PMKS+: Enhancements

A Major Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
degree of Bachelor of Science

Written By:

Trevor Dowd (CS)

Robert Dutile (CS)

Haofan Zhang (CS)

Advisors:

Professor David C. Brown (CS)

Professor Pradeep Radhakrishnan (ME)

May 18, 2019

I. Abstract

The goal of this project is to provide new features for, and enhancements to, the previously developed Planar Mechanism Kinematic Simulator Plus (PMKS+) application. Following user interaction guidelines, different features were developed and implemented to enhance the existing application, so that users can work more efficiently and effectively. Testing was completed to find errors in the application. In addition, user evaluations were conducted that proved the effectiveness of the implemented changes.

II. Acknowledgements

The team would like to thank WPI's Mechanical and Robotics Engineering students for testing the application and providing feedback. We would also appreciate the work of Alex Galvan for helping with the implementation of the forces in the application.

Finally, we would like to thank our advisors, Professors Brown and Radhakrishnan. Without their continuous guidance and help this project would not have been possible.

III. Table of Contents

IV. Authorship

The following table denotes the original authorship of the chapters of the paper. After the initial writing, all team members reviewed and edited the other sections.

Chapters	**Authors**
Abstract	Trevor Dowd
Acknowledgements	Trevor Dowd
1	Trevor Dowd
2	Haofan Zhang
3	Robert Dutile
4	Trevor Dowd, Robert Dutile and Haofan Zhang
5	Trevor Dowd and Haofan Zhang
6	Trevor Dowd
7	Trevor Dowd and Haofan Zhang
8	Haofan Zhang
9	Robert Dutile
10	Trevor Dowd
11	Trevor Dowd

1. Introduction

As we are in a world with many technologies at our disposal, it is important for an application to not only be functional, but also user friendly. Using standards from the user experience community, an existing application can be enhanced, so users can understand how to use the application more efficiently and effectively. By adding features following the principles of user-centered design, users will have a more positive reaction to the application.

The existing Planar Mechanics Kinematic Simulator Plus (PMKS+), developed by Appikatla, et al. as part of their major qualifying project (MQP) at Worcester Polytechnic Institute (WPI) in 2019, is a functional web-based application for linkage analysis that was developed as an improvement over an earlier version, PMKS, developed by Prof. Matthew Campbell (currently Professor of Mechanical Engineering at Oregon State University) and enhanced by students and faculty at WPI. It assists with the simulation of linkages to help students better understand the construction and modeling of such linkages. PMKS is used in various Mechanical Engineering courses at WPI. The current PMKS+ application was developed since the original PMKS is Microsoft Silverlight-based, which is no no longer supported by major browsers. Building on the PMKS+ application developed in 2019, this project's goal is to improve the application by adding features that could not be added in the previous project. This new application will be called PMKS+W as it is a web-based application. By adding these features, the aim is to make the application easier to use for students and to expand the functionality for the courses that will use PMKS+.

The following chapter will discuss our research statement and the goals of this project. Chapter two will describe the goal and purpose for this project. Chapter three will go into depth on what the PMKS+ application does and what purpose it serves. Chapter four will provide a deep literature review based on the team's research into how to improve the user experience and user interface for PMKS+. Chapter five will detail the methodology that we used to accomplish the goals set forth in chapter three. Chapter six will detail all of the new features that this project added to the previous version of PMKS+. As a part of all these features, a new interface design was implemented. The design process and iterations of the user interface design will be discussed in chapter seven. The design of the system itself and the implementation of those designs will be discussed in chapter eight. The ninth chapter of the report will discuss the testing process for the application. An evaluation of the application by mechanical engineering students was produced and will be discussed in chapter ten. Chapter eleven contains the conclusions of the report and will discuss the proposed future work for the project. Finally, we will list the references for the entire report at the end of the paper.

This report documents the progress of the group as we improved upon the existing PMKS+ application. Through these next chapters, the hope is to show the research and reasoning into the decisions to make additions and changes to the existing application.

2. Research Statement and Goals

PMKS+ requires additional features to become usable for our clients, who are mechanical engineering students and faculties studying or teaching linkage design. Features that exist in the original PMKS such as static, dynamic and kinematic analysis, adding force on linkage, and the ability to create sliding joints, are not present on PMKS+. These features are used extensively by WPI students in their coursework and consequently are needed in our revised system.

PMKS is feature packed, but its interface is outdated, and through user testing the user experience was found to be poor (Appikatla, et al. 2019). PMKS is also limited by Silverlight, a deprecated application framework created by Microsoft, restrictions such as not being able to show a context menu on mouse right click.

Our purpose for this project is to provide greater functionality to PMKS+ by implementing features that exist in PMKS but are missing in PMKS+, as well as by adding new features. Missing features include sliding joints, speed and direction control for animations, editable joint table, ability to add forces, and linkage import and export using file or URL. New features include link shapes, help and tutorial. We need to make sure that existing PMKS users can understand how to use newly added features on PMKS+ within a short amount of time. We would also like to revise the interface. This would allow us to fix existing UI design issues, and add new interface components for new features without interfering with the existing UI design decisions.

To summarize, we want to:

1. Implement all the features that exist in PMKS but not in PMKS+
2. Add new features
3. Improve the user experience with PMKS+ features, while still making the transition to the new application feel natural for existing users of PMKS (The features mentioned will be explained in Chapter 6)
4. Fix existing issues with the PMKS+ application and interface (See Chapter 4.2.2)
5. Add new interface components (for new features) without interfering with the existing design

3. What is PMKS+?

PMKS+ (Planar Mechanical Kinematic Simulator Plus) is a web application developed to help students and teachers design and simulate simple planar mechanisms. Planar mechanisms have all important movement occurring in a two dimensional plane. The program supports creation, simulation and animation of these mechanisms. It is an adaptation and upgrade of an older system, PMKS, which has been used for several years by the WPI Mechanical Engineering Department, but is now becoming obsolete.

PMKS was first developed by Dr. Matthew Campbell of Oregon State University and expanded by students and faculty at WPI. Appikatla, et al. (2019) worked on the first iteration of PMKS+ where they ported PMKS to an Angular/Typescript based application, which lets it run on most browsers. They put in significant effort to make the user interface more accessible and easier to understand compared to the original PMKS Still, as of the beginning of this project, several desired features have yet to be adopted from the original PMKS program.

4. Literature Review

The goal of this section is to explain any background information that will be pertinent to the rest of the paper, using a variety of source materials.

4.1 Introduction of Literature Review

The goal of this section is to explain any background information that will be pertinent to the rest of the paper, using a variety of source materials. From the information presented in the MQP report *PMKS+: Recreating a Legacy Application* (Appikatla, et al. 2019), there was a base of knowledge for this project to expand on. The report goes into detail about why the team made certain selections for the user interface of the application. The report also details the structure of the code which will be important as the team expands on the application from last year. This also includes research into certain technologies that were used in the development of the application. Research into the standards and best practices for user interface design and testing was conducted to help guide the team's efforts in creating a practical and easy to use interface for the application. The research conducted by the team includes studying interface evaluation and usability metrics. The next item that the team researched was usability testing of an application. This helps the team better understand how to conduct an evaluation as well as how to measure the information received from the user evaluations.

4.2 PMKS+: Recreating a Legacy Application

The group who worked on PMKS+ provided a report which provided detailed explanation about their UI design principles, program architecture and technology used (Appikatla, et al. 2019). This information is useful to us for us as we are building on top of PMKS+.

4.2.1 User Interface Decisions

When improving upon an application, it is important for developers to look at documentation created by previous developers to ensure new designs are consistent with existing designs. This way the application will give a consistent experience. The decisions below were made during UI design iterations of PMKS+ (Appikatla, et al. p.44).

Toolbar: Figure 4.1 shows the toolbar located at the top of the application. This provides an area that only handles non linkage manipulating actions. One tab is always expanded for quick access. Action buttons are grouped under tabs because they have similar functionality and could therefore be grouped to conserve space (Appikatla, et al. p.60).

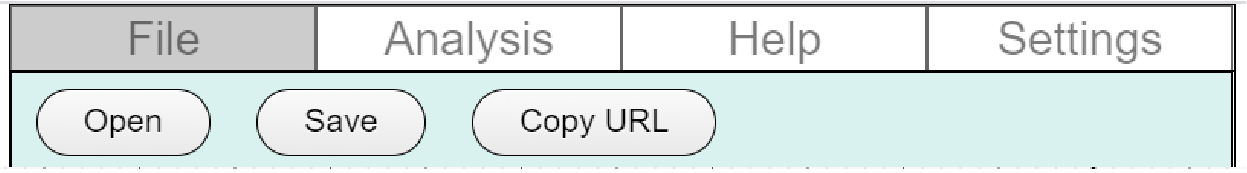


Figure 4.1: Toolbar located above the grid

Consistent style for non-linkage manipulating action buttons: All non-linkage manipulating action buttons have rounded edges, as well as hover feedback and click feedback. They wanted to make sure that users could tell them apart from the tabs, so they decided to give them a different but consistent design (Appikatla, et al. p.59).

Context menu for grid action involving selection: A context menu is a list of actions doable when selecting an object. It is hidden from the interface, but will appear when the user clicks on an object. Figure 4.2 shows the context menu used in PMKS+ to provide selection on joints to perform actions such as linking multiple joints that already exist on the grid. It is used in many other applications, as it is more descriptive and intuitive than other methods since the user can select and manipulate on the grid.

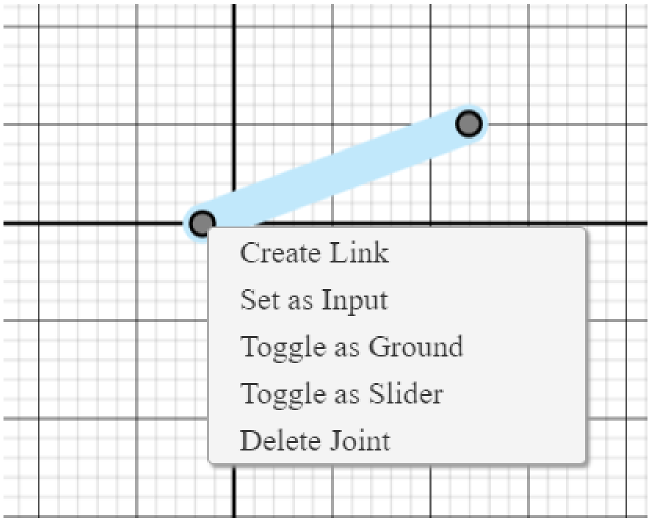


Figure 4.2: The context menu for the PMKS Application

Click-click linkage creation: PMKS+ uses the click-click linkage creation method (Appikatla, et al. p.67). In order to create a link, the user has to click to mark the starting point and click again to mark the end point. Click-Click was the default option in PMKS+ because it proved to be the most usable linkage creation method that worked on both mouse and trackpad (Appikatla, et al. p.69).

Theme color: The colors for PMKS+ are designed to match the colors of the logo. It is a practice that many well established web pages use, such as Facebook and Youtube.

Animation bar: In the original PMKS, the animation bar is vertical and on the right side of the screen. For PMKS+, it is now horizontal in order to mimic a chronological timeline, and therefore stretches from left to right (Appikatla, et al. p.52). The top of the interface is occupied by the toolbar, thus the animation bar is at the bottom.

4.2.2 Design Features That are Inconsistent

As we checked through the PMKS+ documentation, we discovered some interface designs that violate the principle of consistency, which will be further considered in section 4.3.

No consistent place/style for grid action buttons: Actions that don’t change the linkage (such as zooming and centering the grid) are grouped together with actions that change the linkage (such as clearing the grid). These buttons are not in the context menu and interact very differently from using a context menu. They also have a different style than the play/pause button for the animation bar.

The toolbar tab “Get Url” is an action, while all other titles are objects: This happened because Open File and Save File were originally grouped under File tab in iteration 6, but changed later (Appikatla, et al. p.60). This leaves all other actions under a category, except “Get Url”.

4.2.3 PMKS+ Architecture

PMKS+ follows the Model, View, Controller (MVC) model. Figure 4.3 shows the general model used in PMKS+. The model includes the simulator, and object classes used by the simulator, such as Link and Joint. The main controller is the MainPage component script, which manipulates the model and view. It also synchronizes data between view components and model using 2-way data binding, a feature provided by Angular (Wikipedia, 2020a). The views are Angular view components, each comes with an HTML file, and a CSS stylesheet. Each component also has a Typescript sub-controller.

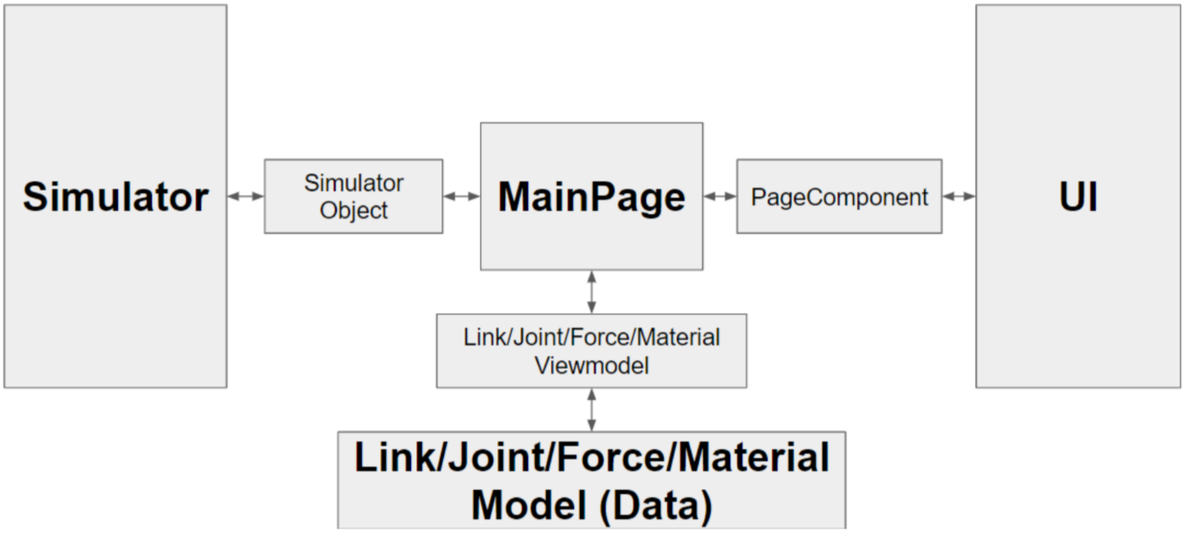

Simulator
Simulator Object
MainPage
PageComponent
UI
Link/Joint/Force/Material Viewmodel
Link/Joint/Force/Material Model (Data)

Figure 4.3: System architecture of PMKS (Appikatla, et al. p.22)

4.2.4 Technologies Used in PMKS+

Angular: A TypeScript-based open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations (Wikipedia, 2020a). Angular's architecture involves Templates, Directives, Components, Services and Modules. A Template forms the layout and UI of the application, and Directives are written inside the Template to provide basic interactive actions to the interface. A Component acts as a controller and communicates with the Template using event and property binding. Usually the Template triggers an event, which is often due to user input, and the Component handles it. When a Component's property changes, the Template will be updated automatically. Services and Modules provide additional functionalities when injected or loaded.

TypeScript: An open-source programming language developed and maintained by Microsoft (Wikipedia, 2020b). It is a strict syntactical superset of JavaScript, and adds optional static typing to the language (Wikipedia, 2020b). TypeScript allows explicit definition of classes, interfaces, enums, generics, namespaces and more features that are available for object oriented languages such as Java. It is the default language for Angular. The Angular Command Line Interface (CLI) tool can translate TypeScript into JavaScript, which can then be interpreted by modern browsers.

HTML/CSS: Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML (Wikipedia, 2020a). An Angular component is built using HTML, CSS and TypeScript.

4.3 User Interface Design

This section discusses the various aspects of User Interface Design that the team considered as we started development of PMKS+W, the version we would be creating.

4.3.1 UI Design Principles

Besides the report of the previous group, we have researched various topics important to this project, perhaps most pertinently the principles of User Interface Design. Some of our sources were the same as those of the previous group, but we added others to acquire a broader view of the subject matter.

In “User Interface Design Principles For Interaction Design,” Blair-Early, Adream, and Zender (Blair-Early et al., 2008) lay out a series of primary principles for the design of User Interfaces, including:

- *Design an Obvious Starting Point:* Give the user an obvious place to start their exploration of the interface;

- *Design an Internally Consistent Logic for Content, Action and Effects:* Make sure that there are clear, easy to parse patterns in the interface that remain consistent throughout, to aid the user in learning about it;
- *Identify and Consider the Impact of Familiar Interface Conventions:* Keep in mind what the user might consider to be obvious meanings for symbols, color associations, etc;
- *Design Tangible Responses to Apt User Actions:* Give the users immediate feedback when they take an action, so that they know they've done something;
- *Design Interface Elements in Consistent Proximity to Their Content Objects and Each Other:* Make sure that the user doesn't have to move around the interface too much to do similar actions;
- *As Necessary, Provide a Readily Accessible Overall Mechanism for Assistance:* Make sure there's an easy to spot and use way for the user to get help if they need it;
- *Design Interface Elements that Minimize Interface and Maximize Content:* Where at all reasonable, avoid letting the interface get in the way of content;
- *Use Metaphors Where Content is New, Obscure, or a Narrative-Based Visual Metaphor:* Use simple metaphors to give the user a more concrete model to interact with for the more abstract concepts.

These principles allow for smoother interaction between users and the program, and allow the interface to be learned more quickly and easily.

Further principles more relevant to the application of such ideas are explained by Spolsky in *User Interface Design for Programmers*, (Spolsky, 2001) such as:

- The importance of *Activity Based Design*, where you make sure to keep in mind what a user might actually be wanting to do overall, so that you can make goals as simple to achieve as possible, without getting too bogged down in any particular feature.
- The utility of developers seeing a user interact with the program on their own, so that they can get a better idea of how the users would see your system, as opposed to how you think they would see it.
- The pitfalls of disconnects between the perspectives of users and of programmers: the latter are inevitably much more familiar with the interface, since they've likely been designing and redesigning it for a while.
- The necessity of awareness that any metaphors or symbolism you use may have unintended consequences or interpretations, which can throw the intended purpose off track

Krug gave the much quoted central principle of many interface designs in his book, *Don't Make Me Think* (Krug, 2006), which was fittingly focused on the primary goal of allowing the user to think as little as possible about what they need to do to accomplish what they want. This makes for a smoother, less frustrating experience working with the program, and reduces the learning curve significantly.

4.3.2 Industry Standard Practices

PMKS+ is a Single Page Web Application, for which there is a set of best practices available. In "User Interface Design Practices in Simple Single Page Web Applications" (Tesarick et al., 2008), the authors lay out a series of recommendations for working on single page Web Applications, such as:

- *Details on Demand:* To avoid overly cluttering the limited screen space, it is best to provide low effort ways for the user to gain additional details about an application on demand, such as by implementing tool-tips or pop-up menus.
- *Actions on Demand:* Similarly, if an object in your interface has many possible actions a user might perform upon it, it is recommended to find some way to move many controls into some kind of hidden area where they won't take up screen space until they are desired, such as by using a pop-up or drop-down menu.
- *User Driven Page Components Visibility:* It is considered wise to let the user suppress or move out of the way portions of the page that they do not need at the current moment.
- *Dedicated Mode:* It is recommended to use modal forms and similar such techniques for more complicated actions, when necessary.
- *Multifunction Controls:* When dealing with a more dynamically developed program, it is wise to consider blending new features together with old ones, to allow for a more harmonious control scheme.

