1 = new ArrayList<Integer>(1);
                break;
            case 2:
                myArrx2 = new ArrayList<Integer>(1);
```



```

        myArry2 = new ArrayList<Integer>(1);
        break;
    case 3:
        myArrx3 = new ArrayList<Integer>(1);
        myArry3 = new ArrayList<Integer>(1);
        break;
    }

    //If the program is not restarting, then draw the
panel
    if(restarting==false){
        panel = new Panel();
        label = new Label("Enter the Initial x-
velocity");

        panel.add(label);
        tf = new TextField(5);
        panel.add(tf);
        label = new Label("Enter Initial y-position (0-
700)");

        panel.add(label);
        tf1 = new TextField(5);
        panel.add(tf1);
        label = new Label("Enter the Initial y-velocity
(-100-100)");

        panel.add(label);
        tf2 = new TextField(5);
        panel.add(tf2);
        button = new Button("  RUN  ");
        button.addActionListener(new buttonMenu(0));
        panel.add(button);
        button = new Button("RESTART");
        button.addActionListener(new buttonMenu(1));
        panel.add(button);
        button = new Button(" PAUSE ");
        button.addActionListener(new buttonMenu(2));
        panel.add(button);
        add(panel);}

        repaint();}

@Override
public void paint(Graphics g){
    //Setup for grid
    //Set color to black
    g.setColor(Color.BLACK);
    //draw grid lines
    g.drawLine(40, 40, 40, 740);
    g.drawLine(39, 40, 39, 740);
    g.drawLine(41, 40, 41, 740);
    g.setColor(Color.RED);
    g.drawLine(20, 719, 1040, 719);
    g.setColor(Color.BLACK);
    g.drawLine(20, 721, 1040, 721);
    g.drawLine(20, 720, 1040, 720);
    g.drawLine(20, 620, 1040, 620);
    g.drawLine(20, 520, 1040, 520);
    g.drawLine(20, 420, 1040, 420);
    g.drawLine(20, 320, 1040, 320);

```

```

g.drawLine(20, 220, 1040, 220);
g.drawLine(20, 120, 740, 120);
g.drawLine(140,20, 140,740);
g.drawLine(240,20, 240,740);
g.drawLine(340,20, 340,740);
g.drawLine(440,20, 440,740);
g.drawLine(540,20, 540,740);
g.drawLine(640,20, 640,740);
g.drawLine(740,20, 740,740);
g.drawLine(840,220, 840,740);
g.drawLine(940,220, 940,740);
//Setup for Y velocity Graph
g.drawLine(810,60,810,180);
g.drawLine(810, 120, 840, 120);
//Set color to Magenta
g.setColor(Color.BLUE);
//Write label
g.drawString("Y Velocity", 750, 125);
//Logic code to draw the velocity bar graph depending
on the sign of the velocity
if(yvelocity>=0){
    g.fillRect(815, 120-yvelocity/2, 20,
abs(yvelocity)/2);
    g.drawString(""+abs(yvelocity)/2, 845, 125);}
if(yvelocity<0){
    g.fillRect(815, 120, 20, abs(yvelocity));
    g.drawString("-"+abs(yvelocity), 845, 125);
}
//Setup for Statistics Corner
g.setColor(Color.GREEN);
g.drawString("Max Height Reached: ", 855, 60);
g.drawString("Time in Air: "+timex, 855, 80);
//Setup for Ball Path
g.setColor(Color.RED);
g.fillOval(ballx, bally, 10, 10);
//Logic to determine which ball path to draw

if((place==0&&time!=0)||holder>0){
    for(int a=0;a<myArrx.size();a++){
        g.setColor(Color.RED);
        g.fillOval(myArrx.get(a),
myArray.get(a), 5, 5);}}
if(holder>1||(place==1&&time!=0)){
    for(int a=0;a<myArrx1.size();a++){
        g.setColor(Color.BLUE);
        g.fillOval(myArrx1.get(a),
myArray1.get(a), 5, 5);}}
if(holder>2||(place==2&&time!=0)){
    for(int a=0;a<myArrx2.size();a++){
        g.setColor(Color.GREEN);
        g.fillOval(myArrx2.get(a),
myArray2.get(a), 5, 5);}}
if(holder>3||(place==3&&time!=0)){
    for(int a=0;a<myArrx3.size();a++){
        g.setColor(Color.CYAN);
        g.fillOval(myArrx3.get(a),
myArray3.get(a), 5, 5);}}

```

```

        g.setColor(Color.RED);
        g.fillRect(ballx+2, 720, 5, 10);
        g.fillRect(30, bally+2, 10, 5);
        //Writes the coordinates of the ball at each
instant in time
        g.setColor(Color.BLUE);
        g.drawString("(" +xpos+" , "+ypos+"",ballx-
5,bally-5);

        g.drawString("(" +xpos+"", ballx+15, 735);
        g.drawString("(" +ypos+"", 20, bally+15);
        //Write the highest point only when the ball
has reached the end
        g.setColor(Color.GREEN);
        if(hitend==true){
            g.drawString("Max Height Reached:
"+maxval, 855, 60);}

        //Run the ball in flight code only if the run button has
been clicked,
        //the pause function is not enabled, and the ball has not
yet hit the end
        if(running==true&&paused==false&&hitend==false){
            runit();
        }

        @SuppressWarnings("static-access")
        public void runit(){
            //try catch block used to slow code run
            try {
                Thread.sleep(125);
            } catch (InterruptedException e) {
                e.printStackTrace();}
            //logic to determine if the ball has reached the end
of flight
            if(time!=0&&ypos<=0){
                hitend=true;}
            //While the ball is still in the air, perform the
equations to determine movement
            if(hitend==false){
                time = time+(.15);
                timex = (int) time;
                ballx = (int) (ballxi+(vox*time));
                xpos = (int) ((vox*time));
                ypos = (int) ((xox+voy*time-((.5)*ay*time*time));
                bally = (int) (ballyi+(-xox-
voy*time+((.5)*ay*time*time));
                yvelocity = (int) ((voy-ay*time));
                //stores the path of the ball into the correct
arrayList
            if(place==0){
                myArrx.add(ballx);
                myArray.add(bally);}
            if(place==1){
                myArrx1.add(ballx);
                myArray1.add(bally);}
            if(place==2){
                myArrx2.add(ballx);

```

```

        myArry2.add(bally);}
    if(place==3){
        myArrx3.add(ballx);
        myArry3.add(bally);}
    //Run paint
    repaint();
}

//calculates the maximum value based on the
input conditions
    if(hitend==true){
        maximum();}
}
public void maximum(){
    int maxy = (int) ((-voy)/(-ay));
    if(maxy<0){
        maxy=0;
    }
    maxval = (int) (xox+voy*maxy-(.5)*ay*maxy*maxy);
    repaint();
}
public int abs(int a){
    //A function used to find the absolute value of an integer.
Used in the Y-Velocity graph
    if(a<0){
        a=a*-1;
    }
    return a;
}

public class buttonMenu implements ActionListener{
    private int what;
    //Constructor for the ButtonMenu. The function of each
button depends on its integer value
    public buttonMenu(int a){
        what = a;
    }
    public void actionPerformed(ActionEvent arg0) {
        //If the run button is pressed, and the ball isn't
already in run mode:
        if(what==0&&running==false){
            //Try to turn the values in each of the text
fields into an integer.
            try{
                vox = Double.parseDouble(tf.getText());
                xox = Double.parseDouble(tf1.getText());
                voy = Double.parseDouble(tf2.getText());
                //Check that values are within bounds
                //Initial x velocity must be greater than
zero
                if(xox<0){
                    xox=0;
                    tf1.setText(""+0);
                }

                if(xox>700){
                    xox=700;
                    tf1.setText(""+700);

```

```

    }
    if(vox<0){
        vox=0;
        tf.setText(""+0);
    }
    if(voy>100){
        voy=100;
        tf2.setText("100");
    }
    if(voy<-100){
        voy=-100;
        tf2.setText("-100");
    }
}
//if there was an exception trying to parse the
values to ints, then make the boolean value
//caughtException true
catch(Exception e){
    caughtException = true;
}
//As long as there wasn't an exception, the
values were entered properly, and the ball can be put into run mode
if(caughtException==false){
    running=true;
    paused=false;
}
caughtException=false;
repaint();
}
if(what==0&&running==true){
    restarting=false;
}
//Restart button was pressed, change all the values
to their initial states, then run the initializer
//with the boolean value of restarting true
if(what==1&&restarting!=true){
    if(running=true){
        if(place==3){
            place=-1;
        }
        place++;
        holder++;
        restarting=true;
        running =false;
        paused =false;
        hitend=false;
        init();
    }
    repaint();
}
//Pause button was pressed, change the value of the
boolean paused to the opposite of what it was
if(what==2){
    paused = !paused;
    repaint();
}}}}

```

## Appendix 4.b Collision Java Code