4.4 UI Interface Evaluation

It is of course a rather pointless endeavor to try to improve a user interface without testing that you are actually creating a substantial improvement. Indeed, altering a user interface without actually checking that you've had the expected effect can lead to untested ideas proliferating just because people assume they work or are necessary somehow (Spolsky, 2001). So, if one wants to make an interface better, there needs to be a set of clear, testable criteria by which the designer can discern the effectiveness of their work.

4.4.1 UI Testing Criteria

The best way to test is to just observe new users working with the program, though this is sometimes impractical. This is because acquiring users from your target audience, setting up tasks for them, and having them run through while you watch is time-consuming, and requires a fully functional, already implemented program. There are several short cuts, such as the slightly more time-consuming but much less technically demanding paper prototype. Proper experimental procedures should be observed as much as possible, so as to avoid contaminating results. Fortunately, for basic testing, relatively low sample sizes (e.g. about 6 people) tend to be fairly acceptable, if all that is desired is to find flaws in your design (Spolsky, 2001).

4.4.2 Target User Base

For the purposes of our project, our target audience is Mechanical Engineering students. We will prefer volunteers who have never used PMKS+ or PMKS.

4.4.3 Usability

Usability is the most common issue by which a user interface is evaluated. It is in general an estimate of both how easy it is for a new user to learn what they need to do, and how much effort it takes for a user to complete a task. Learnability is really only discernible through testing and prior experience, but a reasonable estimate of the effort can be acquired by counting the number of discrete actions a user is required to make to complete a task.

4.5 Application Testing

From a young age, students are told that the only way to achieve quality writing is to have others edit and critique the work. The design of a user interface is no different. The only way to create a quality application is to have others critique the application (Roose, 2017). Because the designer created the application, they will have a severe bias towards liking and understanding how the features work (Chi, 2018). The only way to determine what works in an application and what does not is to do extensive user testing. Without extensive testing, the application will likely be buggy and user-unfriendly. These issues can be three to four times more expensive to fix after the application release, meaning that user testing can save on time and money (Vogels, 2019). User testing is what allows people designing and creating applications to make a product that puts the users needs and wants first.

User Testing can encompass many ideas including product testing, design testing, usability testing, and application testing, but they are all defined the same way (Roose, 2017). User Testing is the way that a designer can test the application with an unemotional and unbiased user base to get feedback for the usability and functionality of the application (Chi, 2018). Through User Testing, a designer can get feedback that is invaluable to the design of the application.

There are many different ways to conduct User Testing, but Roose (2017), in the article *How to Conduct Usability Testing in Six Steps,* describes a simple and concrete way to conduct a user test. The first step is to create a testable prototype. This prototype can be as simple as a sketch or as complex as a high-fidelity mockup that is responsive. From this prototype, a testing plan must be created. When creating this plan, there are a couple of things that must be considered to create a successful test, such as what is being tested, how it will be tested, what questions will be answered, and how the tester will organize the results. The third thing that must be done is selecting a random user base that represents the target audience for the application.

Now with a group selected, a suitable location must be selected to conduct the test. The location should be a place where both parties can meet to conduct the test in person, and it should be a place where both people are comfortable; Roose (2017) suggests a cafe as the ideal spot. The next step would be to conduct the test. To make sure the test results are unbiased and a correct representation of the state of the application it is important to make sure that when the test is conducted that the test conductor does not lead the user to certain answers. A great way to make sure this happens is to practice answering questions a user might have in an unbiased way before conducting the test. The final step is to analyze and document the results of the test (Roose, 2017). If the test was carried out carefully and correctly, these results will be invaluable when making changes to the application.

After User Testing, there are some more types of testing that must be considered to help avoid errors and issues when the application is released. Another important type of testing that must be conducted is compatibility testing. Compatibility testing is done by testing the application on a wide range of browsers (i.e., Chrome, Microsoft Edge, Firefox, Safari, etc.) and operating systems (i.e. Windows, Linux, iOS, etc.), so that the application functions with any of these browsers and operating systems (Vogels, 2019). Performance Testing, a way to test how an application handles stressful situations like poor internet connection and a surge of users, should also be considered before releasing an application, so that the application can function under suboptimal circumstances (Vogels, 2019). Taking these testing types into consideration, it can help ensure that the release of an application is smooth and limited in errors.

5. Methodology

In this chapter we describe the methods by which we pursued the goals laid out in Section 2. We also record our decision making process therein.

5.1 Literature Research

The team first conducted a review of relevant literature that is related to our project. We started by reviewing the report from the previous MQP that this project is building on, *PMKS+: Recreating a Legacy Application*. After reviewing that report, we investigated background information on the technologies used in the application. From there, the team researched the aspects of user interface design. We researched best practices and industry standards. Different types of application testing was the last item that we researched. This was to help understand the best ways to evaluate and test our application.

5.2 Requirement Finding

After the literature review, one of the most important duties we pursued, was establishing the requirements for our project. Over the course of much of A-term we gathered information through several methods, including:

- Comparing the capabilities and features of PMKS and PMKS+, trying to see what traits the latter was lacking that could be found in the former.
- Considering suggestions for further work by the previous project team (Appikatla, et al. 2019)
- Brainstorming amongst ourselves on potentially useful features for the program.

- Having several meetings with Professor Radhakrishnan about what he wanted from the program.

Furthermore, we sought to organize these desired features and functionalities, based on their importance, and the expected difficulty of implementing them. The hierarchy of importance for these requirements was primarily hashed out through another meeting with our advisors, while difficulty was estimated after reviewing our analysis of the existing code. This gave us a clear set of goals which we could use to plan the rest of our work on the project.

5.3 User Interface Design Iterations

Throughout the project, the team developed many iterations of the user interface design. These iterations were analyzed and built upon throughout the process of the project. The evaluation of these interfaces informed decisions that lead to the final user interface that is used in the application of the project.

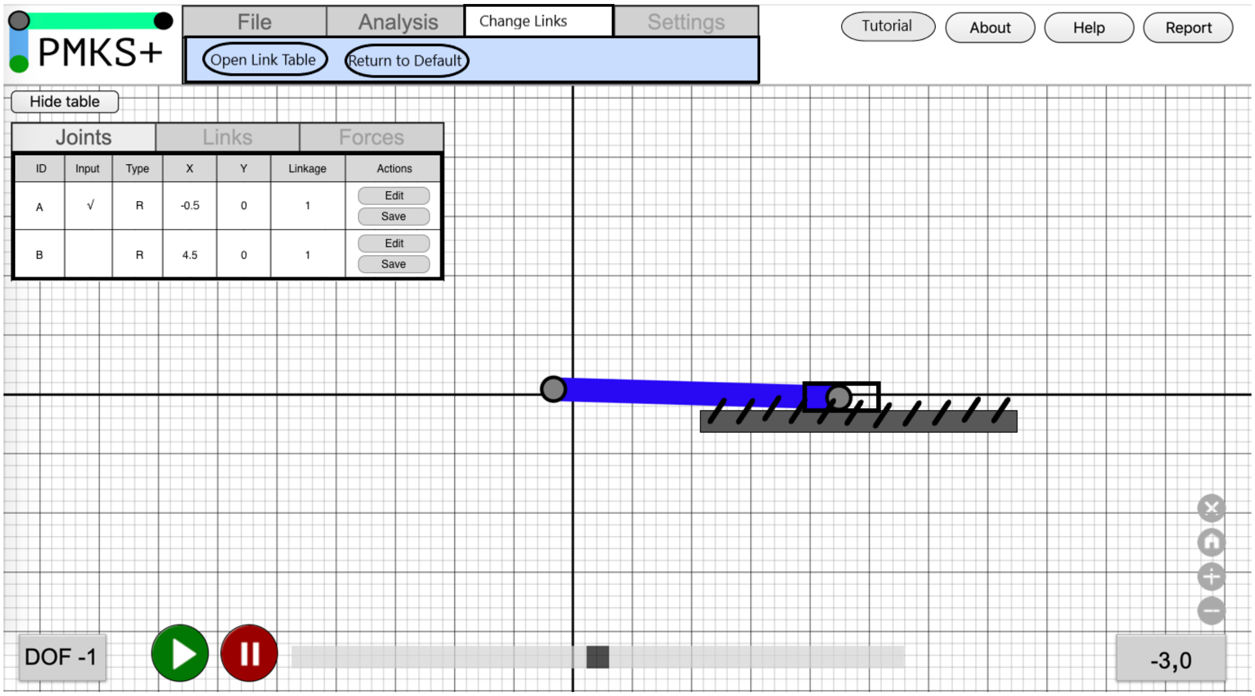
5.3.1 Pre-Iteration

Before the first UI design iteration, the team reviewed the list of features that were previously gathered. As we reviewed these features, we documented the possible ways of implementing the features that we wanted to add to the system. After the possibilities had been listed, we listed pros and cons for the features. These guided and informed the team as we made decisions about what we wanted to do for our initial iteration, and possible options for future iterations of the interface design. The reasonings and thought process behind all of the decisions made in each of the iterations are documented in Chapter 7: User Interface Design. The following sections will document the features added, edited, or removed in each of the iterations.

5.3.2 UI Design Iteration

For the first iteration, we prioritized the features that were stressed as being important in our requirements gathering. We start every design with a mockup for each feature. For an example, see figure 5.1. The first thing that was addressed was the ability to access a tutorial. The tutorial in this iteration was accessed by pressing a button in the top right of the application. This opened a new tab that provided a video giving a step by step tutorial of how to use PMKS+W. The second feature added was the sliding joint, see figure 5.1. This was accessed by either right clicking on the joint to set it as a sliding joint or by pressing Edit” in the table and changing the type to “S". Another feature added in this section was the ability to edit the location, types, and IDs of the joints by directly editing the table. This action can be performed by clicking the “Edit” button, then typing the change into the table. Finally, pressing “Save” makes the changes apply to the linkage.

The “Get URL” tab was removed from the design and was moved to be a button under the “File” tab. The tab was replaced by a new tab, “Change Links”, which allows the user to change the types of links that the user can create: for example instead of just being able to make straight line links, the user can make links from a preset list of shapes. The user can change the link by selecting “Open Link Table” and selecting a link. Then the user can change the size and rotation of the link. When the user wants to return to the default links, the user presses return to default.



ID	Input	Type	X	Y	Linkage	Actions
A	√	R	-0.5	0	1	Edit Save
B		R	4.5	0	1	Edit Save

Figure 5.1: Mockup design for a tab to change the linkage, the new sliding joint, the ability to edit the joints by typing, and a tutorial button.

From this design, the team conducted an internal heuristic evaluation for the mockup. This evaluation had each team member rate on 12 heuristics of the design on a scale from one to ten. The scores for each heuristic were averaged from the three team members' scores and a total score was calculated for each feature. Chapter 7: User Interface Design will go into greater detail about each heuristic that was analyzed and how the team scored each feature.

5.4 Implementation

This section will discuss the implementation that the team took when making the new application.

5.4.1 New Table Design

One of the things that needed to change was the implementation of the table. The former table only displayed the joints and offered no interactions with the rest of the application. The first two things that we wanted to add were the link and the force tables. The link table allows the user to see what type the links are and the joints that comprise them. The force table displays the location and magnitude of the force on a specific link.

The next update to the table allows the user to edit, create, or delete joints directly from the table. This allows the user to set exact coordinates in the application, and allows greater control over the application.

The changing of the table turned out to be more difficult than expected as it required not only changing the table, but also changing how links are created. Because of this, we had to wait for link and joint creation to be finished before being able to finish the table.

5.5 User Evaluation

For user evaluation, we worked with another PMKS team working on a desktop version of the application. Based on the advice of our advisors, we decided to compare the two applications, PMKS+W and PMKS+D (a desktop version of the application being created by another MQP team), to another existing application to see how it compares to other apps. The application that was selected was the Working Model, requested by Professor Radhakrishnan. After that, the two teams gained some familiarity with the Working Model application. Then, we started with developing what we wanted to test and how we wanted to test the applications.

The groups knew that we needed to come up with ways to combat the issues that COVID-19 created. We would no longer be able to conduct the in-person testing that we wanted to do. Instead, we decided that a Zoom meeting and a Google form would have to stand in for the testing of the application. The two teams determined that this would be the best way to conduct the testing.

The Google form served as the guide for the user when they are conducting the test. For as many users that we could, one of the team members would be in a zoom meeting with the user. This allowed us to get detailed information about all of the tests that we could, but reach everyone regardless of schedule. The user would then be using a remote desktop that will have all the applications needed to conduct the test. The Google form would then step the user through the processes that the team wanted to test. The user would take a pre-survey that would allow us to get a baseline understanding. Then, the users would use either the PMKS+W or the Working Model application, assigned at random, and complete all the steps outlined that test things like linkage creation, editing, animation controls, and the forces. Then, the user would use the other application. Once they complete the two tests, the user would fill out the post-survey questionnaire.

The teams decided that this was the best course of action considering the limitations brought on by the COVID-19 virus. This allowed the teams to obtain as much information as possible within the constraints of the pandemic and the shutting down of WPI’s campus.

6. New Features

This section will discuss the features that we, the team, wanted to implement over the course of the project. This chapter serves as a preview to the design and implementation sections shown in the following two chapters.

6.1 Speed and Direction Control

The first thing we wanted to implement was the speed and control of the linkage. This would allow the user to have greater control over the animation. This new feature was implemented so that the user could make the simulations more like real life. The speed and direction was something that a person could control when they were making a linkage in the real world, so translating the real world property to a virtual environment seemed like a natural step in improving the functionality of the application.

6.2 Table Editing

Another feature to be added was the ability to edit the joints of a linkage from the table. This is necessary because it is the only way to ensure exact positioning of the joints in a linkage. This addition allows the user to have more than one option for joint editing.

6.3 Force Implementation

Forces are an important part of linkages. Including them into the interface allows the user to see how certain forces, such as gravity, would interact with the linkage design. This meant that having forces in the application was extremely important. This feature allows the user to add forces to the links that are a part of the linkage.

6.4 Shape Creation and Editing

Links do not only come in the shape of a bar. Because of this, we created a way for the user to change the shapes of the links in the linkage to different shapes. This allows the user greater control over the simulation. It also opens up many new linkages for the user to be able to simulate with the addition of new shapes.

6.5 Help Section

With all the new features, we thought that users could get lost in the application. It was decided to add a new help section as a way to help the user navigate the application.

6.6 Import and Exporting

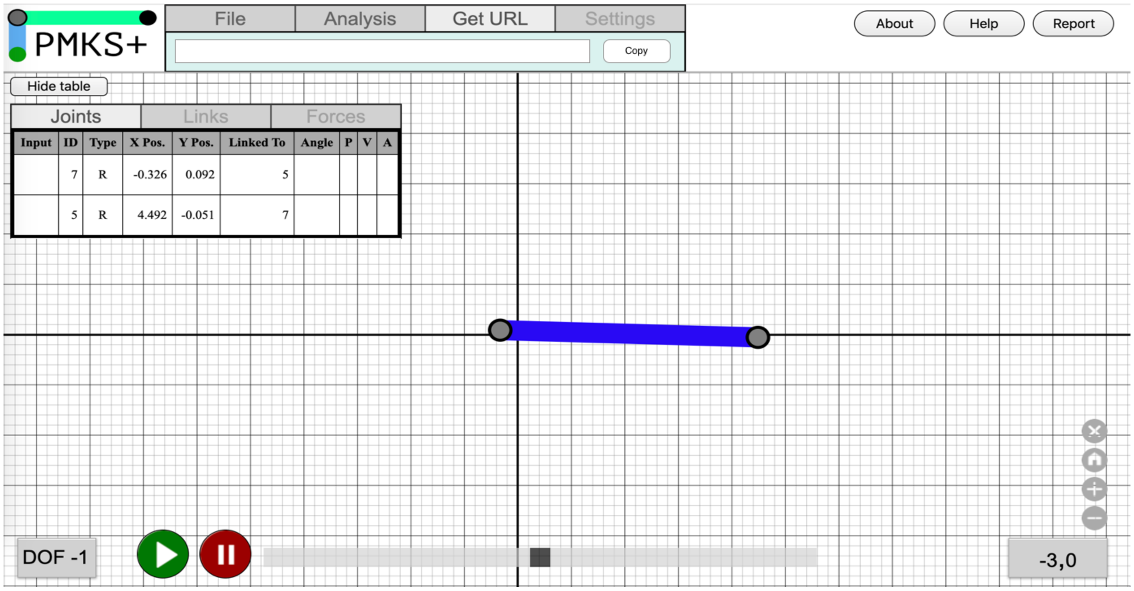
It was also determined that a user should be able to save and reload the linkages that they create. We also wanted a way to share linkages between people. A new way of importing and exporting designs was added with the Get URL feature. That allows the user to save and share URLs with others, so that the linkage could be reloaded at a later date.

7. User Interface Design

In this section, we describe the details about how our features look and interact with users. For the detail implementation (how we made our features to work), see Chapter 8: Feature Implementation.

7.1 Get URL

Design 1:



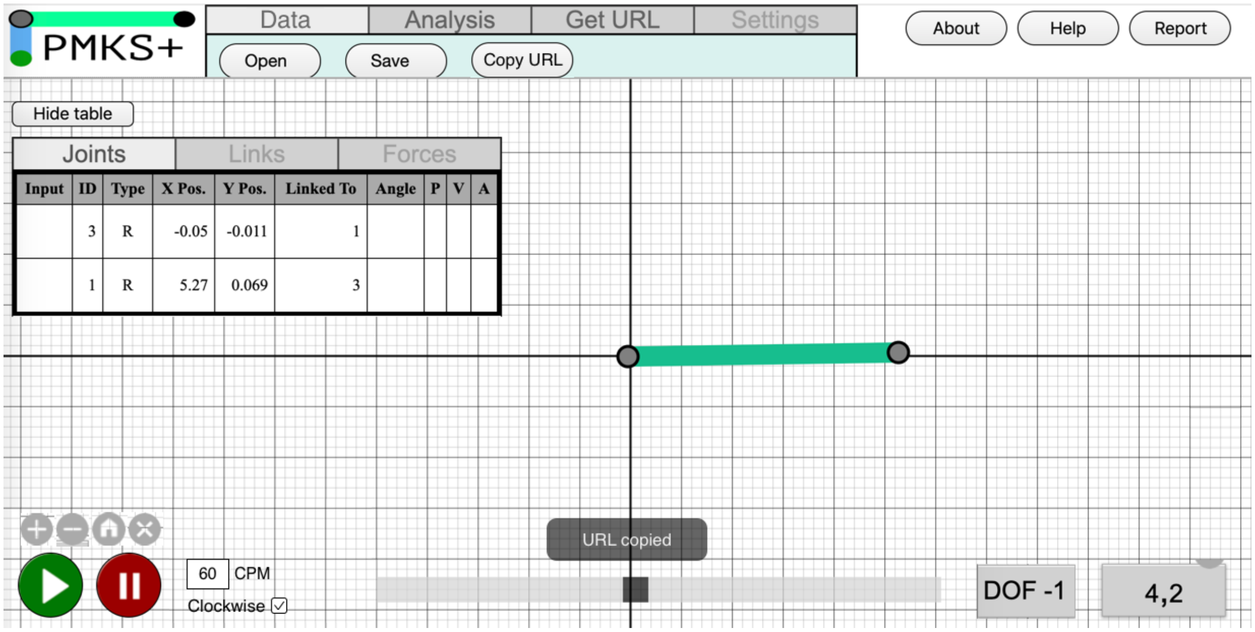
Input	ID	Type	X Pos.	Y Pos.	Linked To	Angle	P	V	A
	7	R	-0.326	0.092	5				
	5	R	4.492	-0.051	7				

Figure 7.1: Get URL design 1

First we designed the feature to behave the same way as in PMKS Silverlight: “Get URL” has its own tab, see Figure 7.1. By clicking on the tab, the full url is shown in a text box below the tab on the left, with the copy button on the right. Users can either click the copy button to copy the URL, or view/edit the URL in the textbox.

This design had a few flaws. The major one being that the URL will be extremely long and not very human readable, so allowing users to see or even edit it can lead to confusion and user errors. The question discussed was does the user really need to see the URL, and the answer, for our target user is no. “Get URL” is also an action, while every other tab is a noun, which is inconsistent.

Design 2:



Input	ID	Type	X Pos.	Y Pos.	Linked To	Angle	P	V	A
	3	R	-0.05	-0.011	1				
	1	R	5.27	0.069	3				

URL copied

60 CPM

Clockwise ☑

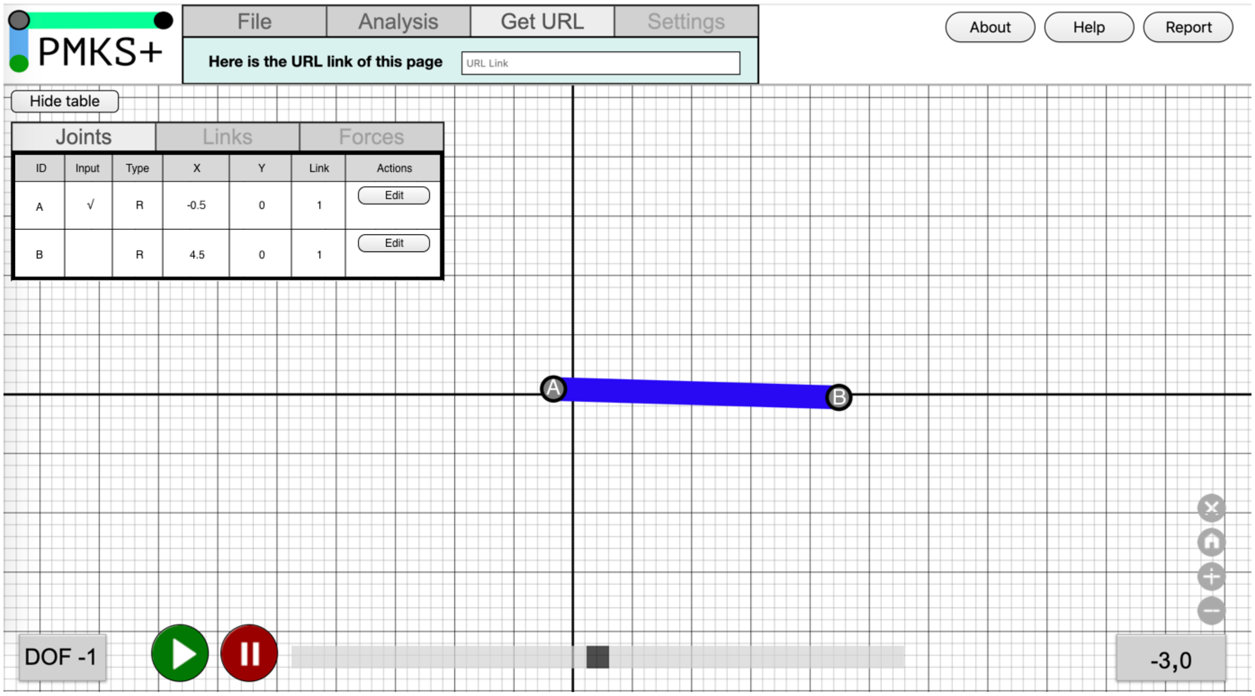
DOF -1

4,2

Figure 7.2: Get URL design 2

In the second design we removed the “Get URL” tab, renamed it to “Copy URL” and put it under the Data tab. When clicked, a tag shows up at the bottom of the button briefly, indicating the user that the URL has been copied. This way users don’t have to see the long URL string. In addition, the tabs are more consistent and “Copy URL” fits quite well under this category.

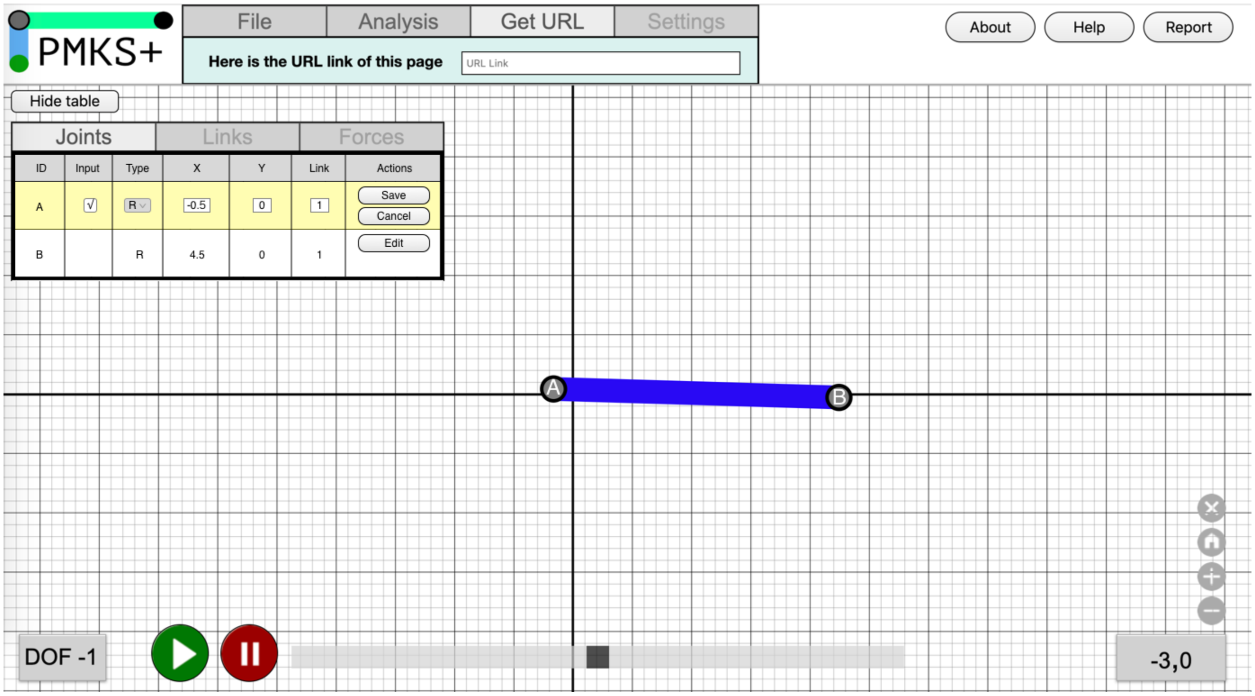
7.2 Tables



ID	Input	Type	X	Y	Link	Actions
A	√	R	-0.5	0	1	Edit
B		R	4.5	0	1	Edit

Figure 7.3: Normal mode of the joints table

Since there is no direct input or display of values on the link when users create or relocate links, the table is important to provide and show precise coordinates of joints, links and forces. Figure 7.3 shows the redesigned table to provide ID, input, ground, type, x and y positions and the link of a joint. There is an Action header, which should allow users to edit or delete an element.

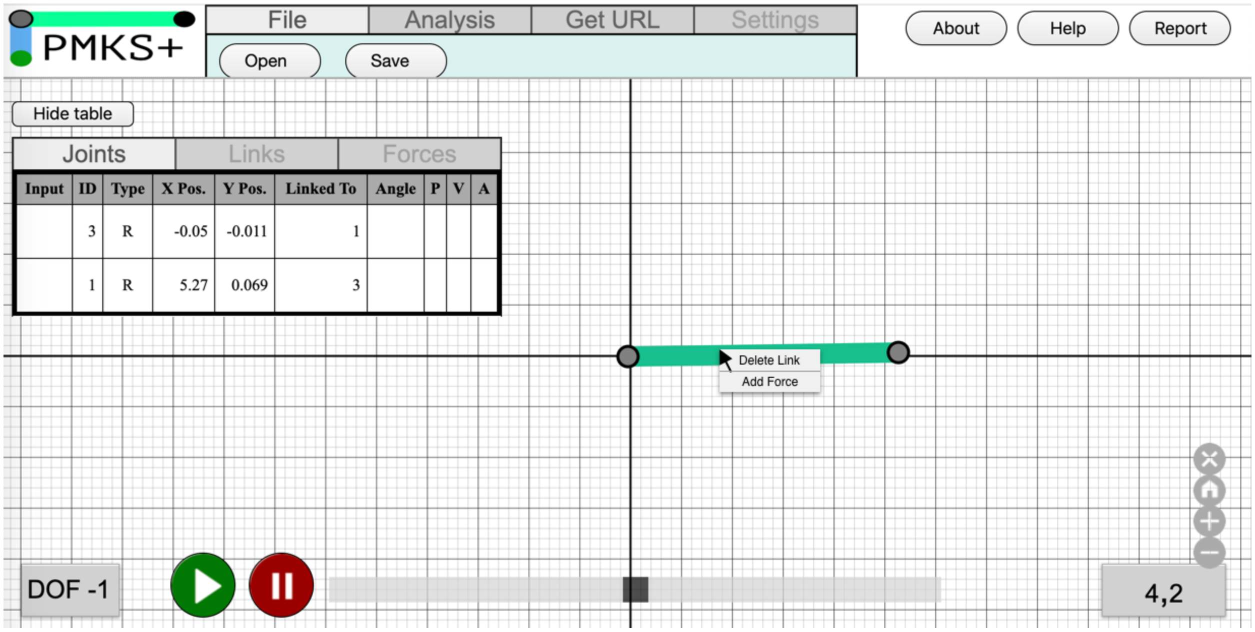


ID	Input	Type	X	Y	Link	Actions
A	√	R	-0.5	0	1	Save Cancel
B		R	4.5	0	1	Edit

Figure 7.4: Edit mode of the joints table

When the user clicks on the Edit button for an element, the table row will enter edit mode, see Figure 7.4. The row will be highlighted, and each editable field will be changed into either a checkbox, drop down select, or text box. In this case, the joint ID is not editable, so it will stay as a label, Input is a boolean so it becomes a checkbox, Type should only be one of the few types, so it is a drop down selector, XY coordinates are numbers therefore text boxes. The Link ID should be a string, which is shown in a text box. In edit mode, buttons under the Actions column are replaced with Save and Cancel buttons. Clicking on either of them will go back to normal mode. Clicking the Edit button on another row will cancel the edit of the current editing row and start editing of the new highlighted selected row.

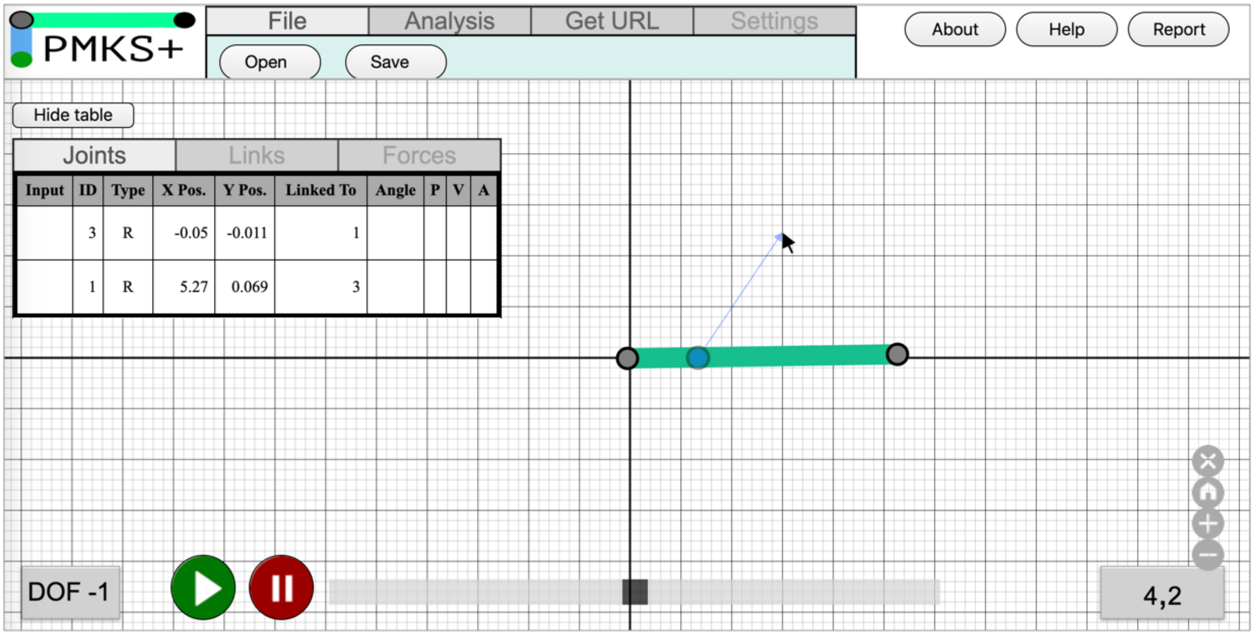
7.3 Adding Force



Input	ID	Type	X Pos.	Y Pos.	Linked To	Angle	P	V	A
	3	R	-0.05	-0.011	1				
	1	R	5.27	0.069	3				

Figure 7.5: Context menu item "Add Force"

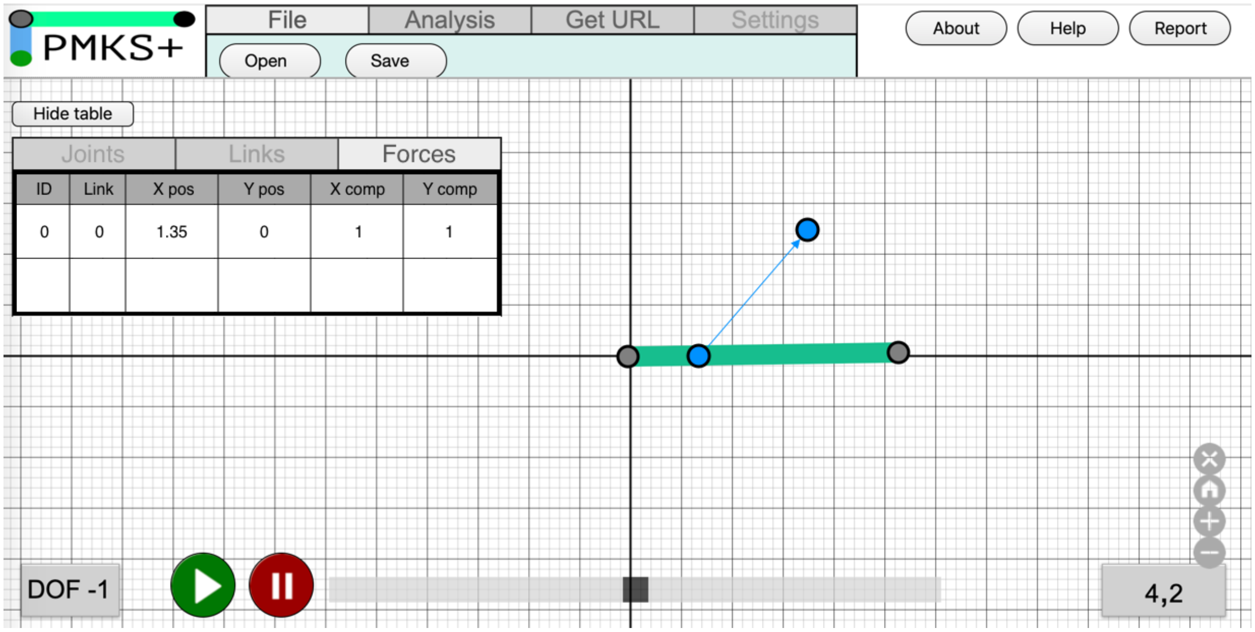
For the Add Force feature, we designed it to be similar to how a link is created. This way users will feel consistent: to create any vector, right click on something, bring up the context menu and create something using the click-click method. In this case, first right click on a link to bring up the context menu, click Add Force in the context menu.



Input	ID	Type	X Pos.	Y Pos.	Linked To	Angle	P	V	A
	3	R	-0.05	-0.011	1				
	1	R	5.27	0.069	3				

Figure 7.6: Selecting force endpoint

The position right clicked on the link will be the starting point of the force. Users can now move the cursor around to pick an ending point of the force. Then they can click again to confirm the endpoint.



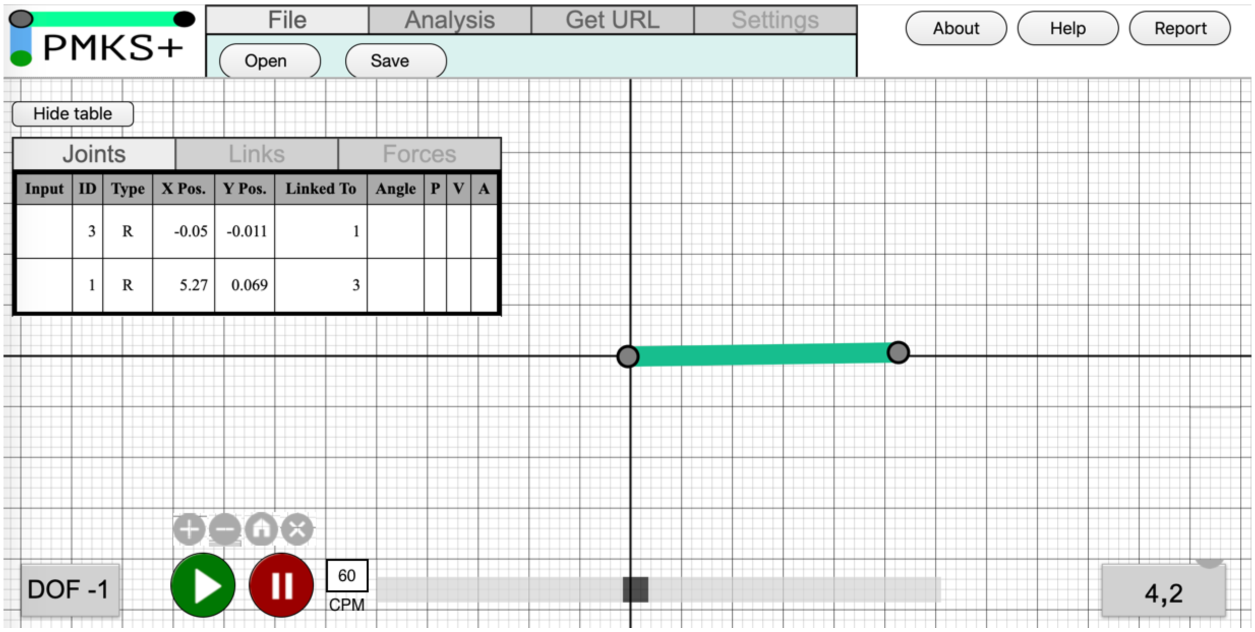
ID	Link	X pos	Y pos	X comp	Y comp
0	0	1.35	0	1	1

Figure 7.7: Created force

After a force is created, each point can be dragged around to change it. The points are blue to distinguish them from joints. However, the starting point should be restricted to stay in the link it belongs to. We decided to put the starting point back to its initial position after it is dragged outside of its link. An alternative design includes forcing the starting point to at least stay on the edge of its link event if the mouse cursor is outside as the user is dragging.