```
import java.applet.Applet;
import java.awt.BasicStroke;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Label;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/*
 * Collision2 is an applet that simulates an elastic collision of two
objects
 * These objects are allowed to have masses between 500 and 2000 units,
and
 * velocities between 0 and 25 units.
 */
@SuppressWarnings("serial")
public class Collision2 extends Applet
{
    private Panel panel;
    private TextField mass1, mass2, veloc1, veloc2;
    private Label label;
    private Button button;
    private Thread thread;
    private double massA, massB, velocA, velocB, time, time2,
collision, vfa, vfb,vfat,vfvt;
    private boolean running, restart = false, collide;

    public Collision2()
    {
        initialize();}
    public void initialize()
    {
        //Initializes values used later in the code, as well as
creates the objects that
        //the panel is made of, including the labels, text fields
and buttons.
        time2=0;
        thread = new Thread();
        time = 0;
        collide = false;
        if (restart == false)
        {
            panel = new Panel();
            label = new Label("Enter Mass of Object A:");
            panel.add(label);
            mass1 = new TextField(5);
            panel.add(mass1);
            label = new Label("Enter Velocity of Object A:");
```

```

        panel.add(label);
        veloc1 = new TextField(5);
        panel.add(veloc1);
        label = new Label("Enter the Mass of Object B:");
        panel.add(label);
        mass2 = new TextField(5);
        panel.add(mass2);
        label = new Label("Enter Velocity of Object B:");
        panel.add(label);
        veloc2 = new TextField(5);
        panel.add(veloc2);
        button = new Button(" RUN ");
        button.addActionListener(new buttonMenu(0));
        panel.add(button);
        button = new Button("RESET");
        button.addActionListener(new buttonMenu(1));
        panel.add(button);
        add(panel);
    }
    repaint();
}
/*
 * paint contains all the code which draws the objects as well as
the graph at the bottom
 * of the applet. There are four sections of the paint() funtion.
The first sets up the
 * initial velocity displays, as well as the velocity-time graph
setup. The second contains
 * the code that runs while the objects are moving but have not
yet collided. The third is the
 * section of the code that contains the action that occurs when
the collision has occurred
 * and the objects are still in motion. The last part of the
paint() funtion operates once
 * the objects have stopped moving.
 */
public void paint(Graphics g)
{
    //Displays the initial velocities entered by the user.
    g.drawString("Initial Velocity of Object A: "+velocA, 20,
225);
    g.drawString("Initial Velocity of Object B: "+(-velocB), 20,
250);

    //Sets up the velocity time graph
    g.drawLine(20, 350, 1000, 350);
    g.drawLine(20, 300, 20, 400);
    g.drawString("V", 10, 352);
    //For loop draws the dash marks on the graph
    for (int a = 1; a < 21; a++){
        g.drawLine(20 + a * 49, 340, 20 + a * 49, 360);}
    Graphics2D g2 =(Graphics2D)g;
    g2.setStroke(new BasicStroke(2));
    //Section of the code that runs when the objects are
running but have not yet collided.
    //Notice the boolean values !collide and running.
    if (!collide&&running)
    {

```

```

        g.setColor(Color.BLUE);
        //Draws the graph of the velocity
        g.drawLine(20, (int)(350 - velocA), (int)(20 + time
*490/collision), (int)(350 - velocA));
        //Draws object A
        g.fillOval((int)(300 + (time * velocA)), (int)(150 -
(massA/40)), (int)(massA/20), (int)(massA/20));
        g.setColor(Color.RED);
        //Draw the graph of the velocity of B
        g.drawLine(20, (int)(350 + velocB), (int)(20 + time *
490/collision), (int)(350 +velocB));
        //Draws object B
        g.fillOval((int)(700 - (time * velocB)), (int)(150 -
(massB/40)), (int)(massB/20), (int)(massB/20));
        //Changes the font to Bold
        Font font = new Font("Arial",Font.BOLD,12);
        g.setFont(font);
        //Draws the labels in both object A and B
        g.drawString("A", (int)(300+massA/40 + (time *
velocA)), (150));
        g.setColor(Color.BLUE);
        g.drawString("B", (int)(700+massB/40- (time *
velocB)),150);}
        //Section of the code that runs when the objects are
running and have collided
        if (collide&&running)
        {
            g.setColor(Color.BLUE);
            //Draws the graph of the velocity of object A. First
it draws the initial velocity
            //Then a connecting line between the initial and the
final velocities, and then the
            //final velocity
            g.drawLine((int)(20+time2*490/collision), (int)(350-
velocA*5/4), (int)(20+time2*490/collision), (int)(350 - vfa*5/4));
            g.drawLine(20, (int)(350-
velocA*5/4), (int)(20+time2*490/collision), (int)(350-velocA*5/4));
            g.drawLine((int)(20+time2*490/collision), (int)(350
- vfa*5/4), (int)(20 + (time2 + time) *490/collision), (int)(350 -
vfa*5/4));

            //Draws object A
            g.drawString("Final Velocity of Object A: "+vfat, 250,
225);

            g.fillOval((int)((300 + collision * velocA) + (time *
vfa)), (int)(150 - (massA/40)), (int)(massA/20), (int)(massA/20));
            g.setColor(Color.RED);
            //Draws the graph of the velocity of object B. First
it draws the initial velocity
            //Then a connecting line between the initial and the
final velocities, and then the
            //final velocity
            g.drawLine((int)(20+time2*490/collision),
(int)(350+velocB*5/4), (int)(20+time2*490/collision), (int)(350+
vfb*5/4));

            g.drawLine(20, (int)
(350+velocB*5/4), (int)(20+time2*490/collision), (int)(350+velocB*5/4));

```



```

        g.drawLine((int) (20+time2*490/collision), (int) (350
+ vfb*5/4), (int) (20 + (time2 + time) *490/collision), (int) (350 +
vfb*5/4));
        //Draws object B
        g.fillOval((int)((700 - collision * velocB) - (time *
vfb)), (int) (150 - (massB/40)), (int) (massB/20), (int) (massB/20));
        g.drawString("Final Velocity of Object B: "+(-vfbt),
250, 250);
        //Changes the font to Bold
        Font font = new Font("Arial",Font.BOLD,12);
        g.setFont(font);
        g.drawString("A", (int) (300+massA/40 + (collision *
velocA) + (time * vfa)), (150));
        g.setColor(Color.BLUE);
        g.drawString("B", (int) ((700+massB/40- collision *
velocB) - (time * vfb)),150);
    }
    if (collide&&!running)
    {
        g.setColor(Color.BLUE);
        g2.drawLine((int) (20+time2*490/collision), (int) (350-
velocA*5/4), (int) (20+time2*490/collision), (int) (350- vfa*5/4));
        g2.drawLine(20, (int) (350-
velocA*5/4), (int) (20+time2*490/collision), (int) (350-velocA*5/4));
        g2.drawLine((int) (20+time2*490/collision), (int) (350
- vfa*5/4), (int) (20 + (time2 + time) *490/collision), (int) (350-
vfa*5/4));
        g.setColor(Color.RED);
        g2.drawLine((int) (20+time2*490/collision),
(int) (350+velocB*5/4), (int) (20+time2*490/collision), (int) (350+
vfb*5/4));
        g2.drawLine(20, (int)
(350+velocB*5/4), (int) (20+time2*490/collision), (int) (350+velocB*5/4));
        g2.drawLine((int) (20+time2*490/collision),
(int) (350+ vfb*5/4), (int) (20 + (time2 + time)*490/collision),
(int) (350+ vfb*5/4));
        g.setColor(Color.BLUE);
        g.fillOval((int)((300 + collision * velocA) + (time *
vfa)), (int) (150 - (massA/40)), (int) (massA/20), (int) (massA/20));
        g.drawString("Final Velocity of Object A: "+vfat, 250,
225);
        g.setColor(Color.RED);
        g.drawString("Final Velocity of Object B: "+(-vfbt),
250, 250);
        g.fillOval((int)((700 - collision * velocB) - (time *
vfb)), (int) (150 - (massB/40)), (int) (massB/20), (int) (massB/20));
        Font font = new Font("Arial",Font.BOLD,12);
        g.setFont(font);
        g.drawString("A", (int) (300+massA/40 + (collision *
velocA) + (time * vfa)), (150));
        g.setColor(Color.BLUE);
        g.drawString("B", (int) ((700+massB/40- collision *
velocB) - (time * vfb)),150);
    }
    if(running == true)

```