7.4 Direction and Speed

Design 1:



Input	ID	Type	X Pos.	Y Pos.	Linked To	Angle	P	V	A
	3	R	-0.05	-0.011	1				
	1	R	5.27	0.069	3				

Figure 7.8: Direction and speed design 1

Initially we designed this feature to be similar to PMKS Silverlight, in which users have to enter a number representing rotations per minute in the box labeled CPM, see figure 7.8. If they enter a positive value, the animation will play clockwise, if they enter a negative value, it will play counterclockwise. This design has several issues. First, this has an increased chance of user error as users need to type in things. They might enter a value that is not a number, or a number that is out of the range that we support. We need to check for these errors, and if they happen, we have to inform the users that they made a mistake. The interaction can get complicated very quickly. It is also not immediately obvious that they can enter a negative value, so they would need to check help.

Design 2:

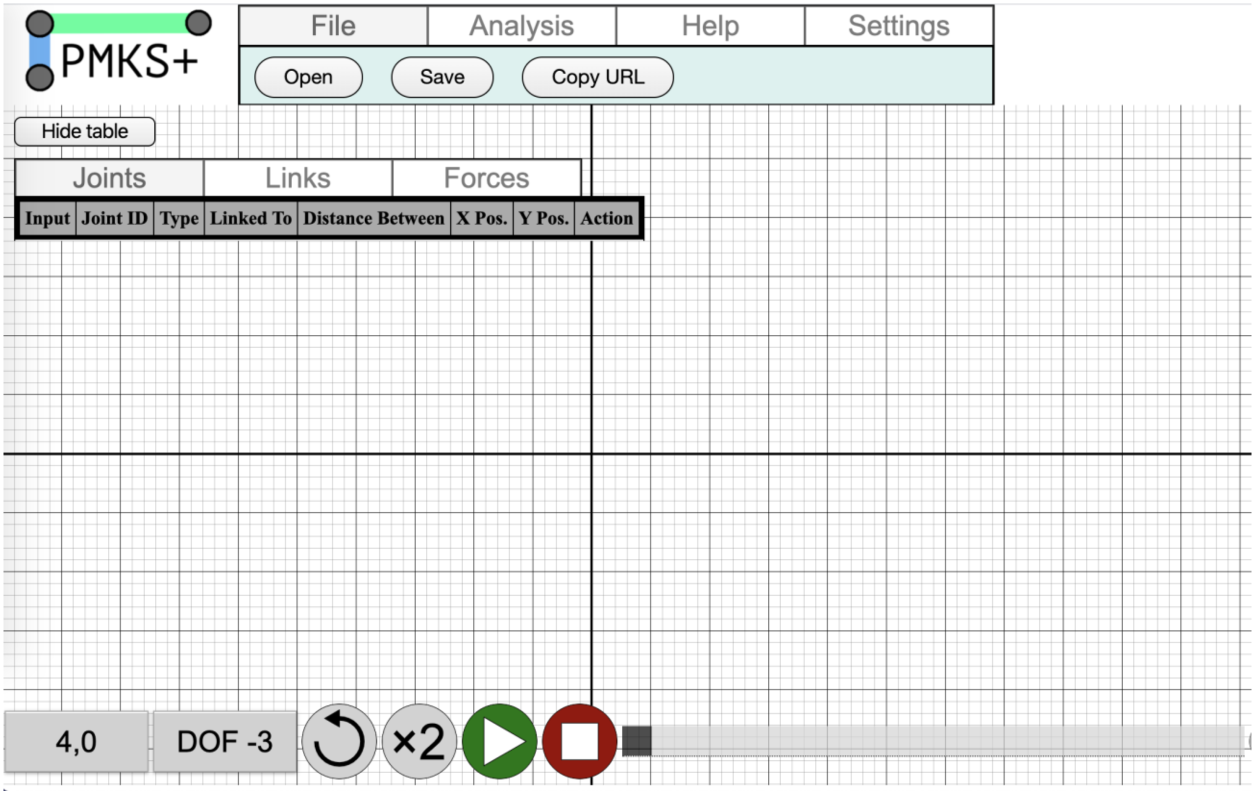


Figure 7.9: Direction and speed design 2

Figure 7.9 shows our second design. For this design, we simplified the interaction, so that RPM values are not visible and cannot be directly edited. Instead we have a button that toggles between clockwise and counterclockwise, and a speed button which toggles between x1, x2, and x3, corresponding to slow, medium and fast speed. This is a way better solution, as we don't need the exact RPM numbers just for the animation. RPM value is only required for static, dynamic and stress analysis, and we can have a separate settings page for users to edit the values for the analysis. By letting users toggle between the speeds, there will be no user error. In this case, it is also better than a dropdown select menu, since toggle buttons require very little motor skill. Users only need to click on one place for all the options, and we only have 2 options for direction and 3 options for speed. Users will need 1 click at most to get the desired direction and 2 clicks at most to get the desired speed.

7.5 Animation Controls

Design 1:

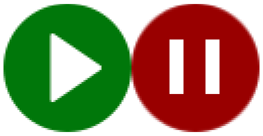


Figure 7.10: The play and pause buttons in the application

The first design changed the stop button that was originally there to be a pause button. This was done because the original stop button only paused the animation, so it did not make any sense. This made the symbols more in line with what the actions of what the buttons did.

Design 2:

The second design was a complete overhaul of the play/pause system. The stop button was added again but with new functionality. The stop button now resets the linkage instead of just pausing it. The play button now turns into the pause button when the animation is playing and the pause button changes to play when the user pauses the animation. The user now has the option to play, pause or stop/reset the linkage. This was done to give the user more control, and it allows the application to work like most video software with the play/pause functionality.

7.6 Grid Controls

Design 1:

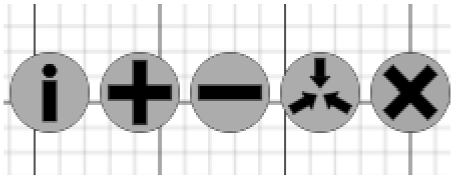


Figure 7.11 The Grid controls used in PMKS+W

There was one major design change to the design of the grid control and a smaller change. The major change was that the buttons became black images instead of white which allows the user to see these buttons easier. The second change was the change of the home button to three arrows to show that the button recenters the canvas. These changes allow the user to have a better understanding about the application.

7.7 Link Creation and Edit

One of the major new features added in PMKS+ W is the ability to create links as shapes. PMKS+ has bar links and polygon links, but the restriction is that joints must stay on the corners of links, they cannot be anywhere inside any link. The new design needed to give users the ability to change where the joints are. Several methods were considered for both creating and editing.

7.7.1 Creating a link

Through the design iteration process, the team came up with 3 alternative designs for users to create a link:

A. The same as PMKS+

1. User right clicks on the grid;

2. A context menu pops up;
3. User left clicks “create link” on the context menu;
4. A dot appears as a temporary starting joint of the link at where the user right clicked;
5. User can freely move the cursor to determine where they want the ending joint of the link to be;
6. When they decided where to put the ending joint, they left-click again on the grid to confirm the ending joint;
7. A default bar link will be created using the starting and ending joint;
8. User can change the shape later using shape editing methods below;

B. Pick shape when creating

1. User right clicks on the grid;
2. A context menu pops up;
3. User left clicks “create link” on the context menu;
4. A dot appears as a temporary starting joint of the link at where the user right clicked;
5. User can freely move the cursor to determine where they want the ending joint of the link to be;
6. When they decided where to put the ending joint, they left-click again on the grid to confirm the ending joint;
7. A default bar link will be created using the starting and ending joint;
8. The shape selection and editing screen shows up, where the user can resize, reposition and change the shape of the default link;
9. User click Save button under the shape selection screen (more in section 6.7.2: Editing a link);

C. Create the shape without any joints

1. User right clicks on the grid;
2. A context menu pops up;
3. User left clicks “Create Link” on the context menu;
4. A default bar link will be created based on where the user right clicked;
5. The shape selection and editing screen shows up, where the user can resize, reposition and change the shape of the default link;
6. User click save button under the shape selection screen (more in section 6.7.2: Editing a link);
7. User right click on the link to bring up the context menu;
8. User click on “Add Joint” on the context menu to add joints on the link;

We decided to use the first design. We learned from our advisors that shapes are primarily for the realistic visuals, or for uncommon situations that require multiple joints on the same link. Most of the links created will be simple bar links. The first design is faster to use than the second design, if the majority of the links created are simple bar links, as the second design requires 1 more step even if the user just wants a bar link. The first design is also much faster than the third design, as not only must the user go through shape selection, they must also manually add joints using context menus, which will be at least two additional actions (since a link will have at least 2 joints).

7.7.2 Editing a Link

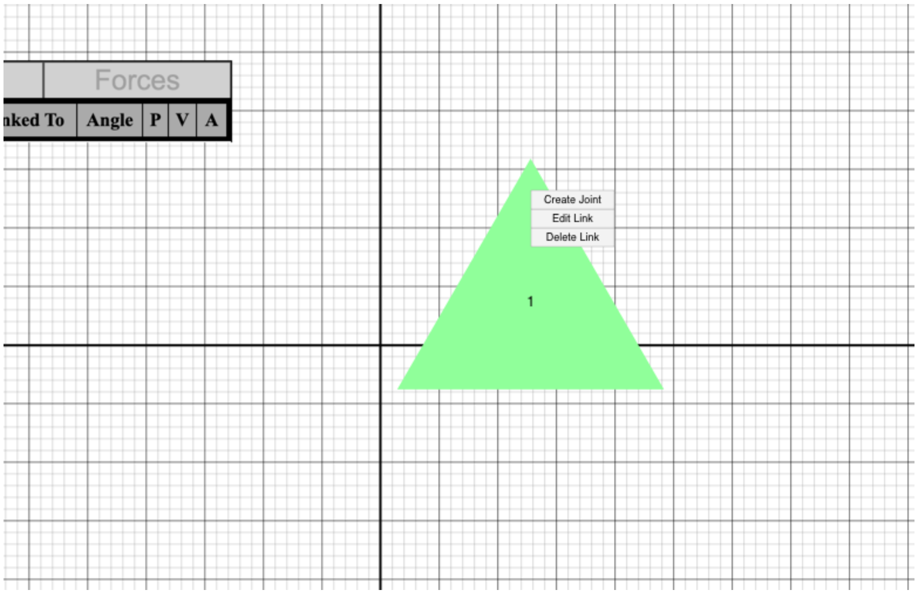


Figure 7.11: Context menu item "Edit Shape"

In order to edit a link, the user can right click on the link, and click the "Edit Link" button to enter edit mode, see figure 7.11. This follows the principle of "every action on a grid element is in the context menu".

Resize, Rotate and Move:

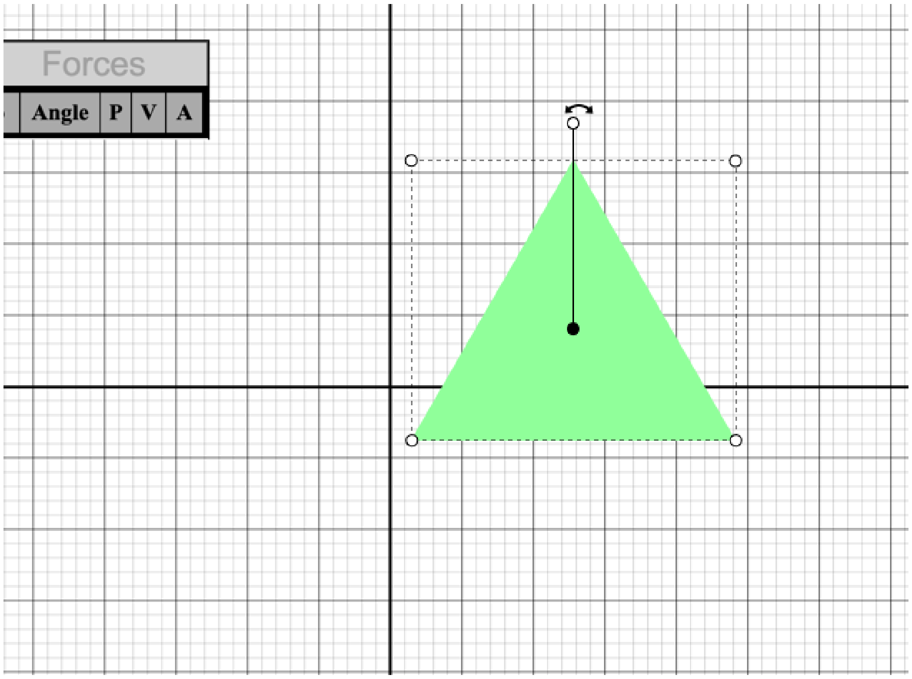


Figure 7.12: Bound edit dots and rotation edit dot

In edit mode, the user can resize the link by dragging bound edit dots (little white dots at the corner of a rectangle around the shape), see figure 7.12. Dragging the edit dot away from the center will cause the shape to scale up, dragging the edit dot towards the center will cause the shape to scale down. The scale ratio of the shape cannot be changed. The reason to use a rectangular bound instead of an edit dot for every corner of the shape, is that for complicated shapes like C shape or T shape, it will get too complicated to edit. Also the user can make any polygons with the 8 corners of a C shape, which may no longer look like a C. The user can rotate the link by dragging the rotation edit dot (white dot extended from a line connected to the center of the shape). The shape will rotate around its center, the same number of degrees as the angle between the rotation edit dot and the center. The line will rotate with the shape. The line is an indicator of the angle of rotation.

However, this design was later changed to not include the rotation edit dot. The link will both rotate and resize when a bound edit dot is dragged. The edit dot moves with the cursor, and the rest of the shape scales proportionally and rotates with the edit dot. This is simpler to implement, and fewer actions are needed to rotate and resize at the same time. The drawback is that the edits have less precision.

In order to move the shape, users can click and drag the entire shape, and the shape will move with the mouse cursor.

Select a New Shape:

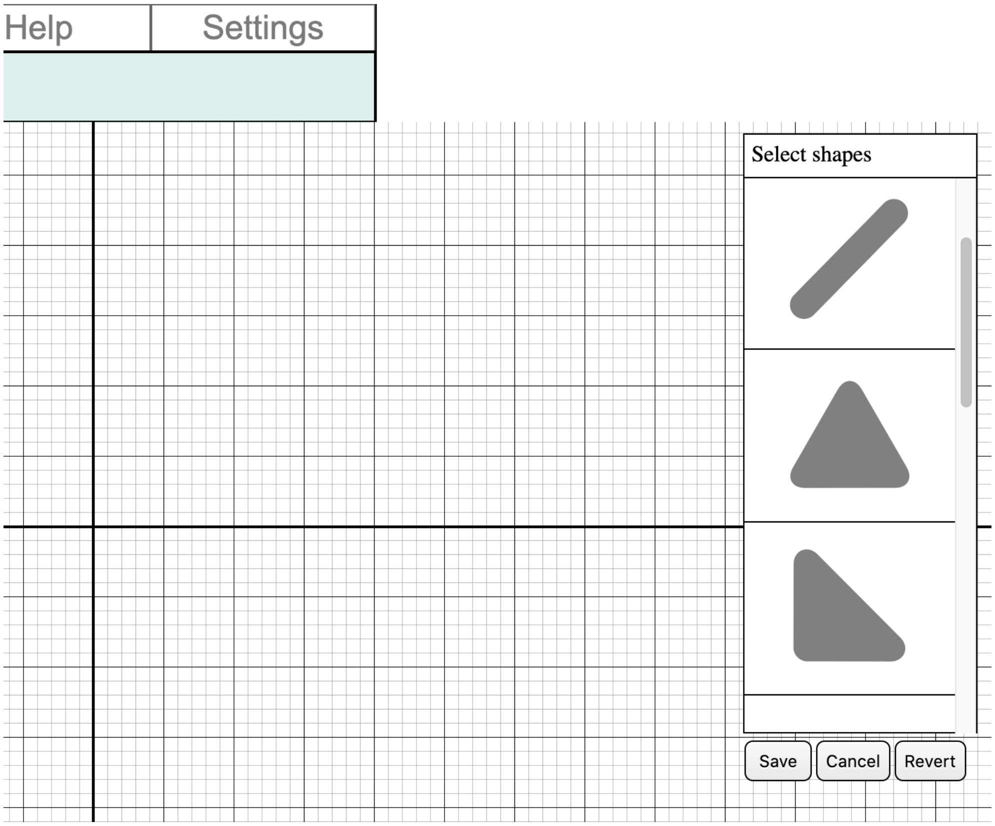


Figure 7.13: Shape selector

In edit mode, a side menu appears on the right side of the interface, see figure 7.13. Under the “Select Shapes” header is a list of shapes that can be selected. Clicking any of the shape icons will change the shape of the currently editing link to that shape. When a shape is changed, it will be placed at a default position between the link's first and second joint. Different shapes will have different default positions.

There are three buttons below the shape selector. Clicking "Save" will save the edited link and exit edit mode, clicking "Cancel" will exit edit mode and revert it to its original position, clicking "Revert" will revert it to its original position but stay in edit mode.

Elements Affected When Editing a Link:

Editing a link is not an independent action. Rotating, moving and resizing a link can cause a linkage to be invalid, as it might leave its joints outside the link, free-floating in the air. The simple solution would be to only allow edits in edit mode, and check the validity when a linkage is saved. However, this conflicts with a feature which already exists in PMKS+, which allows the user to drag a joint even when not in edit mode.

We decided to keep that functionality, since users will be able to conveniently drag a joint and watch how its movement (animation) path changes without entering edit mode. It is also not very intuitive to right click and edit a link in order to move a joint. Therefore the interactions between connected joints and links when a link shape is changed needed to be carefully designed.

We came up with 3 alternative designs:

A. Joints move with their links
    1. When resizing or rotating a link, the joints on the link reposition themselves proportionally to the resize and rotation to stay on the link
    2. When moving a link, the joints on the link move together with the link to stay on the link
    3. When moving a joint, the link stays at its original position, the joint cannot be moved outside its link, or the edit will be canceled or reset

B. Links move with their joints

    1. When moving a joint, the link scales proportionally as the distance between the joint and other joints on the same link increases or decreases
    2. When resizing, rotating or moving a link, its joints stay at their original positions, the link must have all its joints inside its shape, or the edit will be canceled or reset

C. Links and joints are independent

    1. When moving a joint, the link it is on does not change
    2. When resizing or rotating a link, its joints does not change
    3. Animation cannot play unless all joints are inside their links

Ultimately we decided to use a slightly modified version of design C.

Final Design: Links and joints are independent but restricted

1. When moving a joint, the link it is on does not change, but the joint cannot leave the link it is on, or the edit will be canceled or reset
2. When resizing, rotating or moving a link, its joints do not change. The link does not require all its joints to stay on top of it when editing. It does when saving.

The reason behind choosing this design was a process of elimination. The first design looks the simplest to use. However, given that moving the joints will not move shapes, while the joints must move while the shape is moved, consider a case with multiple links interconnected with each other by a joint. What if one of the links is rotated? The interconnected joint must rotate together with that shape, but it cannot move the other links it is connected to, and it also has to stay inside them. This means in order not to invalidate the linkage (keep the joint inside all the links it is on), the degree of angle for every link to rotate is extremely small, thus edits will be difficult.

The second design is not chosen also due to cases that would make edits difficult. Given that moving the joint will cause its link to resize, and resizing the link will not move its joints, consider a case with an equilateral triangle shaped link and two joints on it, each of them at a corner of the triangle. What if one of the joints is dragged towards the triangle’s corner which does not have a joint? The link must scale smaller, but it will be small enough that the other joint will be outside of the shape, which makes the linkage invalid. Dragging the other joints will cause the same problem, so there will be no way to make the shape smaller.

The third design offers the most freedom to edit anything. However, it is not very intuitive, and more prone to user errors, as it allows complete detachment between joints and links (it allows invalid linkages). The modification of the third design is to prevent any possibility for an invalid linkage to exist, by not allowing joints to be dragged outside of their links. This means the user can always do however they want in edit mode, only need to worry about linkage validity while saving. The drawback is that moving joints in non-edit mode is somewhat restricted.

8. Feature Implementation

This section provides a detailed explanation of how the more difficult to understand features are implemented. This includes which tools were used, the process by which each feature works, and the underlying calculations.

8.1 Link shapes

8.1.1 Scalable Vector Graphics

The link shapes in PMKS+ are created using Scalable Vector Graphics (SVG) in JavaScript, more specifically using the HTML path element. The path SVG element is the generic element to define a shape. All the basic shapes can be created with a path element. (MDN Web Docs, 2020). A path can be drawn using a series of actions, which are defined inside the path string. The path string is created using a list of points marking the corners of a shape: linking them together creates a shape. Bezier curves can be added inside the path string to make rounded corners. A Bezier curve is a parametric curve used in computer graphics and related fields (Wikipedia, 2020c). It takes a start point, end point and two control points to draw a curve.

8.1.2 Process of Making a Link Shape

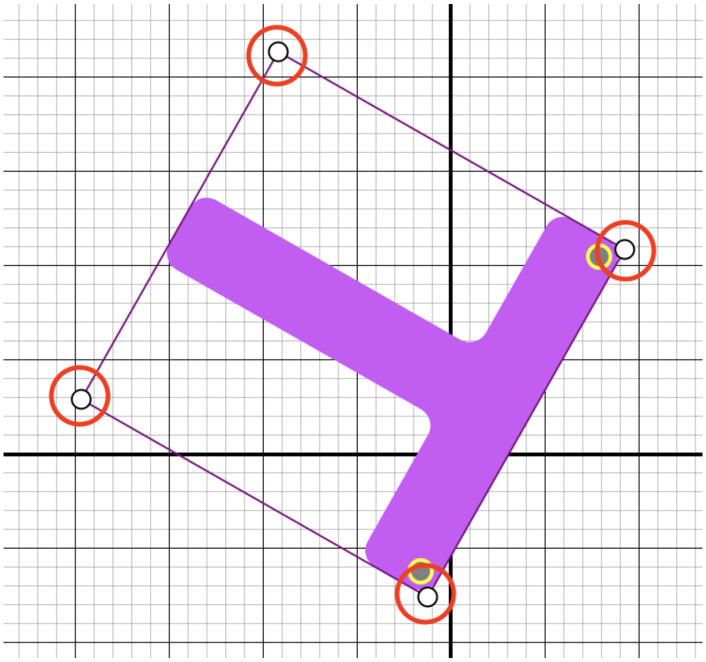


Figure 8.1: Bounds for a T shape link

In order to create a shape, first its Bounds have to be created, see figure 8.1. Bounds is a simple object with 4 points and connecting them always forms a rectangle. Default Bounds are created using the position of the first and second joint of a link. Default Bounds will always be big enough to include the first and second joint on the link. A list of points is then generated using the default Bounds, and finally the shape is drawn using that list of points. Different shapes will have different default Bounds and different lists of points.

8.1.3 Drawing a Link Shape

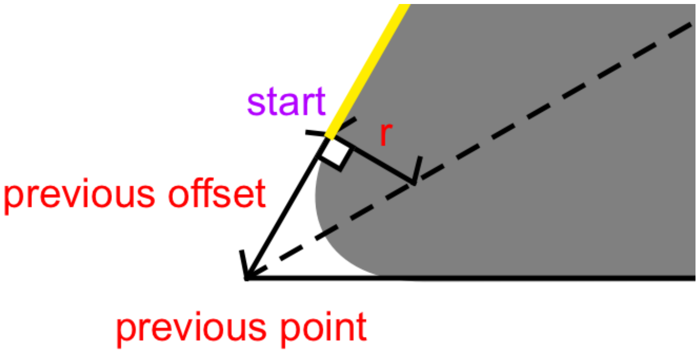


Figure 8.2: calculate the starting point

A shape is drawn by drawing part of the path for each corner and putting all the parts together. For each corner, we first start looking for the previous point. The previous point is defined as the point before the current point in the point list. If the current point is already the first point in the list, the previous point will be the last point in the list. Assuming the path for the previous point is already drawn, we need to know where the previous path ends. This can be found by offsetting the position of the previous point, see figure 8.2. The offset distance is calculated by:

$D_o = | r / \textit{tan}(\text{lastAngle} / 2) |$

where r is the rounded corner radius, lastAngle is of the angle of the last corner.

Using the angle of the last point relative to the current point and the offset distance, we can get the X and Y component of the offset. By adding the offset components to the position of the previous point we can get the starting point.

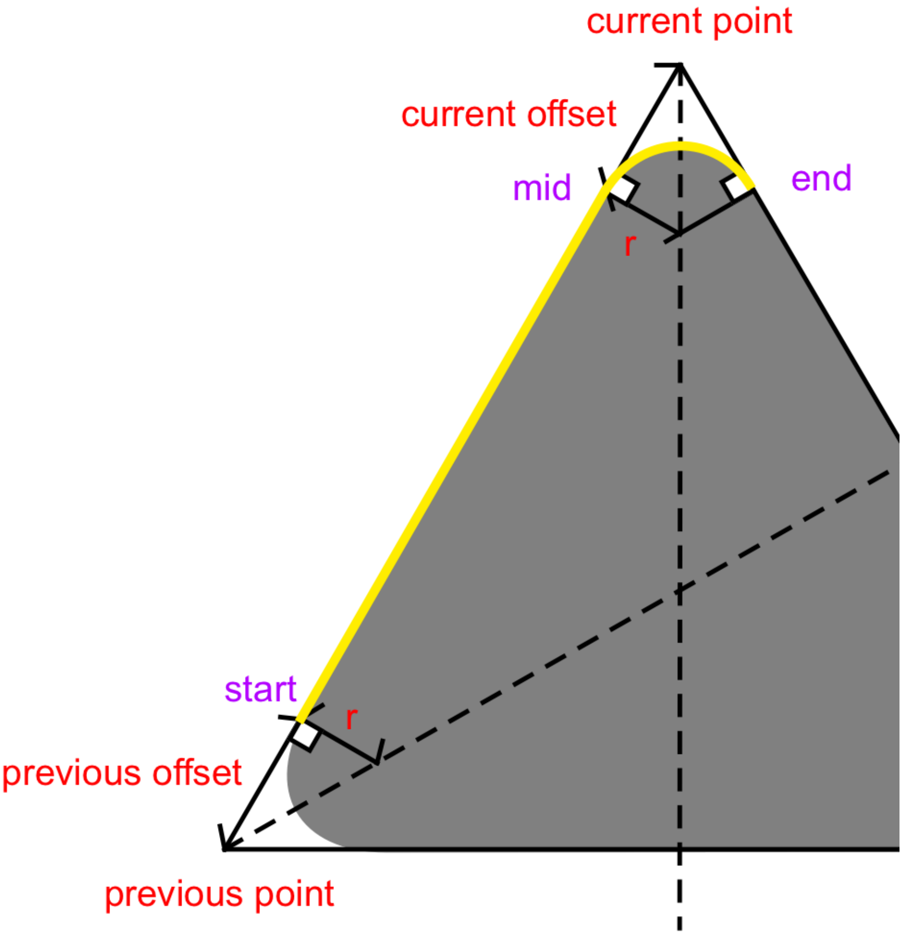


Figure 8.3: calculate the path for one corner

Doing the same for the current point, we can get the offset distance of the current point, and using the angle of the current point relative to the last point, we can get the point where the curve is just about to start (curve start point). Using the angle of the next point relative to the current point, we can get the point where the curve ends (end point).

The control point for the bezier curve is simply pointing towards the current point, each control point has a distance of 0.551915 times the rounded corner radius from the curve start point and end point. This is the value to create a perfect curve when the corner is 90 degrees. It will not be perfectly rounded on corners that are not 90 degrees, but it is good enough. It can be improved in future development.

By connecting the starting point, curve start point, bezier curve and finally end point, we can draw a line from where the previous curve ends all the way to where the next line starts, see figure 8.3. By doing this for each corner and connecting them together, we can draw a rounded corner shape given any list of points.

8.1.4 Editing a Link Shape

The Bounds are also used to allow the user to edit a shape. When a user drags a corner of the Bounds, the center position of the Bounds is saved, and the amount of rotation caused by the drag is calculated. Because every corner has the same distance from the center, the position of each new corner will be:

$P[x, y] = [cx + \mathit{sin}(oAngle + dRotation) * dist, cy + \mathit{cos}(oAngle + dRotation) * dist]$

which cx and cy are the x and y coordinate of the center, oAngle is the original angle of this corner relative to the center, dRotation is the degree of rotation caused by the drag, dist is the current distance from the dragged corner to the center.

8.1.5 Animating a Link Shape

The position of link shapes during animation is not calculated by the simulator. Instead, something called Relative Coordinates is used. Relative Coordinates is a simple object with 4 points, each of them represents the position of each corner of a Bounds relative to the first joint of the link. Note that this position sees the first joint and second joint as if they are on a horizontal surface. Relative Coordinates are saved whenever an edit to the linkage is made. When animating, the Bounds of links are updated using saved Relative Coordinates and positions of joints, and the shape is drawn using the Bounds.

8.2 Force Simulation

Calculation of how the force moves with the linkage when the animation is playing is done inside the simulator. However, it is very similar to how link shapes are animated. Like links, each force also has a Relative Coordinates, but instead of Bounds, it is for the start and end points of this force. With the position of the first and second joint at a specific time, and the Relative Coordinates, the position of the start and end point of a force at that time can be calculated. Note that only the end point of a local force needs Relative Coordinates, the end point of a global force only needs its X and Y component.

8.3 Copy URL

The Uniform Resource Locator (URL) created when sharing a linkage has all its data encoded in query strings. It follows the format of

protocol//hostname:port/pathname?j= ... &l= … &f= ...

All joint data follows the "j=", all link data follows the "l=", all force data follows the "f=". Property values of an element are separated by ",". The "RelatedJointID" property for links has its related joint IDs separated by "|". The data string for each element is separated by a newline character. Some values are shortened. Numbers are rounded to the nearest 2 decimal places, booleans are shortened to "t" or "f", the shapes of links are shortened using the function *shapeFullnameToNickname*(). An example URL is shown below.

http://example.com:4200/? j= A,0,0,R,t,t,0 \n B,3,3,R,f,f,0 \n &l= 1,A|B,l,3,0,0,0,0,3,3,3 \n &f= F1,1,1,1,1,-1,t \n

8.4 Save File

The file created when the “Save” button is clicked is a simple Comma Separated Value (CSV) file. It can be directly opened by spreadsheet software like Microsoft Excel, Google Sheets or Numbers by Apple, and is human readable and editable. It follows the format of

```
joints\n
id,x,y,type,ground,input,angle\n
…
links\n
id,joints,shape,b1x,b1y,b2x,b2y,b3x,b3y,b4x,b4y\n
…
forces\n
id,link,startx,starty,endx,endy,fixed\n
…
```

Property values of an element are separated by “,”. The “joints” property for links has its joint IDs separated by “,”, but the property value as a whole is wrapped by a set of quotation marks. Data string of each element is separated by a newline character. No values are shortened. An example file is shown below.

```
joints
id,x,y,type,ground,input,angle
A,0,0,R,true,true,0
B,3,3,R,false,false,0
links
id,joints,shape,b1x,b1y,b2x,b2y,b3x,b3y,b4x,b4y
```

```
1,"A,B",line,3,0,0,0,0,3,3,3
forces
id,link,startx,starty,endx,endy,fixed
F1,1,1,1,1,-1,true
F2,1,2,2,2,0,false
```

8.5 Direction and Speed Control

Changing the direction and speed of the animation simply creates an alternative Time Sorted List (TSL) for the animation. TSL contains all the positions of joints and forces in a range of time, so the animator can iterate through it and rearrange the linkage at each timestamp, thus playing the animation. Changing the direction simply reverses the order of the TSL. Changing to slow speed is done by duplicating each item, meaning that each position will last two frames instead of one, therefore playing “slower”. Changing to fast speed is done by removing every 1 in 2 items in TSL, meaning the animation will skip every other position, therefore playing “faster”. The modified time sorted list is stored in GridComponent as speedChangedList.

9. Application Testing

This section describes the detailed testing of the system that was conducted by Robert Dutile.

In addition to the user testing, the program was carefully tried in a more controlled fashion by a Computer Science student. We started by getting the program loaded to a website so that it can be accessed by users without launching a local host. Since the tester was a member of the team, they were already very familiar with how to use the program. Then the tester collaborated with Professor Pradeep Radhakrishnan to put together a clear list of what exactly needed to be tested.

The testing process was performed in 2 stages. First, all controls and UI elements were tested individually in order to verify their basic functioning. Then, several test activities were performed in the program, such as re-creating several diagrams which we considered representative of common use-cases for the program. These examples were created through discussion with one of our advisors, Professor Pradeep Radhakrishnan, who has the best understanding of the expected use cases. After this, the recreated diagrams were manipulated in various ways to see if anything broke or behaved unexpectedly. We recorded the results of all of these tests, and any bugs or errors found in this process were checked to see if they could be fixed in a reasonable time frame. After fixes were performed, the testing was repeated, with a particular focus on verifying whether reported bugs had been fixed, and ensuring that no new errors had been introduced.

To ensure that performance was maintained across multiple platforms, we performed full runs of testing separately for 4 distinct browsers (Edge, Safari, Chrome and Firefox). This allowed us to catch any compatibility issues or platform-specific bugs that appeared, of which there were several. For examples of use we attempted to create regular 4-bar, 6-bar and slider crank linkages, with a force acting upon one of the links of the 6-bar, as shown in Figures 9.1-9.3.

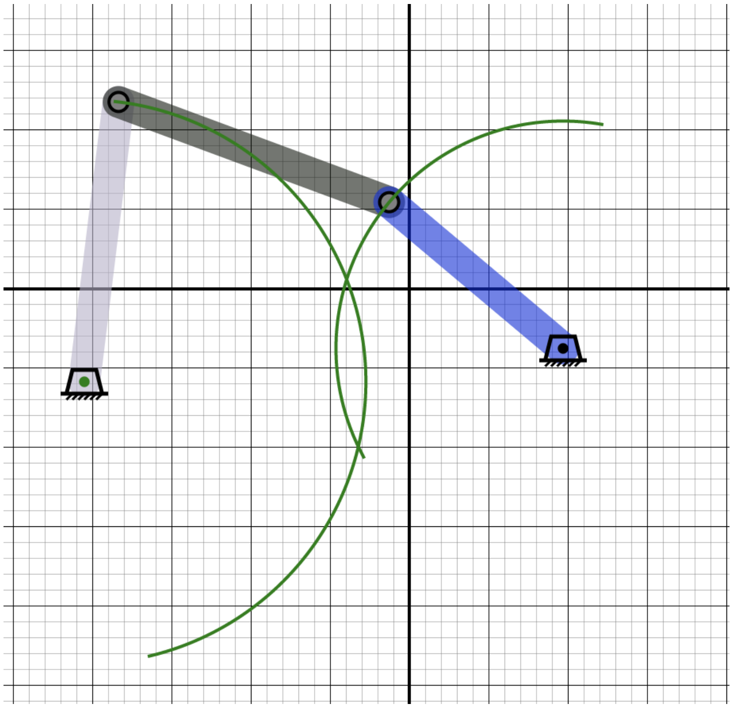


Fig. 9.1: 4 bar revolute linkage

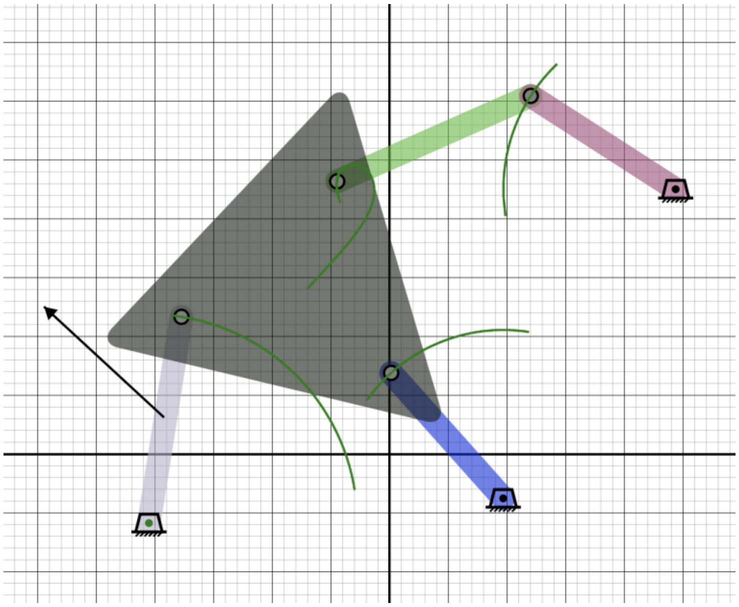


Fig. 9.2: 6 bar linkage with a force

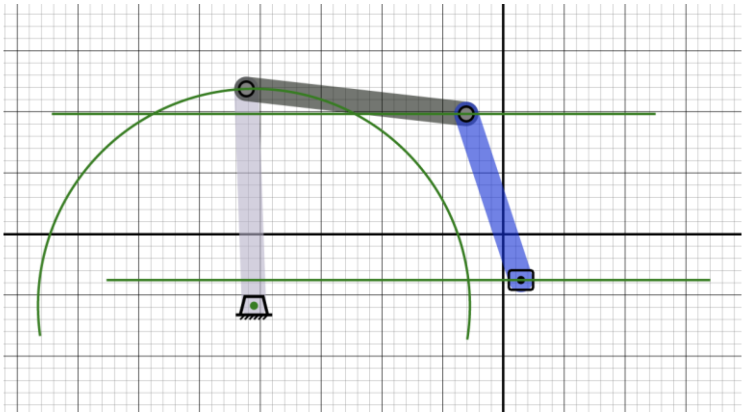


Fig. 9.3: 4 bar slider crank linkage

Over the course of testing we did find several bugs, some of which our team was able to do away with, such as sliding joints being saved as ground joints. Others proved intractable in the time we had remaining, such as the repeating issues that occur when setting joints to ground in the middle of a linkage, or the Safari specific problem where copying a url causes the new page to open with a black background on the canvas, which is apparently a legacy issue that has existed since the last team created the program. The exact results of the testing shall be broken down by the browser the tests were run on.

9.1 Chrome

The first browser to run through each round of tests was Google Chrome, which was tested in a Mac environment.

Example Linkages:

Tests performed by recreating and manipulating sample linkages.