```

        {running();}
    }

    @SuppressWarnings("static-access")
    public void running()
    {
        time = time + .1;
        if (time >= collision && collide == false)
        {
            collide = true;
            time2 = time;
            time = 0;
        }
        if(collide==true&&time+time2>collision*2){//This line
determines when the running stops (65)
            running=false;}
        try
        {
            thread = new Thread();
            thread.sleep(12);
        }
        catch (Exception e){}
        repaint();
    }

    public class buttonMenu implements ActionListener
    {
        private int value;

        public buttonMenu(int a)
        {
            value = a;
        }

        public void actionPerformed(ActionEvent arg0)
        {
            if (value == 0)
            {
                running = true;
                try
                {
                    massA =
Double.parseDouble(mass1.getText());
                    massB =
Double.parseDouble(mass2.getText());
                    velocA =
Double.parseDouble(veloc1.getText());
                    velocB =
Double.parseDouble(veloc2.getText());
                    if (massA < 500){
                        mass1.setText("500");
                        massA = 500;}
                    if (massA > 2000){
                        massA = 2000;
                        mass1.setText("2000");}
                    if (massB < 500){

```

```

        massB = 500;
        mass2.setText("500");}
if (massB > 2000){
    mass2.setText("2000");
    massB = 2000;}
if (velocA < 0){
    velocA = 0;
    veloc1.setText("0");}
if (velocA > 25){
    velocA = 25;
    veloc1.setText("25");}
if (velocB < 0){
    velocB = 0;
    veloc2.setText("0");}
if (velocB > 25){
    velocB = 25;
    veloc2.setText("25");}
    }
catch (Exception e)
{
    running = false;
}
collision = ((400 - (massA/20)) / ((velocA +
velocB)));
vfa = ((massA - massB) / (massA + massB)) *
velocA + ((2 * massB) / (massA + massB)) * (-velocB));
vfb = -((massB - massA) / (massA + massB)) *
(-velocB) + ((2 * massA) / (massA + massB)) * velocA);
long x=(long) (vfa*100);
vfat=(double)x/100;
long y=(long) (vfb*100);
vfbt=(double)y/100;}
if (value == 1)
{
    running = false;
    restart = true;
    time = 0;
    time2=0;
    collide = false;}
repaint();
}}}
```

## Appendix 4.c

### Greek Lower Case Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Panel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

@SuppressWarnings("serial")
public class GreekLower extends Applet implements MouseListener{

    private ArrayList<Integer> list;
    private int r,totaladd,x,y,greek,greekM,times,missedmatch;
    private boolean correct,first,run,win;
    private Button button;
    private Random generator;
    private Image Img;
    private MediaTracker tr;
    private Panel panel;
    private Timer time;
    private Font font, font2;

    public GreekLower(){
        initialize();
    }

    public void initialize(){
        font = new Font("Arial",Font.BOLD,18);
        font2 = new Font("Arial",Font.BOLD,30);
        time= new Timer();
        time.scheduleAtFixedRate(new ToDoTask(), 0, 1000);
        generator = new Random();
        correct=true;
        totaladd=-1;
        list = new ArrayList<Integer>(24);
        Generate();
        first=true;
        panel = new Panel();
        button = new Button("BEGIN");
        button.addActionListener(new ButtonPress(0));
        panel.add(button);
        button = new Button("RESET");
        button.addActionListener(new ButtonPress(1));
        panel.add(button);
    }
}
```

```

        add(panel);
        addMouseListener(this);
        repaint();
    }

    public void paint(Graphics g){
        tr = new MediaTracker(this);
        Img = getImage(getDocumentBase(),"LowerGreek.jpg");
        tr.addImage(Img, 0);
        g.drawImage(Img, 10, 20, null);
        g.setFont(font);
        g.drawString("Click on the Greek Symbol that Matches", 60,
240);
        g.drawString("The name of the Greek Letter Below:", 70,
260);
        g.drawString("Total Correct: "+totaladd,160,320);
        g.drawString("Time: "+times, 190, 340);
        g.setColor(Color.BLUE);

        g.setFont(font2);
        if(!win&&run){
            switch(r){
                case 0:
                    g.drawString("Alpha", 190, 290);break;
                case 1:
                    g.drawString("Beta", 190, 290);break;
                case 2:
                    g.drawString("Gamma", 190, 290);break;
                case 3:
                    g.drawString("Epsilon", 190, 290);break;
                case 4:
                    g.drawString("Zeta", 190, 290);break;
                case 5:
                    g.drawString("Eta", 190, 290);break;
                case 6:
                    g.drawString("Theta", 190, 290);break;
                case 7:
                    g.drawString("Iota", 190, 290);break;
                case 8:
                    g.drawString("Kappa", 190, 290);break;
                case 9:
                    g.drawString("Lambda", 190, 290);break;
                case 10:
                    g.drawString("Mu", 190, 290);break;
                case 11:
                    g.drawString("Nu", 190, 290);break;
                case 12:
                    g.drawString("Xi", 190, 290);break;
                case 13:
                    g.drawString("Omicron", 190, 290);break;
                case 14:
                    g.drawString("Pi", 190, 290);break;
                case 15:
                    g.drawString("Rho", 190, 290);break;
                case 16:
                    g.drawString("Sigma", 190, 290);break;
                case 17:

```

```

        g.drawString("Tau", 190, 290);break;
    case 18:
        g.drawString("Upsilon", 190, 290);break;
    case 19:
        g.drawString("Phi", 190, 290);break;
    case 20:
        g.drawString("Chi", 190, 290);break;
    case 21:
        g.drawString("Psi", 190, 290);break;
    case 22:
        g.drawString("Omega", 190, 290);break;
    case 23:
        g.drawString("Delta", 190, 290);break;}}

    if(first){
        first=false;
        repaint();
    }
    if(win){
        g.setFont(font);
        g.setColor(Color.RED);
        g.drawString("You matched 24 Greek letters with
their",50,360);
        g.drawString("names in a total of "+times+"
seconds",85,380);
    }
    Generate();

}

public void Generate(){
    if(correct){
        r = generator.nextInt(24);
        greek=r;
        Object o = new Object();
        o = (Object) r;
        if(list.contains(o)&&totaladd<23){
            Generate();
        }
        else{
            totaladd++;
            if(totaladd<25){
                list.add(r);}
            repaint();
        }
    }
    if(totaladd==24){
        time.cancel();
        win=true;
        list.clear();
        generator = new Random();
    }
    correct = false;
}

public void Reset(){
    run=false;
    win=false;
    totaladd=-1;
}

```

```

correct=true;
list.clear();
time.cancel();
times=0;
time = new Timer();
time.scheduleAtFixedRate(new ToDoTask(), 0, 1000);
repaint();
}

```

```

@Override
public void mouseClicked(MouseEvent e) {
    x=e.getX();
    y=e.getY();
    if (x>23&&x<66&&y>39&&y<85) {
        greekM=0;}
    if (x>86&&x<116&&y>39&&y<85) {
        greekM=1;}
    if (x>139&&x<170&&y>39&&y<85) {
        greekM=2;}
    if (x>191&&x<217&&y>39&&y<85) {
        greekM=23;}
    if (x>236&&x<267&&y>39&&y<85) {
        greekM=3;}
    if (x>287&&x<324&&y>39&&y<85) {
        greekM=4;}
    if (x>343&&x<373&&y>39&&y<85) {
        greekM=5;}
    if (x>402&&x<429&&y>39&&y<85) {
        greekM=6;}
    if (x>21&&x<46&&y>115&&y<154) {
        greekM=7;}
    if (x>66&&x<103&&y>115&&y<154) {
        greekM=8;}
    if (x>123&&x<154&&y>115&&y<154) {
        greekM=9;}
    if (x>177&&x<212&&y>115&&y<154) {
        greekM=10;}
    if (x>236&&x<262&&y>115&&y<154) {
        greekM=11;}
    if (x>292&&x<318&&y>115&&y<154) {
        greekM=12;}
    if (x>345&&x<371&&y>115&&y<154) {
        greekM=13;}
    if (x>395&&x<429&&y>115&&y<154) {
        greekM=14;}
    if (x>16&&x<46&&y>178&&y<221) {
        greekM=15;}
    if (x>74&&x<107&&y>178&&y<221) {
        greekM=16;}
    if (x>128&&x<156&&y>178&&y<221) {
        greekM=17;}
    if (x>179&&x<211&&y>178&&y<221) {
        greekM=18;}
    if (x>236&&x<265&&y>178&&y<221) {
        greekM=19;}
    if (x>288&&x<322&&y>178&&y<221) {

```

```

        greekM=20;}
    if (x>343&& x<382&& y>178&& y<221) {
        greekM=21;}
    if (x>406&& x<448&& y>178&& y<221) {
        greekM=22;}
    if (greek!=greekM) {
        missedmatch++;
        if (missedmatch==5) {
            missedmatch=0;
            list.remove(list.size()-1);
            correct=true;
            totaladd--;
            Generate();
        }
    }
    if (greek==greekM) {
        correct=true;
    }
    repaint();
}

@Override
public void mouseEntered(MouseEvent e) {}

@Override
public void mouseExited(MouseEvent e) {}

@Override
public void mousePressed(MouseEvent e) {}

@Override
public void mouseReleased(MouseEvent e) {}

public class ButtonPress implements ActionListener{
    private int value;
    public ButtonPress(int a){
        this.value=a;
    }
    public void actionPerformed(ActionEvent arg0) {
        if (this.value==0) {
            run=true;
            repaint();
        }
        if (this.value==1) {
            Reset();
        }
    }
}

class ToDoTask extends TimerTask {
    public void run ( ) {
        if (run&&!correct) {
            times++;
            repaint();
        }
    }
}
}
}
}

```



## Appendix 4.d Greek Upper Case Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Panel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

@SuppressWarnings("serial")
public class GreekUpper extends Applet implements MouseListener{

    private ArrayList<Integer> list;
    private int r,totaladd,x,y,greek,greekM,times,missedmatch;
    private boolean correct,first,run,win;
    private Button button;
    private Random generator;
    private Image Img;
    private MediaTracker tr;
    private Panel panel;
    private Timer time;
    private Font font, font2;

    public GreekUpper(){
        initialize();
    }

    public void initialize(){
        font = new Font("Arial",Font.BOLD,18);
        font2 = new Font("Arial",Font.BOLD,30);
        time= new Timer();
        time.scheduleAtFixedRate(new ToDoTask(), 0, 1000);
        generator = new Random();
        correct=true;
        totaladd=-1;
        list = new ArrayList<Integer>(24);
        Generate();
        first=true;
        panel = new Panel();
        button = new Button("BEGIN");
        button.addActionListener(new ButtonPress(0));
        panel.add(button);
        button = new Button("RESET");
        button.addActionListener(new ButtonPress(1));
```

```

        panel.add(button);
        add(panel);
        addMouseListener(this);
        repaint();
    }

    public void paint(Graphics g){
        tr = new MediaTracker(this);
        Img = getImage(getDocumentBase(),"UpperGreek.jpg");
        tr.addImage(Img, 0);
        g.drawImage(Img, 10, 30, null);
        g.setFont(font);
        g.drawString("Click on the Greek Symbol that Matches", 60,
240);
        g.drawString("The name of the Greek Letter Below:", 70,
260);
        g.drawString("Total Correct: "+totaladd,160,320);
        g.drawString("Time: "+times, 190, 340);
        g.setColor(Color.BLUE);
        g.setFont(font2);
        if(!win&&run){
            switch(r){
                case 0:
                    g.drawString("Alpha", 190, 290);break;
                case 1:
                    g.drawString("Beta", 190, 290);break;
                case 2:
                    g.drawString("Gamma", 190, 290);break;
                case 3:
                    g.drawString("Epsilon", 190, 290);break;
                case 4:
                    g.drawString("Zeta", 190, 290);break;
                case 5:
                    g.drawString("Eta", 190, 290);break;
                case 6:
                    g.drawString("Theta", 190, 290);break;
                case 7:
                    g.drawString("Iota", 190, 290);break;
                case 8:
                    g.drawString("Kappa", 190, 290);break;
                case 9:
                    g.drawString("Lambda", 190, 290);break;
                case 10:
                    g.drawString("Mu", 190, 290);break;
                case 11:
                    g.drawString("Nu", 190, 290);break;
                case 12:
                    g.drawString("Xi", 190, 290);break;
                case 13:
                    g.drawString("Omicron", 190, 290);break;
                case 14:
                    g.drawString("Pi", 190, 290);break;
                case 15:
                    g.drawString("Rho", 190, 290);break;
                case 16:
                    g.drawString("Sigma", 190, 290);break;
                case 17:

```

```

        g.drawString("Tau", 190, 290);break;
    case 18:
        g.drawString("Upsilon", 190, 290);break;
    case 19:
        g.drawString("Phi", 190, 290);break;
    case 20:
        g.drawString("Chi", 190, 290);break;
    case 21:
        g.drawString("Psi", 190, 290);break;
    case 22:
        g.drawString("Omega", 190, 290);break;
    case 23:
        g.drawString("Delta", 190, 290);break;}}
    if(first){
        first=false;
        repaint();
    }
    if(win){
        g.setFont(font);
        g.setColor(Color.RED);
        g.drawString("You matched 24 Greek letters with
their",50,360);
        g.drawString("names in a total of "+times+"
seconds",85,380);
    }
    Generate();
}

public void Generate(){
    if(correct){
        r = generator.nextInt(24);
        greek=r;
        Object o = new Object();
        o = (Object) r;
        if(list.contains(o)&&totaladd<23){
            Generate();
        }
        else{
            totaladd++;
            if(totaladd<25){
                list.add(r);}
            repaint();
        }
    }
    if(totaladd==24){
        time.cancel();
        win=true;
        list.clear();
        generator = new Random();
    }
    correct = false;
}

public void Reset(){
    win=false;
    run=false;
    totaladd=-1;
}

```

```

correct=true;
list.clear();
time.cancel();
times=0;
time = new Timer();
time.scheduleAtFixedRate(new ToDoTask(), 0, 1000);
repaint();
}

```

```

@Override
public void mouseClicked(MouseEvent e) {
    x=e.getX();
    y=e.getY();
    if (x>43&&x<77&&y>39&&y<75) {
        greekM=0;}
    if (x>91&&x<122&&y>39&&y<75) {
        greekM=1;}
    if (x>138&&x<169&&y>39&&y<75) {
        greekM=2;}
    if (x>184&&x<215&&y>39&&y<75) {
        greekM=23;}
    if (x>229&&x<258&&y>39&&y<75) {
        greekM=3;}
    if (x>275&&x<304&&y>39&&y<75) {
        greekM=4;}
    if (x>316&&x<355&&y>39&&y<75) {
        greekM=5;}
    if (x>368&&x<404&&y>39&&y<75) {
        greekM=6;}
    if (x>38&&x<55&&y>98&&y<134) {
        greekM=7;}
    if (x>68&&x<105&&y>98&&y<134) {
        greekM=8;}
    if (x>117&&x<153&&y>98&&y<134) {
        greekM=9;}
    if (x>167&&x<212&&y>98&&y<134) {
        greekM=10;}
    if (x>228&&x<259&&y>98&&y<134) {
        greekM=11;}
    if (x>276&&x<307&&y>98&&y<134) {
        greekM=12;}
    if (x>321&&x<355&&y>98&&y<134) {
        greekM=13;}
    if (x>372&&x<410&&y>98&&y<134) {
        greekM=14;}
    if (x>38&&x<63&&y>161&&y<196) {
        greekM=15;}
    if (x>80&&x<107&&y>161&&y<196) {
        greekM=16;}
    if (x>122&&x<155&&y>161&&y<196) {
        greekM=17;}
    if (x>164&&x<202&&y>161&&y<196) {
        greekM=18;}
    if (x>216&&x<253&&y>161&&y<196) {
        greekM=19;}
    if (x>268&&x<303&&y>161&&y<196) {

```



## Appendix 4.e Hooke's Law Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Label;
import java.awt.MediaTracker;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

@SuppressWarnings("serial")
public class HookeLaw extends Applet{

    private Panel panel;
    private TextField f1, f2, f3, f4, f5, x1, x2, x3, x4, x5;
    private Label label;
    private double F1, F2, F3, F4, F5, X1, X2, X3, X4, X5, answer;
    private Button button;
    private Boolean running, answered, first;
    private Font font;
    public Image Img;
    public MediaTracker tr;

    public HookeLaw(){
        running=false;
        answered=false;
        first=true;
        initialize();
    }
    public void initialize(){
        panel = new Panel();
        panel.setLayout(new GridLayout(6,4));
        label = new Label("Force 1:");
        panel.add(label);
        f1 = new TextField(5);
        panel.add(f1);
        label = new Label("Distance 1:");
        panel.add(label);
        x1 = new TextField(5);
        panel.add(x1);
        label = new Label("Force 2:");
        panel.add(label);
        f2 = new TextField(5);
        panel.add(f2);
        label = new Label("Distance 2:");
        panel.add(label);
        x2 = new TextField(5);
        panel.add(x2);
        label = new Label("Force 3:");
```

```

panel.add(label);
f3 = new TextField(5);
panel.add(f3);
label = new Label("Distance 3:");
panel.add(label);
x3 = new TextField(5);
panel.add(x3);
label = new Label("Force 4:");
panel.add(label);
f4 = new TextField(5);
panel.add(f4);
label = new Label("Distance 4:");
panel.add(label);
x4 = new TextField(5);
panel.add(x4);
label = new Label("Force 5:");
panel.add(label);
f5 = new TextField(5);
panel.add(f5);
label = new Label("Distance 5:");
panel.add(label);
x5 = new TextField(5);
panel.add(x5);
label = new Label(" ");
panel.add(label);
button = new Button("Calculate");
button.addActionListener(new buttonMenu(0));
panel.add(button);
label = new Label(" ");
panel.add(label);
button = new Button("Reset");
button.addActionListener(new buttonMenu(1));
panel.add(button);
add(panel);
repaint();
}

public void paint(Graphics g){
    tr = new MediaTracker(this);
    Img = getImage(getDocumentBase(),"Spring.jpg");
    tr.addImage(Img, 0);
    g.drawImage(Img, 20, 200, 50, 150, null);
    g.drawImage(Img, 200, 200, 50, 250, null);
    g.setColor(Color.BLACK);
    g.fillRect(42, 350, 6, 25);
    g.fillRect(222, 450, 6, 25);
    int[] x = new int[3];int[] y = new int[3];
    x[0]=35;x[1]=55;x[2]=45;
    y[0]=375;y[1]=375;y[2]=395;
    g.fillPolygon(x,y, 3);
    g.drawString("Force=0", 25, 405);
    x[0]=x[0]+180;x[1]=x[1]+180;x[2]=x[2]+180;
    y[0]=y[0]+100;y[1]=y[1]+100;y[2]=y[2]+100;
    g.fillPolygon(x,y,3);
    g.drawString("Applied Force", 190, 505);
    g.drawLine(120, 350, 150, 350);
    g.drawLine(120,450,150,450);
}