4-bar linkage (all revolute)	● Creation ○ Works as Expected ● Animation ○ Works as expected ● Save ○ Immediately creates and downloads a CSV file as expected ● Reopen it ○ Colors changed, but all else functions and updates appropriately. ● Change the input/middle/output shapes ○ Shapes change without problem, animation remains functional. ● Reopen in separate browser ○ Opening in Safari works same as opening in chrome ● Test movements ○ Works as expected ● Check copy url ○ Colors of links change, but seems otherwise identical

6-bar linkage with force (all revolute)	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Works as expected ● Reopen it ○ Colors change, does not preserve animation speed settings, otherwise works fine ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Safari is no different to chrome in this respect ● Test movements ○ Works as expected ● Check copy url ○ Same as reopen

4-bar Slider Crank Linkage	● Creation ○ Works as expected, but: ■ **Problem**: the DOF display doesn't seem to adjust as it should. Stopped me from animating on one joint, but a different one worked fine. Also, gave no indication or control of what the angle of the track would be ○ No track, square doesn't tilt to match angle. ○ **Update**, mostly the same, the default surface angle seems to have become horizontal however. ● Animation ○ Works as expected ● Save ○ Works as expected ● Reopen it ○ **Problem**: The slider comes out as just a ground joint. ○ **Update,** fixed, now works as expected. ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Safari experiences the same issue ○ **Update**, now works as expected ● Test movements

	○ Works as expected ● Check copy url ○ **Problem**: The slider comes out as a ground joint ■ **Update**, fixed, now works as expected.

Individual Feature Testing:

Tests of individual features in the user interface.

General Examination of UI	● Buttons in the top tab all seem Identical, swaps are easy to see and understand. ● The Settings tab has no buttons in it however. Should either have some added or have the tab removed. ○ **Update**, the settings tab now contains RPM Input. ● Joints table seems to work and appear properly, however, Links and Forces do not seem to populate with information. ○ **Update**, links table has been removed, but forces table remains and now appears to function properly. ● All other aspects of UI appear as expected.

Bottom Panel	● Clear Button ○ Works as expected ● DOF Display ○ Seems to react appropriately, but: ■ **Problem**: still displays -3 on empty canvas ○ **Update**, now an empty canvas defaults to 0, which is correct. ● Coordinate display ○ Works as expected ● Play/pause ○ Works as expected ● Stop ○ Works as expected ● Reverse ○ **Problem**: Failed to work properly on 4 bar linkage, had seemingly no effect regardless of state of animation (playing, paused, stopped, etc. ○ **Later Update:** problem has been fixed, now works as expected. ● Speed Change ○ Cycled appropriately, it had a clear effect on animation. ● Info ○ No effect

	● Zoom Buttons ○ Works as expected ● Center ○ Works as expected

Top Tab	● About ○ **Problem**: Still links to the original PMKS website, which is undesirable. ○ **Later Update**: now does nothing ○ **Another Update**: now links to pmks.mech.website, which is correct ● File tab Buttons ○ All works as expected ● Analysis Tab Buttons ○ All bring up the appropriate window, none of those are functional at this time ○ **Update**, now the window clearly states that this functionality will be added in a later update. The bottom button is non-functional, but while pressed shows this weird vertical stretching, up to the top of the window. ● Tutorial ○ Button works, but: ■ **Problem**: its contents are massively outdated. They only give basic instructions for creating a link, and obsolete instructions for creating a link of an odd shape. ○ **Update**: now links to a full page of instructions and how-to videos explaining basic functionality, much improved

	● Help ○ Sub sections open properly, but: ■ **Problem**: all of them only contain temporary placeholder text. ○ **Update**: subsection buttons now arranged immediately next to the relevant areas of the screen, and contain relevant contextual information. ■ **Problem**: Unfortunately, the information on the top bar in particular seems slightly out of date. Maybe a miscommunication? ● Report ○ **Problem**: Doesn’t do anything ○ **Update**, now links to appropriate google form ● Linkages ○ Shows window properly, but ■ **Problem**: linkages contained therein don’t work/aren’t clickable ○ **Update**, now marked as “Templates”, no images now but the buttons work as expected, opening up a new tab with the example linkage ready and. Fully functional. ■ Minor note, joints table begins unpopulated, but shuffling between tables causes it to start working properly again.

	● Settings ○ This tab is empty ○ **Update**: Now contains rpm and direction input controls

Adaptation to Different Resolutions	Works as Expected.

Tables	● Joint Table: ○ When the number of Joints increases beyond the reasonable bounds of the table, it becomes scrollable with a clear scroll bar on the side. Responds perfectly well to a correctly positioned touchpad or mouse scroll ○ **Problem**: Input column does not display anything ○ Joint IDs seem to iterate appropriately ABCD etc. ■ Once exceeding the length of the alphabet, it loops around to AA AB AC properly too ○ LinkedTo works fine ○ **Problem**: Distance Between only displays one number even when the link is connected to multiple joints. ○ X and Y positions update and display appropriately ○ **Problem**: Edit and Delete buttons on the table are non-functional at this time. ○ **Problem**: If you tab to another table and back, the joints table comes out empty. Fixes itself when you move, create or delete a joint. ● Link Table: ○ Table is empty ○ **Update**, table now missing entirely ● Forces Table: ○ Table is empty

	○ **Update**, table is now working as expected in general ■ LinkId attribute seems to work as expected. ■ **Problem**: Angle seems to initialize properly, but no later alterations or edits made seem capable of updating it. ■ **Problem**: X and Y magnitude appear to be exclusively filled with default placeholders, and there does not seem to be any way to alter them at this time. ■ X and Y position seem to work and update as expected. ■ **Problem**: Edit and delete buttons have no effect at this time.

Input Toggling	● **Problem**: Works properly on 4 bar linkages for the most part, but does not acknowledge the input itself being toggled unless you toggle ground instead, and even then, if you toggle it back, it has a black instead of green dot but still animates as an input. After doing this, it also doesn’t seem to properly register changing the positions of joints. No stretching, bars just snap back into place on play, and remain in their original positions after hitting stop instead of jumping back to the new position. Alterations to the shape of the link persist, but they still all snap to the old animation positions, even if that causes the joint to no longer be inside of the link. They just follow the old animation anyway. ○ Clearing does seem to fix this, fortunately ○ **Update**, changing random joint to input in normal slider crank refuses to animate as it should, but the DOF still displays 1 for some reason. ○ Also, swapping the ground and input for the template 4-bar linkage causes the animation to experience a weird skip partway through. ■ Moving one of the joints fixes it.

9.2 Safari

The second browser for each round of tests was Safari, which we tested in a Mac environment.

Example Linkages:

Tests performed by recreating and manipulating sample linkages.

4-bar linkage (all revolute)	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Works as expected ● Reopen it ○ Colors of links change, otherwise works entirely as expected ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Same in chrome as in safari ● Test movements ○ Works as expected ● Check copy url ○ **Problem**: When I load a url copied from this in a separate tab, the background of the canvas is black. Same url works fine if loaded on chrome. Result seems otherwise identical to copy url on chrome ■ Same issue occurs if loaded from a url created in chrome.

6-bar linkage with force (all revolute)	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Works as expected ● Reopen it ○ Same as in Chrome ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Same in chrome as in Safari ● Test movements ○ Works as expected ● Check copy url ○ Same as the 4-bar

4-bar Slider Crank Linkage	● Creation ○ Same as Chrome ● Animation ○ Works as expected ● Save ○ Works as expected ● Reopen it ○ Same as Chrome ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Chrome produces the same result ● Test movements ○ Works as expected ● Check copy url ○ **Problem**: Combines slider issue that showed up in the chrome version with the safari specific black background ○ **Update**, see chrome, though black background error remains.

Individual Features Testing:

Tests of individual features in the user interface.

General Examination of UI	All looks the same as Chrome version

Bottom Panel	● Clear Button ○ Works as expected ● DOF Display ○ Seems to react appropriately, but: ■ **Problem**: still displays -3 on empty canvas ○ **Update**, now same as chrome ● Coordinate display ○ Works as expected ● Play/pause ○ Works as expected ● Stop ○ Works as expected ● Reverse ○ Works as expected ● Speed Change ○ Works as expected ● Info ○ Does nothing ● Zoom Buttons ○ Works as expected ○ Touchpad Scrolling also works ● Center ○ Works as expected

Top Tab	● All buttons and tabs same as chrome
Adaptation to Different Resolutions	Same as chrome
Tables	Same as chrome
Input Toggling	**Problem**: Could not toggle input off directly, had to either move it to another joint or toggle ground **Problem**: Toggled a middle joint as input, DOF display didn't seem to update appropriately but it did correctly stop me from animating. **Update,** menu item now says set as input, other problems same as before, also, toggling a middle joint to ground causes the input to just keep going, stretching the link in the process.

9.3 Firefox

The third browser in sequence was Mozilla Firefox, which was tested in a Windows environment.

Example Linkages:

Tests performed by recreating and manipulating sample linkages.

4-bar linkage (all revolute)	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Works as expected ● Reopen it ○ Same as chrome ● Change the input/middle/output shapes ○ **Problem**: Works mostly as expected, but it seems to be possible for the boundary box of a shape to get hidden behind other shapes when editing. ○ **Update**, this has been partially fixed. The opacity of links has been reduced, allowing the boundary box to still be seen when beneath another shape, but the corners hidden behind another shape remain impossible to interact with. ● Reopen in separate browser ○ Performs the same in Edge as in Firefox ● Test movements ○ Works as expected ● Check copy url ○ Colors of links change, otherwise works as expected.

6-bar linkage with force (all revolute)	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Works as expected ● Reopen it ○ Same as chrome ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Same in edge as in Firefox ● Test movements ○ Works as expected ● Check copy url ○ Same as chrome

4-bar Slider Crank Linkage	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Same as chrome ● Reopen it ○ Same as chrome ● Change the input/middle/output shapes ○ **Problem**: If you hit stop while editing a shape, it turns into a thin line that’s almost imperceptible. If you then hit save, it remains that way, and animates too. ■ This does not always happen, sometimes you get a different shape entirely. These seem to snap to position too. ○ **Update**: this issue has been fixed. ● Reopen in separate browser ○ Same in Edge as in Firefox ● Test movements ○ **Problem**: Ran into 2 issues. ■ After setting a second joint connected directly to the input as a slider, the DOF still said 1. Luckily it still refused to animate

	■ **Update**, similar to other browsers, setting a middle joint in a 4-bar slider crank to ground and hitting play causes the input to still revolve, and either ignore the new ground entirely if it is attached directly to the input or cause the link attached to the new ground to stretch weirdly if not. Even moving all of the joints around doesn’t fix it. ■ I hit toggle ground on the middle slider from previous, but it turned into a normal joint. Animated fine, so I thought this would not be an issue.. Then, I toggled it to ground again, but while the joint changed, the DOF hadn’t, and when I checked it animated as if the joint was free ● **Update,** this issue is mostly fixed, the sliding joint still turns into a normal joint when set to ground, but it does so properly now, and if you set it to ground again, it turns into a regular ground joint. ● Check copy url ○ Same as chrome

Individual Features Testing:

Tests of individual features in the user interface.

General Examination of UI	Largely the same as chrome, save that the edit and delete buttons look slightly different.

Bottom Panel	● Clear Button ○ Works as expected ● DOF Display ○ Same as chrome ● Coordinate display ○ Works as expected ● Play/pause ○ Works as expected ● Stop ○ Works as expected ● Reverse ○ Works as expected ● Speed Change ○ Works as expected ● Info ○ Does nothing ● Zoom Buttons ○ Works as expected ● Center ○ Works as expected
Top Tab	● All Same as chrome

Adaptation to Different Resolutions	Same as chrome
Tables	All same as chrome
Input Toggling	**Problem**: toggled input to the third ground joint of six bar, started revolving freely as if unattached, link was bobbing in line with joint as if to follow, think only failed to stretch because it wasn’t a bar.

9.4 Edge

The final browser tested in each round was Microsoft Edge, which was tested in a Windows environment.

Example Linkages:

Tests performed by recreating and manipulating sample linkages.

4-bar linkage (all revolute)	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ **Problem**: Returns error saying file could not be downloaded ○ **Update**: now works as expected ● Reopen it ○ Files from other browsers work fine, but cannot save file ○ **Update**, now works as expected ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Same in Firefox as in Edge ● Test movements ○ **Problem**: Toggled a joint as ground, input joint started rotating freely, stretching the link ● Check copy url ○ Works as expected

6-bar linkage with force (all revolute)	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Same as 4-bar ● Reopen it ○ Works as expected ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Same in Firefox as in Edge ● Test movements ○ Works as expected ● Check copy url ○ Works as expected

4-bar Slider Crank Linkage	● Creation ○ Works as expected ● Animation ○ Works as expected ● Save ○ Same as 4-bar ● Reopen it ○ Works as expected ● Change the input/middle/output shapes ○ Works as expected ● Reopen in separate browser ○ Same in Firefox as in Edge ● Test movements ○ Same issues as previous ● Check copy url ○ Works as expected

Individual Feature Testing:

Tests of individual features in the user interface.

General Examination of UI	Same as all previous tests

Bottom Panel	● Clear Button ○ Works as expected ● DOF Display ○ Same as previous browsers ● Coordinate display ○ Works as expected ● Play/pause ○ Works as expected ● Stop ○ Works as expected ● Reverse ○ Works as expected ● Speed Change ○ Works as expected ● Info ○ Does nothing ● Zoom Buttons ○ Works as expected ● Center ○ Works as expected
Top Tab	● All same as other browsers

Adaptation to Different Resolutions	**Problem**: At smaller resolutions, the DOF display scrunches weirdly, and the scroll bar does not appear until well after the rightmost buttons become inaccessible
Tables	Same as other browsers
Input Toggling	Same as other browsers

10. User Evaluation

To determine the system quality, the team conducted an evaluation that consisted of a user completing a set of tasks, then answering a survey to obtain feedback about the application. Each user was given two sets of tasks, one to test the functionality of PMKS+W, and the other to test Working Model, a similar modeling software. After the user completed a task, the user would answer the questions related to that task. Then the user would go onto the next set of tasks. The users were divided into two groups, one where the user completed the working model set of tasks first, while the other had the PMKS+W task completed first. This helped to eliminate any bias that may have been caused by the order in which the tasks were completed. The users were Mechanical Engineering students from Worcester Polytechnic Institute and Oregon State.

10.1 Comparison to Working Model

One of the reasons that we wanted to compare PMKS+W to Working Model was because we wanted to see how PMKS+W compared to other available systems that are out there. For this to be successful, we needed to gauge the experience that the test subjects had with the software which can be seen in Figure 10.1.

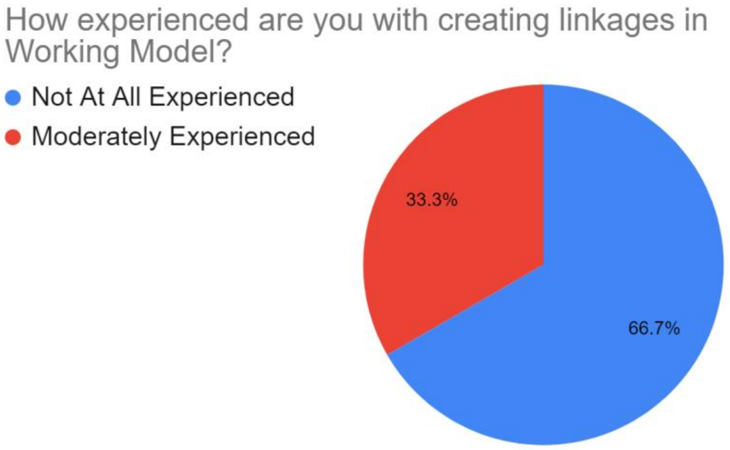


Figure 10.1 User Experience with Working Model

The lack of user experience with the Working Model application was surprising to the team; we thought that the users would have more experience with the other application. The lack of experience meant that most users had never used Working Model before or only had used it a few times. This probably helped as the users would not have preconceived notions about either of the applications as they had little to no experience with the applications. Also, the users that had experience with Working Model had the option to skip the Working Model creation portion of the evaluation.

The overall satisfaction with the application, which can be seen in the two Figure 10.2 graphs, was the most important to the team as that would determine whether or not the user would keep coming back to the application. These numbers were obtained by asking the users what their overall satisfaction of the application was on a scale of 1 to 5, where 1 was very unsatisfied and 5 was very satisfied.

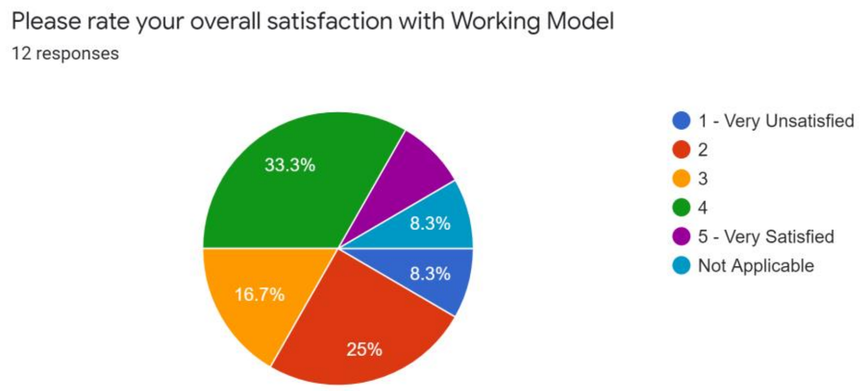


Figure 10.2a Overall satisfaction with Working Model application on a scale 1-5 (Working Model First)

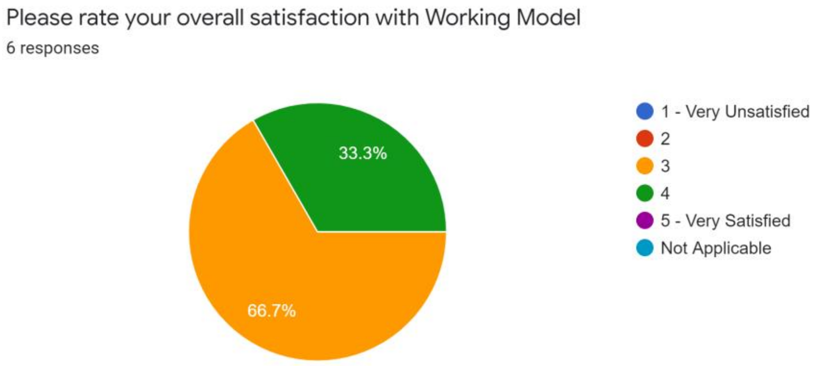


Figure 10.2b Overall satisfaction with Working Model application on a scale 1-5 (PMKS+W First)

The users expressed an average rating of 3, which was calculated from the numbers in 10.2a and 10.2b, for Working Model. This was slightly lower than the value that was expressed for PMKS+W, 3.27, which was calculated from the numbers in Figures 10.4 a and b and are also shown in the Appendix A. These results are not statistically significant at a p=0.05 level due to the low number of tests. While we cannot use these numbers to say that people prefer the PMKS+W application and are more satisfied with the service it provides. This does show that the application works well enough to be compared to services that are paid for by schools.

Another way to compare the two applications was obtained in the free response section of the survey. These responses can be found in their entirety in Appendix A of this paper. The most responses cited the clarity and ease of use for PMKS+W. Users stated "It's much clear[er] to do on PMKS[+W]" and that "PMKS+[W] has a much simpler UI which is great for simple designs. Working Model has a much larger breadth of features and can be used in a wider array of applications." This matches one of the goals that we originally set out to accomplish. Since PMKS+ was created as an introduction to linkage design, it is very reassuring to hear that it is a simpler version of the existing applications. It was very important for the team that the application remained easy for first time users to use so accomplishing that goal is good to hear.

People that favored Working Model over PMKS+W cited the responsiveness and the ability to not crash as reason that Working Model is better:

> “PMKS+[W] looks a lot better and is much easier to use but Working model does not crash all the time when you use it.”

This complaint came up a lot as the major reason for frustration with the overall design. The ability to run without crashing will need to be addressed in order for the application to be considered successful. These issues will be further addressed in section 2.