```

```

g.drawLine(135,350,135,390);
g.drawLine(135,403,135,450);
g.drawString("Distance",110,400);
font = new Font("Arial",Font.BOLD,18);
g.setColor(Color.BLACK);
g.setFont(font);
g.drawString("Spring Constant k:
",25,175);
    if(answered){
        g.drawString("Spring Constant k: "+answer+"
N/m",25,175);
    }
    if(first){
        first=false;
        repaint();
    }
}
public class buttonMenu implements ActionListener{
    private int what;

    public buttonMenu(int a){
        what = a;
    }
    public void actionPerformed(ActionEvent arg0) {
        if(what==0){
            try{
                F1=Double.parseDouble(f1.getText());
                F2=Double.parseDouble(f2.getText());
                F3=Double.parseDouble(f3.getText());
                F4=Double.parseDouble(f4.getText());
                F5=Double.parseDouble(f5.getText());
                X1=Double.parseDouble(x1.getText());
                X2=Double.parseDouble(x2.getText());
                X3=Double.parseDouble(x3.getText());
                X4=Double.parseDouble(x4.getText());
                X5=Double.parseDouble(x5.getText());
                running=true;
            }
            catch(Exception e){
                running=false;
            }
            if(running){
                answer--
((F1+F2+F3+F4+F5)/(X1+X2+X3+X4+X5));
                long x=(long)(answer*100);
                answer=(double)x/100;
                answered=true;
            }
            repaint();
        }
        if(what==1){
            f1.setText("");
            f2.setText("");
            f3.setText("");
            f4.setText("");
            f5.setText("");
            x1.setText("");

```



```
x2.setText("");  
x3.setText("");  
x4.setText("");  
x5.setText("");  
answer=0;  
answered=false;  
repaint();}}}
```

## Appendix 4.f Power of 10 Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

@SuppressWarnings("serial")
public class Power10 extends Applet{
    private Random generator;
    private Panel panel;
    private Button button;
    private ArrayList<Integer> list;
    private ArrayList<Button> blist;
    private int r,times,matchL,matchR,power,total,totaladd,wrong;
    private boolean correct,run,left,right,firstTime,win;
    private Timer time;
    private Label label;
    public Power10(){
        totaladd=-1;matchL=-1;
        matchR=-1;times=-1;
        firstTime=true;left=false;
        right=false;run=false;
        correct = true;win=false;
        list = new ArrayList<Integer>(10);
        blist = new ArrayList<Button>(10);
        blist.add(0,null);
        blist.add(1,null);
        time = new Timer ( );
        time.scheduleAtFixedRate (new ToDoTask(),0,1000);
        generator = new Random();
        wrong=0;
        initialize();
    }

    private void initialize(){
        panel =new Panel();
        panel.setLayout(new GridLayout(6,5));
        Font font = new Font("Arial",Font.PLAIN,18);
        button = new Button(" nano ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0,false,0,0,button));
        button.setBackground(Color.YELLOW);
```

```

        panel.add(button);
        button = new Button("  tera  ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 1, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("    k    ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 4, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("    p    ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 3, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("micro ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 2, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("  pico  ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 3, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("    M    ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 6, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("    f    ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 5, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" kilo ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 4, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("femto ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 5, button));

```

```

        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" n ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 0, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" u ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 2, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" mega ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 6, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("centi ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 7, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" G ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 9, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" T ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 1, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("milli ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 8, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" giga ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 9, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" m ");

```

```

        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 8, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" c ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 7, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("BEGIN");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(1, false, 3, 10, button));
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("RESET");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(2, false, 3, 10, button));
        panel.add(button);
        add(panel);
        repaint();
    }

    public void paint(Graphics g) {
        Font font = new Font("Arial", Font.BOLD, 16);
        g.setFont(font);
        g.drawString("Click on the Prefix and Abbreviation", 85,
200);
        g.drawString("That Matches the Power of Ten Below:", 80,
220);

        Font font1 = new Font("Arial", Font.BOLD, 22);
        if (!win) {
            generate();
            g.setFont(font1);
            g.setColor(Color.RED);
            if (run) {
                switch (r) {
                    case 1:
                        g.drawString("10", 195, 250); power=5;
                        g.drawString("-15", 220, 245); break;
                    case 2:
                        g.drawString("10", 195, 250); power=3;
                        g.drawString("-12", 220, 245); break;
                    case 3:
                        g.drawString("10", 195, 250); power=0;
                        g.drawString("-9", 220, 245); break;
                    case 4:
                        g.drawString("10", 195, 250); power=2;
                        g.drawString("-6", 220, 245); break;
                    case 5:
                        g.drawString("10", 195, 250); power=8;

```

```

        g.drawString("-3",220,245);break;
    case 6:
        g.drawString("10",195,250);power=7;
        g.drawString("-2",220,245);break;
    case 7:
        g.drawString("10",195,250);power=4;
        g.drawString("3",220,245);break;
    case 8:
        g.drawString("10",195,250);power=6;
        g.drawString("6",220,245);break;
    case 9:
        g.drawString("10",195,250);power=9;
        g.drawString("9",220,245);break;
    case 10:
        g.drawString("10",195,250);power=1;
        g.drawString("12",220,245);break;
    }
    g.setColor(Color.BLACK);
    g.setFont(font);
    g.drawString("Time: "+times, 185, 270);
    g.drawString("Number Correct:
++totaladd,145,290);
    }
    if(win){
        g.drawString("Congratulations",145,310);
        g.drawString("You Correctly Matched All 10",100,325);
        g.drawString("In a Total of "+total+"
seconds",120,345);
        g.drawString("With "+wrong+" wrong guesses.",140,365);
        list.clear();
        blist.remove(1);
        blist.remove(0);
        blist.add(0,null);
        blist.add(1,null);
        matchL=-5;
        times=0;
        matchR=-1;
        totaladd=0;
    }
}

public void generate(){
    if(correct){
        r = generator.nextInt(10) + 1;
        Object o = new Object();
        o = (Object) r;
        if(list.contains(o)&&totaladd<9){
            generate();
        }
        else{
            list.add(r);
            totaladd++;
        }
    }
    correct = false;
}

```

```

public class ButtonMatrix implements ActionListener{
    private boolean click;
    private Button buttonPress;
    private int value,l,num;
    public ButtonMatrix(int a,boolean clicked, int b,int c,
Button button){
        click=clicked;
        buttonPress=button;
        value=a;
        l=b;
        num=c;}