10.2 Feedback on Overall Design

The largest problem that people seemed to have with PMKS+ was the responsiveness and speed of the application.

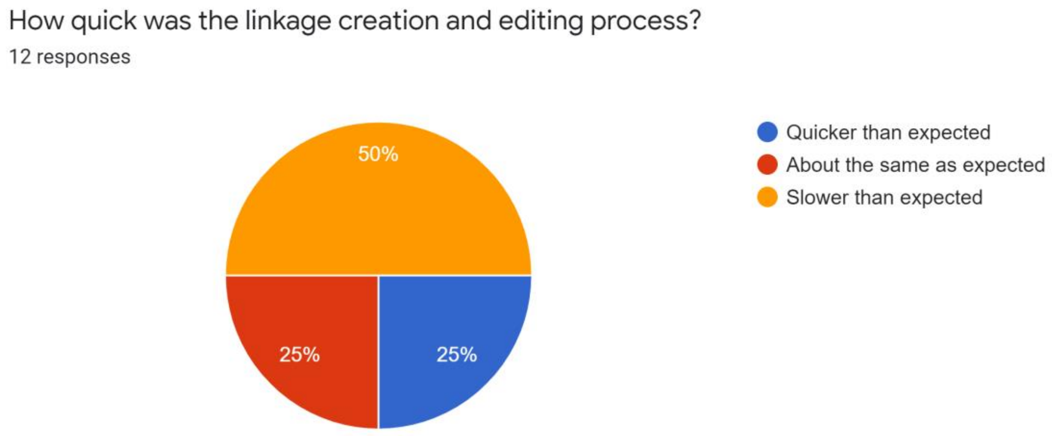


Figure 10.3 The speed at which the user completed the PMKS+W tasks

This lack of speed, seen in Figure 10.3, could be attributed to the fact that many users complained that the application kept on crashing. The team is not sure exactly why these crashings were happening, but we initially theorized that the hosting for the application was not robust enough to handle the testing of the users. Upon further analysis, an issue with the position of joints in the code was found and needed to be corrected. This was most likely the reason for the crashing of the application.

The second thing that was cited as a major issue was with the table not working as expected. This happened because there was a bug in the code. This was a large issue and users cited that “if table edit could be fixed” it would greatly improve the functionality of the application and make it significantly better. These errors can be reflected in the scores received shown below in Figures 10.4 a and b, as well as in the free response answers seen in the Appendix A. Figures 10.4 a and b show the users thoughts on the overall design of the application where 1 is very unsatisfied and 5 is very satisfied. In these free response questions, the users cited the crashing as the errors as major detractors, but had positive reactions to the design of the application.

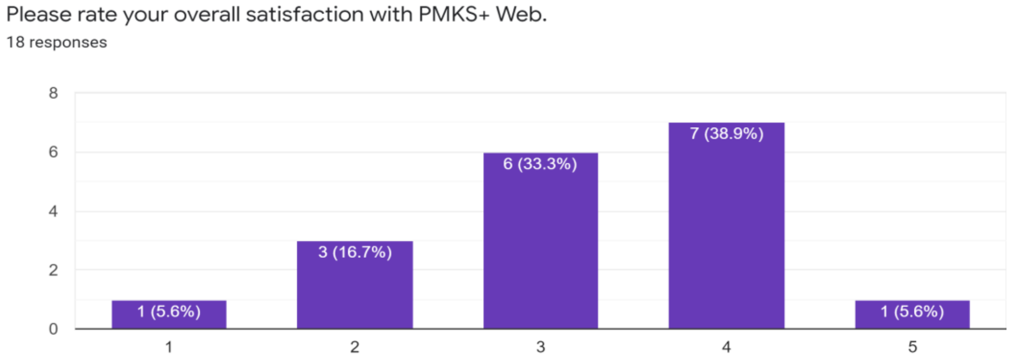


Figure 10.4a Overall Satisfaction of the PMKS+W app on a scale 1-5 (Working Model First)

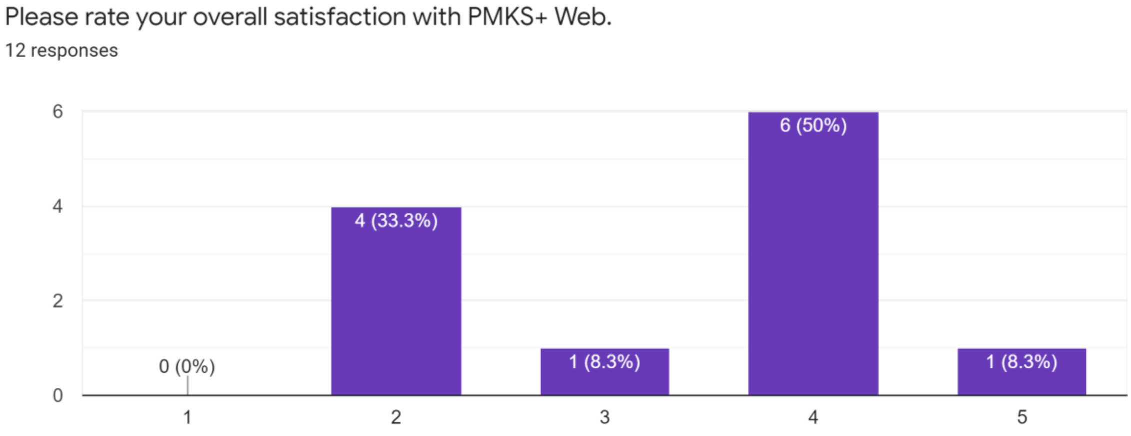


Figure 10.4b Overall Satisfaction of the PMKS+W app on a scale 1-5 (PMKS+W First)

Another thing that the team wanted feedback on was the top features that the user liked about PMKS+W. This was done by asking the user to list the top three features that they liked when using PMKS+W. The complete list can be found in Appendix A. The top three features were linkage connection, shape editing, and tracing of animation. These three were all features that were changed by the team. This helps to support our team's goal of improving upon the features that were implemented in the original PMKS+.

10.3 User Evaluation Overview

Overall, the user evaluation experience gave a lot of good feedback for the team. It helped identify some bugs with the application as well as giving the team a good understanding of what the user base found beneficial for the application. With this information the team can identify what would need to be done next to lead to the launch of this application. It also lets the team know what worked well and what did not work as well in the application.

Issues with the process were that more students participated in the PMKS+ first then Working Model than the other way around. Eighteen students did PMKS+ first, while only twelve worked on Working Model first. Also, some users only completed the PMKS+ linkage design.

Apart from these few issues, the team felt that the user evaluations gave the team a better understanding of the application as well as helping the team find a few errors in the code. The charts helped the team get a general understanding of how the user experience was for PMKS+W and be able to compare it to Working Model. Figures 10.3 and 10.4 showed that PMKS+W had a small advantage over Working Model in overall satisfaction as the users scored PMKS+W a 3.27 compared to Working Model's 3.0. This data is good for the team to get a better understanding of people's reaction to our application, but since it is not statistically relevant, more user evaluations would have to be conducted to say that PMKS+ is better than Working Model.

While the charts were good for a general picture, the free responses gave the team a much clearer picture of what the user's thought of PMKS+. It helped us realize an error in the system that had previously gone unnoticed. They also showed us the positives of the application like how intuitive the application was to use. Overall the feedback was very helpful to the team, and can be used to help make the application even better.

11. Conclusion

This section will discuss the future work that could be done with the PMKS+ system. It also looks at the project's goals and whether or not the team accomplished those goals.

11.1 Future Work

The team had a lot of ideas that they wanted to implement, but there was not enough time to complete these tasks. These ideas could be continued by another MQP and completed at a later date. There would need to be more thought put into the hosting of the application, as what was used for the evaluation was not robust enough for a full scale launch. The issues of crashing would have to be figured out before the application could be used in courses. Other things that could be considered before a full launch are listed below.

One project task was a homework submission system integrated into the application. The team liked this idea but understood the scope of what would have to be done to complete this task. The team decided that it would not have enough time to complete this, as it would be a massive undertaking. A future MQP team could make this the focus of a project of their own. This feature would be very helpful for students and faculty, so we think it would be worth pursuing in the future.

Another feature that should be implemented is the editing of the force table. Currently forces are only editable by clicking and dragging. Having the forces directly editable from the table would be a great benefit as it allows precision.

The application will need to be developed more for the practical use of the application. An additional MQP team could work with the advisors to continue to improve the application.

Our team understands that while we made significant progress in the development of the application, there are still things that must be finished to have a final application.

11.2 Team Goals

To measure the success of the project, the team looked back to the goals that the team initially had. From the start, the team set out to accomplish these five goals:

1. Implement all the features that exist in PMKS but not in PMKS+ ;
2. Add new features that the client, Professor Pradeep Radhakrishnan, requested and those developed ourselves;
3. Improve user experience on PMKS features, and make existing users of PMKS feel comfortable using PMKS+W;
4. Fix existing interface issues with PMKS+;
5. Add new interfaces (for new features) without breaking the existing design.

While not all the goals were completely accomplished, most of the goals that the team set out to achieve were completed.

The team set out to implement all the features in the original PMKS version which was goal 1. The team identified the major features as *table edit* and the *url save and upload* as the major features that needed to be transitioned. The url feature was successfully implemented, and the table edit was partially implemented. This goal may also not have been the best goal to have. The old system was not the most intuitive to use, so it was unclear whether or not that a feature should be brought over to the new system.

The team also wanted to add the new features that the client proposed as well as some features that we developed. The team accomplished the major task of adding link shape editing, as it was the first feature to be added. Other things such as forces and sliding joints were also added. While some features were not implemented (as can be seen in the section Future Work above), the majority were completed. The team considers that it meets this goal.

The next three goals (3 to 5) are intertwined as they are about delivering a positive user interface and a good experience to the user base. This goal was accomplished because we see positive comments in the user provided feedback; i.e., their thoughts about the user interface and experience. We set out to create an interface that was easy to use and understand as this is meant to be an introductory software for linkage design. The user feedback supports this by citing the ease of use and the simplicity of the UI design being the major attractions. These goals can be considered a success due to the results of the user testing and survey.

11.3 Closing Thoughts

Through the development of the application, the team learned a lot about user interface design and programming in a group. The group realized how much thought and care goes into every decision. This was different than every other project at WPI as the team was responsible for coming up with the ideas that needed to be implemented. The challenge of programming over multiple terms, and with a team, also made organizing and making sure everyone was on task, challenging and important. As we finish this project we will be able to take the lessons we have learned and apply them in the real world.

12. References

Appikatla P, Cecil G, Taylor M, Tsiakmakis D. *PMKS+: Recreating a Legacy Application.* Retrieved from https://web.wpi.edu/Pubs/E-project/Available/E-project-042419-230306/unrestricted/PMKS+_Final.pdf (2019, October 1st).

Blair-Early, Adream, and Mike Zender. “User Interface Design Principles for Interaction Design.” *Design Issues*, vol. 24, no. 3, 2008, pp. 85–107. *JSTOR*, www.jstor.org/stable/25224185.

Chi, C. (2018, August 6). The Beginner's Guide to Usability Testing. Retrieved from https://blog.hubspot.com/marketing/usability-testing (2019,September 23).

Fracker, M. L. (2010). Scaling usability in terms of requirements: A method for evaluating user interfaces. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 54*(6), 576-580. doi:10.1177/154193121005400607

Fain, Yakov, and Moiseev, Anton. *Angular Development with TypeScript*. Second edition. Shelter Island, NY: Manning Publications, 2019. Print.

Krug, Steve. *Don’t Make Me Think! : a Common Sense Approach to Web Usability*. 2nd ed. Berkeley, California: New Riders Publishing, 2007. Print.

Roose, J. (2017, March 29). How to Conduct Usability Testing in Six Steps. Retrieved from https://www.toptal.com/designers/ux-consultants/how-to-conduct-usability-testing-in-6-steps (2019,September 23).

Spolsky, Avram Joel. *User Interface Design for Programmers*. 1st ed. 2001. Berkeley, CA: Apress, 2001. Web.

Tesarik, J, Dolezal, L, and Kollmann, C. "User Interface Design Practices in Simple Single Page Web Applications." *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*. IEEE, 2008. 223–228. Web.

Vogels, R. (2019, August 1). A 6-Step Guide to Web Application Testing - Usersnap. Retrieved from https://usersnap.com/blog/web-application-testing/ (2019,September 23).

Wikipedia (2020a). Angular (web framework). Retrieved from https://en.wikipedia.org/wiki/Angular_(web_framework) (2019, October 1st).

Wikipedia (2020b). TypeScript. Retrieved from https://en.wikipedia.org/wiki/TypeScript (2019, October 1st).

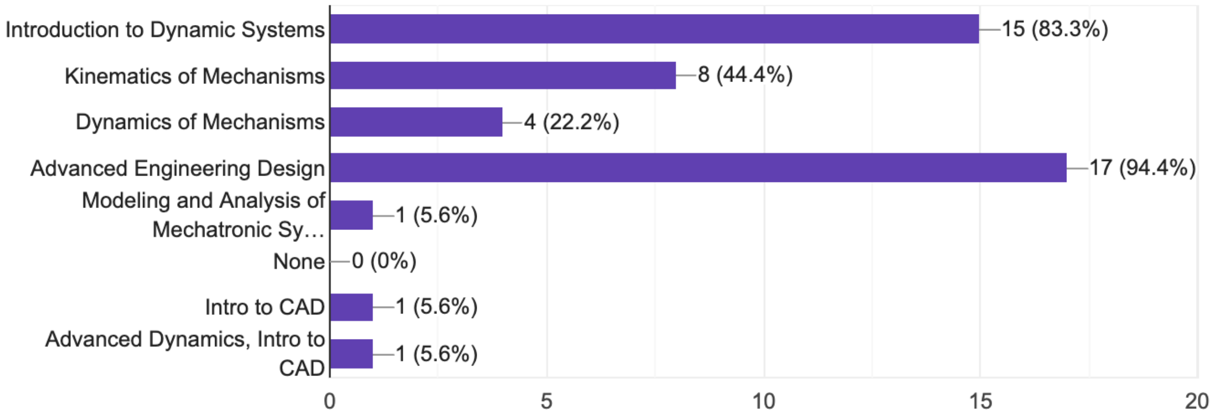
Wikipedia (2020c). Bézier curve. Retrieved from https://en.wikipedia.org/wiki/B%C3%A9zier_curve (2020, May 1st).

MDN Web Docs (2020). Paths. Retrieved from https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Paths (2020, May 1st).

Appendix A

This section displays all the results from the user evaluations of PMKS+W.

A.1 PMKS+ First Evaluation Response



Please select which relevant courses you have taken. 18 responses

What did you find difficult in Working Model? 6 responses

The tools were difficult to use because their icons are not easy to tell apart.

not much

Pretty much everything, had to look up a tutorial online and I still dont know if its possible to create regular links instead of just shapes or even at an angle.

The UI is a lot less intuitive than PMKS. It is annoying that parts collide with one another and it is hard to edit existing parts. It is usually easier to delete everything and start over.

Find how to do things

Getting things to line up properly. Deleting elements. sawing from scroll to select

What did you feel was easy in Working Model? 6 responses

Creating the lnks was easy.

Pretty much everything

Copying exact links to make symmetrical assemblies was easy

There are a lot of features which allow for a lot of options when it comes to design. Once you get the hang of it, it is easy to design complex linkages.

It has many options and tools

Adding a specific type of link

What do you like about Working Model? (This can include qualities that you know from usage outside of this evaluation.) 6 responses

I like the user friendliness of creating shapes.

I could use my keyboard to set link properties

Has a lot more functionality from what it looks like but with my limited exposure to the program I don't really know much of what it can do outside of this evaluation.

All the features such as vector velocity, acceleration, and force analysis are nice to have. I like the vast amount of freedom there is when desiging assembilies as well.

Tool bar option

Layout and toolbar

Please write any comments you have on how PMKS+ Web compares to Working Model. 6 responses

It's much clear to do on PMKS

Working model is easier to use than PMKS+ but PMKS+ is more intuitive.

PMKS+ looks a lot better and is much easier to use but Working model does not crash all the time when you use it.

PMKS is more streamlined and faster for simpler linkages. It also displays the path of all joints easily which is a huge plus. PMKS+ is far better for initial design work.

PMKS would be much better than working model if the table worked

PMKS+ has a much simpler UI which is great for simple designs. Working Model has a much larger breadth of features and can be used in a wider array of applications.

What did you find difficult in PMKS+ Web? 18 responses

Site became unresponsive a lot for no apparent reason, I was just making a link or setting a ground point and it would just become unresponsive. Instructions format was a little clunky - the videos and description separate was weird - I would have liked to see a unified narrated video or text description with photos. I still wasn't able to find how to set the lengths of links to a specified length, I just moved the joints until they were close to what I wanted.

It didn’t work without great WiFi service. It was hard to get the animation to start (or getting DOF to 1) I wish there was a table example that could tell you what each type of link does and how to get a DOF of 1.

It was very hard to move the joints as it would frequently crash.

It crashed almost every other second. It would seem that as you slide around the joints if you move one to a position that would work it just crashes the program.

Not that clear to set input.
Creating a 3 node link.

editing exact link lengths in the table.

The website had to reload a few times so I lost my progress.

Editing linkages was not possible through the tables.

The web client crashed every time i moved linkages to quick or to far

Editing and dragging links can feel clunky. The ability to add entire links with joints would be helpful. Also, the ability to edit the proportions of the shape links, not just the scale would help a lot. Editing specific links from the chart also doesn't work.

the animation process is still confusing.

The fact that I couldn't get the tracer to work, other than that it wasn't bad, ran pretty quickly.

The table said i could edit the vales and link lengths but I could not. It made it more difficult to work with in total and there is no undo button.

The initial set-up of the experiment was not an intuitive process.

Connecting existing links was slightly difficult. Dragging one existing link to connect it with another existing link at a single joint did not work, so I had to delete one and connect it directly.

I can't actually create the infinity shape because it requires exact dimensions, which I was unable to do on the web version. I was not able to directly edit the table values even after pressing edit. I had to click around to try to get exact values for the infinity shape, and even after many tries it did not work.

The page kept freezing and the tutorials assumed things about the linkage that were not stated (input set)

It crashed a couple times for unknown reasons.

What did you like about PMKS+ Web? 18 responses

Animating and exporting the linkage were both very intuitive. Once I learned how to make the links and set ground/input, it was easy to know how to do so when it would become unresponsive (which happened frequently). Saving and opening the linkage as i worked was helpful as it was necessary to make progress when the site became unresponsive.

It was better than downloading a software on desktop.

The integration of the software into the browser.

Its easy to quickly make stuff and adhjust things as long as it doesnt crash.

Different colors of links.

I had a much better experience with the desktop version.

It is very user friendly.

It run much better than the application. It was easy to use and required no installations.

it shows the tracing of the joints as you move them.

I like how new links are given a new color. Layout grid is a huge plus.

the edit and delete buttons in the table, easier to create and delete joints and links, the tutorials are much more accessible and explain more things, the videos are helpful.

That I can access it with my Macbook without having to use remote desktop and it is a pretty fast program.

its web based and didn't crash.

The quick shape manipulation.

The interface was easy to use and rather intuitive.

I can just load it up on a web browser.

The design is simple and easy to understand if you know what you're doing.

I like how it shows the trace pattern in real time when moving joints around.

What did you dislike about PMKS+ Web? 18 responses

The site kept becoming unresponsive. Not able to figure out how to set exact lengths for links or exact coordinates of joints.