    @SuppressWarnings("static-access")
    public void actionPerformed(ActionEvent arg0) {
        int val=0;
        if(this.value==1){
            run=true;
        }
        if(this.value==2){
            win=false;
            run=false;
            if(left){
                blist.remove(0);
                blist.add(0,null);}
            if(right){
                blist.remove(1);
                blist.add(1,null);}
            list.clear();
            left=false;
            right=false;
            repaint();
        }
        if(run&&!win){

            if(this.value==0&&((this.l==0&&!left)|| (this.l==1&&!right))){
                if(!this.click&&val!=1){

                    this.buttonPress.setBackground(Color.CYAN);
                    this.click=true;val++;

                    if(this.l==0){left=true;matchL=num;blist.remove(0);blist.add(0, this.buttonPress);}

                    if(this.l==1){right=true;matchR=num;blist.remove(1);blist.add(1, this.buttonPress);}}
                }
                if(this.value==0&&this.click&&val==0){

                    this.buttonPress.setBackground(Color.YELLOW);
                    this.click=false;

                    if(this.l==0&&left){left=false;;blist.remove(0);blist.add(0,null)
;}}

```

```

    if(this.l==1&&right){right=false;blist.remove(1);blist.add(1,null
);}}

    if(matchR==matchL&&matchR==power){
        blist.get(0).setBackground(Color.GREEN);
        blist.get(1).setBackground(Color.GREEN);
        correct=true;
        firstTime=false;
        Thread thread = new Thread();
        try{thread.sleep(500);}
        catch(Exception e){}
        blist.get(0).setBackground(Color.YELLOW);
        blist.get(1).setBackground(Color.YELLOW);
        left=false;
        right=false;
    }

    if(left&&right&&(matchR!=matchL||matchR!=power||matchL!=power)){
        blist.get(0).setBackground(Color.RED);
        blist.get(1).setBackground(Color.RED);
        Thread thread = new Thread();
        try{thread.sleep(300);}
        catch(Exception e){}
        blist.get(0).setBackground(Color.YELLOW);
        blist.get(1).setBackground(Color.YELLOW);
        blist.remove(1);blist.add(1,null);
        blist.remove(0);blist.add(0,null);
        left=false;
        right=false;
        matchL=-1;
        matchR=-1;
        wrong++;
    }}}}

class ToDoTask extends TimerTask {
    public void run ( ) {
        if(run&&!correct){
            times++;
            repaint();
        }
        if(correct&&!firstTime){
            total=total+times;
            times=-1;
            time.cancel();
            generate();
            time = new Timer();
            time.scheduleAtFixedRate (new ToDoTask(),0,1000);}
            if(totaladd==10){
                win=true;
                totaladd=-1;
                left=false;
                right=false;
                firstTime=true;
                generator = new Random();
                repaint();
            }}}}

```



## Appendix 4.g Power of 10 Abbreviations Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

@SuppressWarnings("serial")
public class Power10Abbrev extends Applet{
    private Random generator;
    private Panel panel;
    private Button button;
    private ArrayList<Integer> list;
    private ArrayList<Button> blist;
    private int r,times,matchL,matchR,power,total,totaladd,wrong;
    private boolean correct,run,left,right,firstTime,win;
    private Timer time;
    private Label label;
    public Power10Abbrev(){
        totaladd=-1;matchL=-1;
        matchR=-1;times=-1;
        firstTime=true;left=false;
        right=false;run=false;
        correct = true;win=false;
        list = new ArrayList<Integer>(10);
        blist = new ArrayList<Button>(10);
        blist.add(0,null);
        blist.add(1,null);
        time = new Timer ( );
        time.scheduleAtFixedRate (new ToDoTask(),0,1000);
        generator = new Random();
        wrong=0;
        initialize();
    }

    private void initialize(){
        panel =new Panel();
        panel.setLayout(new GridLayout(6,5));
        Font font = new Font("Arial",Font.PLAIN,18);
        button = new Button("10^-9 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0,false,0,0,button));
        button.setBackground(Color.YELLOW);
```

```

        panel.add(button);
        button = new Button(" 10^12");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 1, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" kilo ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 4, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" pico ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 3, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-6 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 2, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-12 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 3, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" mega ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 6, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" femto");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 5, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" 10^3 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 4, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-15 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 5, button));

```

```

        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" nano ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 0, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" micro ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 2, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" 10^6 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 6, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-2 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 7, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" giga ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 9, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" Tera ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 1, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-3 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 8, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" 10^9 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 9, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" milli");

```

```

        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 8, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" centi");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 7, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("BEGIN");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(1, false, 3, 10, button));
        panel.add(button);
        add(panel);
        label = new Label("");
        panel.add(label);
        button = new Button("RESET");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(2, false, 3, 10, button));
        panel.add(button);
        add(panel);
        repaint();
    }

    public void paint(Graphics g){
        Font font = new Font("Arial",Font.BOLD,16);
        g.setFont(font);
        g.drawString("Click on the Prefix and Abbreviation", 85,
200);
        g.drawString("That Matches the Power of Ten Below:", 80,
220);

        Font font1 = new Font("Arial",Font.BOLD,20);
        if(!win){
            generate();
            g.setFont(font1);
            g.setColor(Color.RED);
            if(run){
                switch(r){
                    case 1:
                        g.drawString("f",220,250);power=5;
                        break;
                    case 2:
                        g.drawString("p",220,250);power=3;
                        break;
                    case 3:
                        g.drawString("n",220,250);power=0;
                        break;
                    case 4:
                        g.drawString("u",220,250);power=2;
                        break;
                    case 5:

```

```

        g.drawString("m",220,250);power=8;
        break;
    case 6:
        g.drawString("c",220,250);power=7;
        break;
    case 7:
        g.drawString("k",220,250);power=4;
        break;
    case 8:
        g.drawString("M",220,250);power=6;
        break;
    case 9:
        g.drawString("G",220,250);power=9;
        break;
    case 10:
        g.drawString("T",220,250);power=1;
        break;
    }
    g.setColor(Color.BLACK);
    g.setFont(font);
    g.drawString("Time: "+times, 185, 270);
    g.drawString("Number Correct:
"+totaladd,145,290);
    }
    }
    if(win){
        g.drawString("Congratulations",145,310);
        g.drawString("You Correctly Matched All 10",100,325);
        g.drawString("In a Total of "+total+"
seconds",120,345);
        g.drawString("With "+wrong+" wrong guesses.",145,365);
        list.clear();
        blist.remove(1);
        blist.remove(0);
        blist.add(0,null);
        blist.add(1,null);
        matchL=-5;
        times=0;
        matchR=-1;
        totaladd=0;
    }
}

public void generate(){
    if(correct){
        r = generator.nextInt(10) + 1;
        Object o = new Object();
        o = (Object) r;
        if(list.contains(o)&&totaladd<9){
            generate();
        }
        else{
            list.add(r);
            totaladd++;
        }
    }
    correct = false;
}

```

```

    }

    public class ButtonMatrix implements ActionListener{
        private boolean click;
        private Button buttonPress;
        private int value,l,num;
        public ButtonMatrix(int a,boolean clicked, int b,int c,
Button button){
            click=clicked;
            buttonPress=button;
            value=a;
            l=b;
            num=c;}

        @SuppressWarnings("static-access")
        public void actionPerformed(ActionEvent arg0) {
            int val=0;
            if(value==1){
                run=true;
            }
            if(this.value==2){
                run=false;
                win=false;
                if(left){
                    blist.remove(0);
                    blist.add(0,null);}
                if(right){
                    blist.remove(1);
                    blist.add(1,null);}
                list.clear();
                left=false;
                right=false;
                repaint();
            }
            if(run&&!win){
                if(value==0&&((l==0&&!left)|| (l==1&&!right))){
                    if(!click&&val!=1){

                        this.buttonPress.setBackground(Color.CYAN);
                            this.click=true;val++;

                            if(l==0){left=true;matchL=num;blist.remove(0);blist.add(0,this.buttonPress);}

                            if(l==1){right=true;matchR=num;blist.remove(1);blist.add(1,this.buttonPress);}}
                    }
                    if(value==0&&click&&val==0){

                        this.buttonPress.setBackground(Color.YELLOW);
                            click=false;

                            if(l==0&&left){left=false;;blist.remove(0);blist.add(0,null);}

                            if(l==1&&right){right=false;blist.remove(1);blist.add(1,null);}}

```

```

        if (matchR==matchL&&matchR==power) {

            blist.get(0).setBackground(Color.GREEN);
            blist.get(1).setBackground(Color.GREEN);
            correct=true;
            firstTime=false;
            Thread thread = new Thread();
            try{thread.sleep(300);}
            catch(Exception e){}
            blist.get(0).setBackground(Color.YELLOW);
            blist.get(1).setBackground(Color.YELLOW);
            left=false;
            right=false;

        }

        if (left&&right&&(matchR!=matchL||matchR!=power||matchL!=power)) {
            blist.get(0).setBackground(Color.RED);
            blist.get(1).setBackground(Color.RED);
            Thread thread = new Thread();
            try{thread.sleep(300);}
            catch(Exception e){}
            blist.get(0).setBackground(Color.YELLOW);
            blist.get(1).setBackground(Color.YELLOW);
            blist.remove(1);blist.add(1,null);
            blist.remove(0);blist.add(0,null);
            left=false;
            right=false;
            matchL--1;
            matchR--1;
            wrong++;

        }

    }}

class ToDoTask extends TimerTask {
    public void run ( ) {
        if(run&&!correct){
            times++;
            repaint();
        }
        if(correct&&!firstTime){
            total=total+times;
            times=-1;
            time.cancel();
            generate();
            time = new Timer();
            time.scheduleAtFixedRate (new ToDoTask(),0,1000);}
            if(totaladd==10){
                win=true;
                totaladd=-1;
                list.clear();
                generator = new Random();
                repaint();
            }
        }
    }
}