It was hard to create links for 1 DOF.

It would crash a lot.

It would crash a lot. Sorry about beating the dead horse but there is nothing worse then having something crash and having to start again.

Its toutorial can be more direct and clear.

It crashed multiple times while using it. Many of the functions seemed different than the desktop version.

I dislike that the page crashes sometimes, as the website has some bugs.

CTRL+Z did not work to undo mistakes.

I had to restart the program a total of 25 times in 30 minutes since it would crash every time i was trying to move joints to try to resemble an infinity symbol.

Can feel a little clunky at first, but after I got a feel for it, everything made sense.

not much.

I felt the help videos last time were more useful.

It is very clunky and un intuitive to use. the table does not work and that made me really sad.

All of the tutorials lacked sound.

Needing to specifically click the save button to change the shape was a bit annoying. I would prefer to just be able to click out of the shape menu to save it.

I couldn't directly edit the table values.

There are too many assumptions about what the user already knows.

The shape manipulation was a bit strange. It was hard to rotate the shape without altering its size.

Please list the top 3 features you like in PMKS+ Web. 18 responses

Saving the linkage.
Exporting URL of linkage.
Ease of constructing links.
Ease of setting ground/inputs.

Intuitive to animate and export linkage.
Doesn’t fail as much as the desktop version.
Is easy to input links.

Very similar to PMKS Desktop in where everything is.

Creation of joints, adding ground, adding input.

Quick and easy to use, The ui is pretty ok. Shows you the joints locations and link lengths

Report bugs, tamplet, and table.

Easy feature creation.

The table of values (if it worked).

Copy URL, ease of creating links/ground/input, setup of menus.
Making the linkages and joints was easy to use.
Applying ground to limit DOF was simple.

The animation was very easy to use.

Joint Path simulation, inputing shapes as links, and easy of establishing new links and joints.

Path tracing, colors make it easy to see, all the useful information is easy to see.

help, tutorials, templates.

The table of the joints and forces, being able to easily change shape of the link, and the fact it is online.

Web based, quick animation, has help videos.

The joints and forces tables, The copy URL button, and the various simulation manipulation buttons on the bottom.

The path illustration feature was nice to visualize the motion without simulating it. The toggle bar at the top of the interface is very useful and allows easy access to a bunch of useful tools. The linkage simulation toolbar is also very useful. I've never used a linkage software before but most programs I have used for CAM have specific simulation modes and complex setup processes. The simplicity of this program is arguably the best part.

It is on the web, can delete links or joints easily, can copy paste the url to show others your linkage.

1. ease of controls, 2. tracer movement, 3. menu options.
Trace Points.
Play Animation.

Templates.

If you want, please explain your choices. 5 responses

I am fairly certain that I created the correct linkage shape but the simulation does not allow for both links to fully rotate.

Crashing.

See dislikes above.

Its hard to make anything if it crashes every 10 seconds, if it never crashed it would get an easy 4.

This software seems like a big step up from the desktop version. It is a lot more refined and has more intuitive. UI features.

Please list any recommendations for features and improvements you would like to see in PMKS+ Web. 18 responses

I would like to see a way for how to set exact lengths for links or exact coordinates of joints, or instructions on how to do so if this is a feature already.

Have a chart of what each type of link does and how to get 1 DOF.

Fix the crashing.

a way to better make sliders and bars that can be used for trass lines.

It having everything there.

Make the table easier to edit.

Make it easier to add nodes to the links like in the desktop.

An improvement could be the responsiveness of the website.

Add CTRL+Z function. Fix the edit tables to work properly.

Please just make it stop freezing and crashing. Feature wise its very well done, maybe adding some more shape customization. Maybe add a feature where you trace a shape and it auto generates linkages to follow that shape.

Linkage Atlas and Matlab tie in. Can you add the color of the link in the chart.

n/a.

Just make it easier to use the tracer feature.

Please fix the table and add an undo button or ctrl+z functionality.

I think an export to SolidWorks button like in the offline version would be cool.

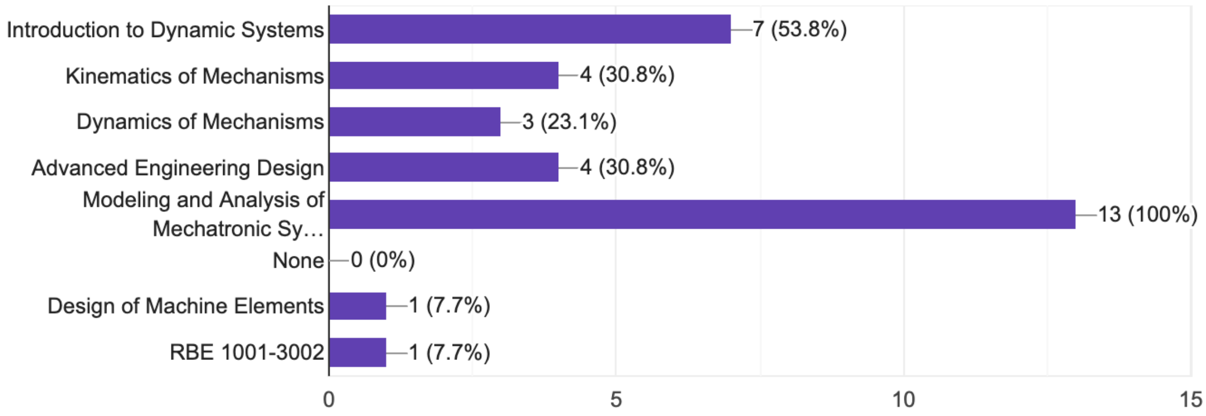
There is significantly less problems with crashing on the web version. I had to restart the windows application at least 5 or 6 times during testing but the web version only needed to be refreshed once. The consistency of the web version was much better but it would be good to try to reduce this to zero.

Make the tables able to be edited.

Make the tutorials more detailed.

The velocity and acceleration graph feature from the desktop version is very nice to have. Also the ability to rotate a shape or link about its axis would be a nice addition.

A.2 Working Model First Evaluation Response:



Please select which relevant courses you have taken. 13 responses

What did you find difficult in Working Model? 13 responses

Working model was pretty hard to understand, not very intuitive. Also when creating the linkage for a infinity the members started to separate, which produced some errors.

I've never used it before, so every step was difficult. The most difficult things were properly sizing links, connecting links via joints at the correct location, and adding points of torque (which i still cannot get to work).

The links could not be moved independently from the joints, the snap components did not snap where it made sense, it was difficult to place the links where you wanted them to be, the tools were not intuitive and were difficult to find, there was a steep learning curve, and there was no way to deal with singularities in a specific fashion.

Connecting links together.

Getting the linkage to stay so that the drive bar and output bar are always opposite of each other to create the infinity symbol instead of turning into a parallelogram.

I found configuring the lengths of the linkages to be especially tough.

The inability to rotate links is very difficult to work around and the linking joint feature makes this program take an incredibly long time to use. also the fact that linkage members were to be made out of shapes that needed to be traced caused some difficulty.

A lot of errors when trying to run a parallelogram 4-bar, it wouldn't travel along the desired path. Sometimes it stops the simulation but doesn't tell the user why.

Making sure that the mechanism animates properly (everything is connected properly). Not the most intuitive software, but it's okay.

I'm not sure if it is due to the Remote Desktop but everytime I ran the simulation it would crash within a few frames. It is also very hard to find the proper preferences for each of the environment variables and even harder to adjust the parameters of individual linkages.

Panning Around the workspace.

Just about everything. This program is not designed to be user friendly in any way. Panning involves extra mouse clicks, connections between elements aren't preserved if one of them is re-positioned, adjusting link geometry is convoluted, and to top it off, the program crashes so often it isn't worth using most of the time.

I haven't used this program in a while so I forgot where things were such as "non-collision setting".

What did you feel was easy in Working Model? 13 responses

Making the linkages to the designated length and coordinates, coordinate snapping feature is nice.

nothing, I'm sure if I spent more than the hour I did for this activity I would be more comfortable with the program.

I don't think any part of it was "easy", the only easy part was defining the width and length of the links and the torque output of the motors.

drawing link shapes.

creating the each bar a certain length and joining them together I felt was very easy to do.

Making the tracing and such.

setting the dimensions of links.

Its easy to change the geometry of the links. It's also easy to place new joints on the links. The link, joint, and motor settings are easily accessible and user friendly.

It's easy to define joints simply by moving around the plane.

The UI seems nice at first and dragging and dropping the linkages, joints, and actuators to attach were very easy.

Putting Motors, Defining Joints.

Nothing.

making linkages, locating shapes and joints.

What did you find difficult in Working Model? 13 responses

Working model was pretty hard to understand, not very intuitive. Also when creating the linkage for a infinity the members started to separate, which produced some errors.

I've never used it before, so every step was difficult. The most difficult things were properly sizing links, connecting links via joints at the correct location, and adding points of torque (which i still cannot get to work)

The links could not be moved independently from the joints, the snap components did not snap where it made sense, it was difficult to place the links where you wanted them to be, the tools were not intuitive and were difficult to find, there was a steep learning curve, and there was no way to deal with singularities in a specific fashion

Connecting links together

Getting the linkage to stay so that the drive bar and output bar are always opposite of each other to create the infinity symbol instead of turning into a parallelogram.

I found configuring the lengths of the linkages to be especially tough.

The inability to rotate links is very difficult to work around and the linking joint feature makes this program take an incredibly long time to use. also the fact that linkage members were to be made out of shapes that needed to be traced caused some difficulty

A lot of errors when trying to run a parallelogram 4-bar, it wouldn't travel along the desired path. Sometimes it stops the simulation but doesn't tell the user why.

Making sure that the mechanism animates properly (everything is connected properly). Not the most intuitive software, but it's okay.

I'm not sure if it is due to the Remote Desktop but everytime I ran the simulation it would crash within a few frames. It is also very hard to find the proper preferences for each of the environment variables and even harder to adjust the parameters of individual linkages.

Panning Around the workspace

Just about everything. This program is not designed to be user friendly in any way. Panning involves extra mouse clicks, connections between elements aren't preserved if one of them is re-positioned, adjusting link geometry is convoluted, and to top it off, the program crashes so often it isn't worth using most of the time.

I haven't used this program in a while so I forgot where things were such as "non-collision setting"

What did you feel was easy in Working Model?

13 responses

Making the linkages to the designated length and coordinates, coordinate snapping feature is nice.

nothing, I'm sure if I spent more than the hour I did for this activity I would be more comfortable with the program.

I don't think any part of it was "easy", the only easy part was defining the width and length of the links and the torque output of the motors

drawing link shapes

creating the each bar a certain length and joining them together I felt was very easy to do.

Making the tracing and such.

setting the dimensions of links

Its easy to change the geometry of the links. It's also easy to place new joints on the links. The link, joint, and motor settings are easily accessible and user friendly.

It's easy to define joints simply by moving around the plane.

The UI seems nice at first and dragging and dropping the linkages, joints, and actuators to attach were very easy.

Putting Motors, Defining Joints

Nothing.

making linkages, locating shapes and joints

What do you like about Working Model? (This can include qualities that you know from usage outside of this evaluation.) 13 responses

Coordinate snapping feature.

Nothing, as I am unaware of its uses and found it to take longer than if I were to create and analyze the linkage by hand.

I really did not like it, but it seemed particularly versatile and powerful overall.

somewhat simple.

I liked how all of the elements and joints needed were easy to find along the left side of the window. I also liked the grid view and being able to look at the coordinates you were using to create the linkage.

I found its images on the left toolbar easy to read and understand, as well as its playback feature and ability to customize.

The ability to set dimensions of links improves its use entirely.

Working model is useful for generating a simple linkage and track how the linkage behaves. I like how it allows you to edit each link on the fly with ease. I also like the various different tracking methods that are available.

There are tons of feedback that you can receive back from Working Model.

Honestly, there is very little I could appreciate about it since the simulation crashed every few seconds.

Motion Visualization, Analysis of Speed, Torque.

Still nothing. I greatly dislike this program.

when you know where everything is, its fairly easy to use.

What did you find difficult in PMKS+ Web? 13 responses

The tracing feature was easy enough to understand but due to the degree of freedom limit i could not do the regular linkage for an infinity.

Crashed multiple times when adjusting linkage sizes by dragging links. Could not edit links by pressing joints>edit.

There is no way to accurately place the individual joints on the links, so you have to roughly place them just based on eye. There is an edit button for location but it does not work.

Slider does not work so could not complete infinity tracing.

The website crashed every time I tried to modify the lengths of the bars after the input and ground were set.

The closest I could get to making the infinity symbol was by getting only half of it. I also found it difficult to modify the lengths to get them to were I wanted them to be. Seems like there is an edit button in the table where I can type in the lengths to get them exactly how long I want them but nothing happens when I click the edit button.

I found moving points difficult, as I expected them to rotate, but instead they just moved, modifying the length of the link.

The system kept crashing for me when i was trying to create the linkage.

It wouldn't let me edit individual point coordinates, so I had to click and drag to get the desired coordinates. Since it is very sensitive, getting exact coordinates was difficult. Also, when I wanted to click and drag a joint within a shape, it would update for a couple seconds but then the software would freeze and then crash, causing me to have to restart.

A lot of times, when you move around a joint when a mechanism is defined (DOF = 1), the web app will crash :/

Again I have spent over 4 hours attempting to test each of these softwares and for the web version of this application everything I moved a joint to a different location there was a 90% chance it would crash making it impossible for me to test. The Desktop version of this works beautifully however and I did the evaluation for it earlier and was able to create the infinity loop there.

The website kept crashing as I was trying to create links, which made the process harder.

Editing link shapes is a little clunky. Creating a trace point takes too many steps as well. Adding a link just to delete it is redundant.

I couldn't figure out how to animate the linkages.

What did you like about PMKS+ Web? 13 responses

How intuitive and straightforward it is. Help resources online are great.

quick for simple linkages, easy to use animation.

it is much more intuitive and easy to use

organized interface

I liked the grid view with coordinates that made it easy to see the scale of each bar that was made.

Definitely the ability to quickly find the right point to make the required shape.

I liked the ability to rotate links and move them as well as the active tracing feature when adjusting the links

I like the look of the software. I also like the change shapes feature because you don't have to add additional joints since it does it for you. I also like the click drag feature to change link positions. I like panning by clicking. The reset view button is also very helpful. Lastly, the live updated paths are very helpful as well.

I really liked how the user could just easily move around a joint, especially when a shape was not just a bar.

The UI again is the same as the desktop version except for creating and modifying links. The edit linkage button for me did not work whatsoever.

It had the same capabilities as PMKS Desktop on the Web which is hard to achieve using javascript.

Automatically showing the joint paths is a nice feature.

I like working model better

What did you dislike about PMKS+ Web? 13 responses

I encountered a bug that broke the software when using the trace feature. It would freeze on the move icon and i would need to open up a new browser and start over. This happened around 15 times.

Very buggy, editing link dimensions not working and several crashes.

It crashed everytime i tried to start moving the joints that were anchored around. It also was difficult to get everything where i needed it to be to properly test.

Toggle slider feature does not work. Poor documentation.

Maybe it was just that my laptop couldn't handle the website but every time I tried to modify the length of a bar after the input and ground were set the website crashed.

It kept running out of memory when I was experimenting with it, as well as the "toggle" of certain things, as I sometimes did not know which way it was toggled.

The difficulty when it came to dimensioning the links as well as the repeated crashing.

I did not like that I couldn't change the joint positions, however I understand the edit feature will be coming in future versions. I also didn't like that I couldn't change the width of the links. I didn't like that there is no undo ability or select all element.

Beside the web crashing, I wish the user could be able to add joints onto a link rather than to just add joints onto joints (don't mean making a triangle and then adding a joint onto that).

Again did not work.

Kept Crashing

Custom link shapes are missing, and the premade links can only be scaled with the same aspect ratio. They can't be elongated/shrank along a single axis.

hard to figure out how to de-bug the errors, I couldn't set a linkage as an input after I created it I had to delete it and then create an input.

Please list the top 3 features you like in PMKS+ Web. 13 responses

Linkage connecting features. Shape features. And Tracing feature.

Easy animation, linkage analysis, visual of link movement while editing the linkage

right click for editing, easy link creation from previous joints, more intuitive set up

Shape choices, grid system, copy URL sharing

the grid view, being able to see the trace lines before the simulation, right clicking to change the joint to input or ground.

1. Preview of Path

2. Web Based

3. Degree of Freedom Counter.

Linkage rotation, forming of links, tracing feature in real time.

Change shape, joint type toggle, and the create link feature.

I really like how easy it is to export all kinds of data mechanisms that are not slider. I really like the page showing all of the different possible links a user can use. I really like how the user can add forces onto a link.

The help button and tutorials are useful and I was able to use the static analysis which was helpful. I also like the fact that you can save individual graphs instead of having to save the entire model.

Creating Links, Toggling Ground, Setting Input.

Link creation, joint tracing, and assigning link parameters (inputs, grounds, etc).

easily create a linkage, easily change the shape of a linkage.

Please list any recommendations for features and improvements you would like to see in PMKS+ Web. 13 responses

Iron out a few bugs.

More stable web app.

an easier delete and undo button. Functionality of the edit and delete buttons in the table

Fix sliding joint feature

Put in a feature to type in the lengths of the bars so you can make them exactly how long you want. Are there certain specs that your computer should have to run the website without crashing?

Add UI elements for the right-click options, so I can left click on the part and have a box with all the options listed.

As said above improve the stability of running the program and add a dimension option to links

I recommend fixing the joint coordinate edit features, as well as adding the link geometry edit. Being able to edit the geometry of one link without changing the connection point is important. In addition, a way to select multiple joints or links at a time. Also, the create joint feature should allow for a creation of a point on a link without having to create the link, delete it, then move the joint around.

Have graphs that show how kinematics change for a joint after a certain amount of time. Different tabs (joints, links, forces) that give you information for each of these three. I like the information that is given out on the joint but think the chart should be writable.

Again not sure why it was constantly crashing on my side.

SolidWorks Export, Crash Handling.

See my previous comments.

I prefer a drop down menu , easier access, of all the different options.

Please write any comments you have on how PMKS+ Web compares to Working Model

13 responses.

It is monumentally better, very easy to understand and start working with in comparison to Working Model.

Much easier to use, seems to have less features.

SO so much better for this particular process, i wish PMKS+ was a finished product.

Working Model seems more robust.

I liked the ability in working model to change the links to the exact size I wanted them by typing them in.

I found the way of adding joints and sources more intuitive in working model, but much easier to add and have work properly in PMKS+.

PMKS+ Web works significantly better than working model though a bit less stable.

I think that this software has several elements that are better than working model such as the change shape feature, joint movement feature, panning with a simple click, and joint type toggling. That being said, there need to be some added features to make it easier for the user such as link geometry, coordinate editing, point creation within links, and an undo feature that working model already has. With a little added features I think this software can really compete with working model.

PMKS+ Web definitely feels a lot more intuitive using than Working Model. Rather than dragging joints to the grid, the user can easily create joints just by right clicking. I really like how links mechanisms can be stored through a link URL and Saved as an Excel file (I know Working Model is a Desktop application but still).

Both were equally difficult to use due to crashing frequently but the UI for PMKS is much easier to understand but the Working Model offers more options. The Desktop version of PMKS is very easy to use and works better than Working Model.

PMKS+ Web has a more easy to use UI and has a much better flow than Working Model.

Leaps and bounds better. For one, PMKS+ Web actually functions. The UI is much more user friendly.

Working Model is more straight to the point and you can easily figure everything you need.