```

## Appendix 4.h

### Power of 10 Prefix Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

@SuppressWarnings("serial")
public class Power10Prefix extends Applet{
    private Random generator;
    private Panel panel;
    private Button button;
    private ArrayList<Integer> list;
    private ArrayList<Button> blist;
    private int r,times,matchL,matchR,power,total,totaladd,wrong;
    private boolean correct,run,left,right,firstTime,win;
    private Timer time;
    private Label label;
    public Power10Prefix(){
        totaladd=-1;matchL=-1;
        matchR=-1;times=-1;
        firstTime=true;left=false;
        right=false;run=false;
        correct = true;win=false;
        list = new ArrayList<Integer>(10);
        blist = new ArrayList<Button>(10);
        blist.add(0,null);
        blist.add(1,null);
        time = new Timer ( );
        time.scheduleAtFixedRate (new ToDoTask(),0,1000);
        generator = new Random();
        wrong=0;
        initialize();
    }

    private void initialize(){
        panel =new Panel();
        panel.setLayout(new GridLayout(6,5));
        Font font = new Font("Arial",Font.PLAIN,18);
        button = new Button("10^-9 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0,false,0,0,button));
        button.setBackground(Color.YELLOW);
    }
}
```



```

        panel.add(button);
        button = new Button(" 10^12");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 1, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("  k  ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 4, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("  p  ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 3, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-6 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 2, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-12 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 3, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("  M  ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 6, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("  f  ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 5, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" 10^3 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 4, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-15 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 5, button));

```

```

        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" n ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 0, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" u ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 2, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" 10^6 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 6, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-2 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 7, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" G ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 9, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" T ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 1, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button("10^-3 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 8, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" 10^9 ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 0, 9, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button(" m ");

```

```

        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 8, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        button = new Button(" c ");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(0, false, 1, 7, button));
        button.setBackground(Color.YELLOW);
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("BEGIN");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(1, false, 3, 10, button));
        panel.add(button);
        label = new Label("");
        panel.add(label);
        button = new Button("RESET");
        button.setFont(font);
        button.addActionListener(new
ButtonMatrix(2, false, 3, 10, button));
        panel.add(button);

        add(panel);
        repaint();
    }

    public void paint(Graphics g){
        Font font = new Font("Arial", Font.BOLD, 16);
        g.setFont(font);
        g.drawString("Click on the Prefix and Abbreviation", 85,
200);
        g.drawString("That Matches the Power of Ten Below:", 80,
220);

        Font font1 = new Font("Arial", Font.BOLD, 22);
        if(!win){
            generate();
            g.setFont(font1);
            g.setColor(Color.RED);
            if(run){
                switch(r){
                    case 1:

g.drawString("femto", 195, 245); power=5;
                        break;
                    case 2:

g.drawString("pico", 195, 245); power=3;
                        break;
                    case 3:

g.drawString("nano", 195, 245); power=0;
                        break;
                    case 4:

```

```

g.drawString("micro",195,245);power=2;
           break;
           case 5:

g.drawString("milli",195,245);power=8;
           break;
           case 6:

g.drawString("centi",195,245);power=7;
           break;
           case 7:

g.drawString("kilo",195,245);power=4;
           break;
           case 8:

g.drawString("mega",195,245);power=6;
           break;
           case 9:

g.drawString("giga",195,245);power=9;
           break;
           case 10:

g.drawString("tera",195,245);power=1;
           break;
           }
           g.setColor(Color.BLACK);
           g.setFont(font);
           g.drawString("Time: "+times, 185, 270);
           g.drawString("Number Correct:
"+totaladd,145,290);
           }
           }
           if(win){
           g.drawString("Congratulations",145,310);
           g.drawString("You Correctly Matched All 10",100,325);
           g.drawString("In a Total of "+total+"
seconds",120,340);
           g.drawString("With "+wrong+" wrong guesses.",145,360);
           list.clear();
           blist.remove(1);
           blist.remove(0);
           blist.add(0,null);
           blist.add(1,null);
           matchL=-5;
           times=0;
           matchR=-1;
           totaladd=0;
           }
           }

public void generate(){

           if(correct){
           r = generator.nextInt(10) + 1;

```

```

        Object o = new Object();
        o = (Object) r;
        if(list.contains(o)&&totaladd<9){
            generate();
        }
        else{
            list.add(r);
            totaladd++;
        }
    }}
    correct = false;
}

public class ButtonMatrix implements ActionListener{
    private boolean click;
    private Button buttonPress;
    private int value,l,num;
    public ButtonMatrix(int a,boolean clicked, int b,int c,
Button button){
        click=clicked;
        buttonPress=button;
        value=a;
        l=b;
        num=c;}

    @SuppressWarnings("static-access")
    public void actionPerformed(ActionEvent arg0) {
        int val=0;
        if(value==1){
            run=true;
        }
        if(this.value==2){
            run=false;
            win=false;
            if(left){
                blist.remove(0);
                blist.add(0,null);}
            if(right){
                blist.remove(1);
                blist.add(1,null);}
            list.clear();
            left=false;
            right=false;
            repaint();
        }
        if(run&&!win){
            if(value==0&&((l==0&&!left)|| (l==1&&!right))){
                if(!click&&val!=1){

                    this.buttonPress.setBackground(Color.CYAN);
                        click=true;val++;

                    if(l==0){left=true;matchL=num;blist.remove(0);blist.add(0,this.buttonPress);}

                    if(l==1){right=true;matchR=num;blist.remove(1);blist.add(1,this.buttonPress);}}

```

```

        }
        if (value==0&&click&&val==0) {

this.buttonPress.setBackground(Color.YELLOW);
        click=false;

if (l==0&&left) {left=false;;blist.remove(0);blist.add(0,null);}

if (l==1&&right) {right=false;blist.remove(1);blist.add(1,null);}}

        if (matchR==matchL&&matchR==power) {
            blist.get(0).setBackground(Color.GREEN);
            blist.get(1).setBackground(Color.GREEN);
            correct=true;
            firstTime=false;
            Thread thread = new Thread();
            try{thread.sleep(300);}
            catch (Exception e) {}
            blist.get(0).setBackground(Color.YELLOW);
            blist.get(1).setBackground(Color.YELLOW);
            left=false;
            right=false;

        }

if (left&&right&&(matchR!=matchL|matchR!=power|matchL!=power)) {
    blist.get(0).setBackground(Color.RED);
    blist.get(1).setBackground(Color.RED);
    Thread thread = new Thread();
    try{thread.sleep(300);}
    catch (Exception e) {}
    blist.get(0).setBackground(Color.YELLOW);
    blist.get(1).setBackground(Color.YELLOW);
    blist.remove(1);blist.add(1,null);
    blist.remove(0);blist.add(0,null);
    left=false;
    right=false;
    matchL=-1;
    matchR=-1;
    wrong++;

    }}

}}

class ToDoTask extends TimerTask {
    public void run ( ) {
        if (run&&!correct) {
            times++;
            repaint();
        }
        if (correct&&!firstTime) {
            total=total+times;
            times=-1;
            time.cancel();
            generate();
            time = new Timer();
            time.scheduleAtFixedRate (new ToDoTask(),0,1000);}
            if (totaladd==10) {

```

```
win=true;
totaladd=-1;
list.clear();
generator = new Random();
repaint();
}}}}
```

## Appendix 4.i Statistical Analysis Tool Java Code

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.ArrayList;

/*
 * StatisticalAnalysis is an applet which takes in data from a user
using a text field.
 * While the data is being entered, it is graphed as a scatter plot,
auto-scaling the plot as
 * values continue to be entered. After all the data is entered, the
user presses the "analyze data" button, and the code
 * determines the standard deviation, average, minimum and maximum
values, and number of terms.
 */

@SuppressWarnings("serial")
public class StatisticalAnalysis extends Applet implements KeyListener{

    private Panel panel;
    private Button b;
    private TextField t1;
    private Label label;
    private ArrayList<Double> list;
    private ArrayList<Integer> rangeL;
    private int keyID,plotx,histogram[],Curterm;
    private double term,sum,max,min,ave,standarddev,range;
    private boolean
exception,analyze,toolarge,toomany,toosmall,analyzed,histogramPlot,scat
terPlot;
    public StatisticalAnalysis(){
        initialize();
    }

    public void initialize(){
        sum=0;
        rangeL = new ArrayList<Integer>(10);
        histogram = new int[10];
        for(int a=0;a<10;a++){
            histogram[a]=0;
        }

        analyzed=false;
    }
}
```



```

        scatterPlot=false;
        standarddev=0;
        keyID = 0;
        analyze=false;
        list = new ArrayList<Double>(10);
        max=-10000000;
        min=10000000;
        panel = new Panel();
        panel.setLayout(new GridLayout(3,2));
        label = new Label("Enter each term here then press enter:");
        panel.add(label);
        t1 = new TextField(5);
        t1.addKeyListener(this);
        panel.add(t1);
        b = new Button("Analyze Data");
        b.addActionListener(new EnterData(1));
        panel.add(b);
        b = new Button("Histogram");
        b.addActionListener(new EnterData(2));
        panel.add(b);
        b = new Button("Reset Data");
        b.addActionListener(new EnterData(0));
        panel.add(b);
        b = new Button("Remove Last Entry");
        b.addActionListener(new EnterData(3));
        panel.add(b);
        add(panel);
    }

    public void paint(Graphics g){
        int x = 0;
        int y = 0;
        g.setColor(Color.BLACK);
        g.drawString("Data Entered:", 25, 100);
        g.drawLine(220, 350, 220, 650);
        g.drawLine(210, 500, 620, 500);
        Curterm=0;
        for(int a=0;a<list.size();a++){
            x=x+15;
            Curterm++;
            if(x>600){
                x=15;
                y=y+100;
            }
            g.drawString(""+Curterm+":
"+list.get(a),25+y,100+x);

        }
        if(tooLarge){
            g.drawString("The number you entered is too
large",325,660);
            tooLarge=false;
        }
        if(tooMany){
            g.drawString("Maximum number of terms
entered",325,660);
        }
    }

```

```

        if(toosmall){
            g.drawString("The number you entered is too
small",325,660);
            toosmall=false;
        }
        plotx=220;
        if(!histogramPlot||scatterPlot){
            for(int a=0;list.size()>a;a++){
                double temp;
                temp=list.get(a);
                if(analyzed){
                    g.setColor(Color.BLUE);
                    if(max>=-min){
                        g.drawLine(220,(int) ((500-
ave*150/(max))+2),610,(int) ((500-ave*150/(max)+2)));}
                    if(max<-min){
                        g.drawLine(220,(int) ((500-
ave*150/(-min))+2),610,(int) ((500-ave*150/(-min))+2)));}
                    g.setColor(Color.RED);
                    if(list.get(a)==max){
                        g.setColor(Color.CYAN);
                    }
                    if(list.get(a)==min){
                        g.setColor(Color.GREEN);
                    }
                    if(max<-min){
                        g.fillOval(plotx,(int) (500-temp*150/(-
min)),5,5);
                    }
                    if(max>=-min){
                        g.fillOval(plotx,(int) (500-
temp*150/(max)),5,5);
                    }
                    plotx=plotx+4;
                }
            }
        }
        if(histogramPlot&&!scatterPlot){
            g.setColor(Color.RED);
            for(int a=0;a<10;a++){
                g.fillRect(220+30*a,500-(10*histogram[a]),30,
(10*histogram[a]));}
            g.setColor(Color.BLACK);
            for(int a=0;a<15;a++){
                g.drawLine(215,490-10*a,225,490-10*a);
                g.drawLine(220+30*a,495,220+30*a,505);}
        }
        if(analyze){
            g.setColor(Color.BLACK);
            y=y+100;
            x=0;
            long l=(long) (standarddev*1000);
            standarddev=(double) l/1000;
            long k=(long) (ave*1000);
            ave=(double) k/1000;
            Font font = new Font("Arial",Font.BOLD,18);
            g.setFont(font);
        }
    }
}

```

```

        g.drawString("Standard Deviation:
"+standarddev,25+y,120);
        g.setColor(Color.BLUE);
        g.drawString("Average: "+ave,25+y,160);
        g.setColor(Color.CYAN);
        g.drawString("Maximum: "+max, 25+y, 200);
        g.setColor(Color.GREEN);
        g.drawString("Minimum: "+min, 25+y, 240);
        g.setColor(Color.BLACK);
        g.drawString("Number of Terms:
"+list.size(),25+y,280);
    }

}

public class EnterData implements ActionListener
{
    private int value;

    public EnterData(int a)
    {
        value = a;
    }

    public void actionPerformed(ActionEvent arg0)
    {
        if(this.value==3&&!list.isEmpty() &&!analyzed) {
            list.remove(list.size()-1);
            repaint();
        }
        if(this.value==1&&!list.isEmpty() &&analyzed) {
            scatterPlot=true;
        }

        if(this.value==1&&!list.isEmpty() &&!analyzed) {
            analyzed=true;
            if(!exception) {
                analyze = true;
                min = list.get(0);
                max = list.get(0);
                for(int a=0;a<list.size();a++) {
                    sum=sum+list.get(a);
                    if(max<list.get(a)) {

                        max=list.get(a);
                    }
                    if(min>list.get(a)) {

                        min=list.get(a);
                    }
                }
                ave=sum/list.size();
                for(int a=0;a<list.size();a++) {
                    standarddev=standarddev+Math.pow((list.get(a)-ave), 2);}

                standarddev=Math.sqrt(((1/(double)list.size())*standarddev));
            }
        }
    }
}

```

```

        repaint();
    }
}

if(this.value==2&&analyzed&&scatterPlot){
    scatterPlot=false;
    histogramPlot=true;
}
if(this.value==2&&analyzed&&!histogramPlot){
    histogramPlot=true;
    scatterPlot=false;
    if(list.size()<10){
        range= (int) ((max-min)/(list.size()/2));
        for(int a=0;a<list.size()/2+2;a++){
            rangeL.add(a, (int) (min+range*a));
        }
        for(int a=0;a<list.size();a++){
            for(int b=0;b<rangeL.size()-1;b++){

if(list.get(a)>=rangeL.get(b)&&list.get(a)<rangeL.get(b+1)){

histogram[b]=histogram[b]+1;}

                }

                if(list.get(a)>=rangeL.get(rangeL.size()-1)){
                    histogram[rangeL.size()-
1]=histogram[rangeL.size()-1]+1;}}
                if(list.size()>=10){
                    range= (max-min)/10;
                    for(int a=0;a<10;a++){
                        rangeL.add(a, (int) (min+range*a));
                    }
                    for(int b=0;b<list.size();b++){

if(list.get(b)>=rangeL.get(0)&&list.get(b)<=rangeL.get(1)){
                    histogram[0]=histogram[0]+1;
                }

if(list.get(b)>rangeL.get(1)&&list.get(b)<=rangeL.get(2)){
                    histogram[1]=histogram[1]+1;
                }

if(list.get(b)>rangeL.get(2)&&list.get(b)<=rangeL.get(3)){
                    histogram[2]=histogram[2]+1;
                }

if(list.get(b)>rangeL.get(3)&&list.get(b)<=rangeL.get(4)){
                    histogram[3]=histogram[3]+1;
                }

if(list.get(b)>rangeL.get(4)&&list.get(b)<=rangeL.get(5)){
                    histogram[4]=histogram[4]+1;
                }

if(list.get(b)>rangeL.get(5)&&list.get(b)<=rangeL.get(6)){
                    histogram[5]=histogram[5]+1;
                }
            }
        }
    }
}

```

```

        if(list.get(b)>rangeL.get(6)&&list.get(b)<=rangeL.get(7)){
            histogram[6]=histogram[6]+1;
        }

        if(list.get(b)>rangeL.get(7)&&list.get(b)<=rangeL.get(8)){
            histogram[7]=histogram[7]+1;
        }

        if(list.get(b)>rangeL.get(8)&&list.get(b)<=rangeL.get(9)){
            histogram[8]=histogram[8]+1;
        }
        if(list.get(b)>rangeL.get(9)){
            histogram[9]=histogram[9]+1;
        }
    }}
}

    if(this.value==0){
        list.clear();
        list = new ArrayList<Double>();
        min=0;
        max=0;
        sum=0;
        ave=0;
        Curterm=0;
        range=0;
        standarddev=0;
        toomany=false;
        toolarge=false;
        toosmall=false;
        histogramPlot=false;
        rangeL.clear();
        for(int a=0;a<10;a++){
            histogram[a]=0;
        }
        t1.requestFocusInWindow();
    }
    repaint();
}

@Override
public void keyPressed(KeyEvent e) {
    keyID = e.getKeyCode();
}

@Override
public void keyReleased(KeyEvent e) {
    if(keyID==10&&!analyzed){
        try{
            term = Double.parseDouble(t1.getText());
            if(term>10000000){
                toolarge=true;
            }
        }
    }
}

```

```

        }
        if(term<-10000000) {
            toosmall=true;
        }
        if(list.size()>79) {
            toomany=true;
        }
    }
    catch(Exception a){
        exception =true;
    }
    long l=(long) (term*10000);
    term=(double)l/10000;
    if(!exception){
        if(!toolarge&&!toomany&&!toosmall){
            list.add(term);
            if(term>max) {
                max=term;}
            if(term<min) {
                min=term;
            }
        }
        t1.setText("");
        t1.transferFocus();
        repaint();
    }
    exception=false;
    t1.requestFocusInWindow();

    }
}

@Override
public void keyTyped(KeyEvent e) {

}
}

```