



**PMKS+Desktop:
Modernizing and Expanding a
Legacy Silverlight Application**

*A Major Qualifying Project submitted to the Faculty of
Worcester Polytechnic Institute in partial fulfillment of the
requirements of the degree of Bachelor of Science*

Written By:

Peter Prygocki (CS)

Fabian Gaziano (CS)

Advisors:

Professor David C. Brown (CS)

Professor Pradeep Radhakrishnan (ME)

May 18, 2020

I. Abstract

The goal of this project was to recreate the Planar Mechanism Kinematic Simulator (PMKS), a legacy Silverlight application, as a modern desktop application with additional features. The recreation, named PMKS+Desktop (PMKS+D), contains user interface elements and functionality adapted from PMKS and PMKS+, a previous recreation of PMKS. The team used modern desktop development tools and software to create PMKS+D to ensure the longevity of the application. By conducting evaluations to test the user interface and our new functionality, the team was able to demonstrate that PMKS+D has improved upon both the functionality of PMKS and the user interface experience of PMKS+.

II. Acknowledgements

First and foremost, we would like to thank Professor David C. Brown and Professor Pradeep Radhakrishnan for providing guidance throughout this project. We would also like to thank Trevor Dowd, Robert Dutile, and Haofan Zhang of the PMKS+ Web team for coordinating and working with us throughout the project. We would like to thank Alex Galvan and Dr. Ahmet C. Sabuncu for providing additional support. We would also like to thank Dr. Matthew Campbell of Oregon State University for creating the open-source application, PMKS, on which our work is built. Finally, we would like to thank the students of the Mechanical and the Robotics Engineering departments of WPI who participated in our user studies and helped us improve the user experience in our application.

III. Table of Contents

I. Abstract	1
II. Acknowledgements	2
III. Table of Contents	3
IV. List of Figures and Tables	8
V. Authorship	13
1. Introduction	17
2. What is PMKS and PMKS+?	19
2.1. What are Linkages?	19
2.2. What is PMKS?	21
2.3. What is PMKS+?	22
3. Research Statement and Goals	25
4. Methodology	27
4.1. Understanding the Project	27
4.1.1. Understanding Linkages	27
4.1.2. Understanding PMKS and PMKS+	29
4.1.3. Understanding the Requirements	29
4.2. Desktop Application Justification	30
4.3. Literature and Application Research	31
4.4. Exploring WPF	31
4.5. Creating PMKS+Desktop	32
4.5.1. C# Documentation and Naming Conventions	32
4.5.2. Reformatting the WPF Coding Explorations	32
4.5.3. Creating a New User Interface	33
4.5.4. Developing Additional Functionality	33
4.6. User Evaluations	34
4.6.1. Designing the Evaluation	34
4.6.2. Designing the Data Analysis Script	35
4.6.3. Performing the Evaluation	35
4.6.4. Performing the Analysis	36

5. Literature Review	37
5.1. UI Design	37
5.1.1. Usage-Centered Design	37
5.1.2. User-Centered Design	40
5.1.3. UI Design Rules	42
5.2. Metrics for UI Evaluation	43
5.2.1. The Most Common Metrics	43
5.2.2. Additional Metrics	45
5.3. Heuristic UI Evaluation	47
5.4. User Evaluation Design	49
5.4.1. Requirements for an Evaluation	49
5.4.2. Moderating the Evaluation	52
5.4.3. Data Collection	55
5.4.4. Data Analysis	56
5.5. Other Linkage Simulation Applications	59
6. Desktop Application Justification	61
6.1. Why PMKS Needs to Be Replaced	61
6.2. Why PMKS+ Needs a Desktop Alternative	62
6.2.1. Ease of Accessibility	62
6.2.2. Incomplete Functionality	63
6.2.3. Integration with Other Desktop Applications	63
6.3. Why Windows?	64
6.4. Why Windows Presentation Foundation?	65
6.4.1. Why not Win32 API?	65
6.4.2. Why not Windows Forms?	66
6.4.3. WPF or UWP?	66
7. WPF Coding Explorations	68
7.1. Guru99 C# Tutorial	68
7.2. Window and Display Creation in WPF	68
7.2.1. Draw a Coordinate Plane	69
7.2.2. Start Program Maximized	70
7.2.3. Create Window Subsections	71
7.2.4. Zoom In and Zoom Out	72
7.3. Functionality	74
7.3.1. Clock	74

7.3.2. Rotating a Link From Points	76
7.3.3. Drawing a Link Using Composite Geometries	78
7.3.4. Creating Curves	80
7.3.5. TextBox Link	83
7.3.6. Updating the Link and Curve From TextBoxes	84
7.3.7. Click-and-Drag	86
8. PMKS+ Interface	88
8.1. PMKS+ Layout Justification	89
8.2. Linkage Creation Justification	92
9. Interface Design	93
9.1. Tables	93
9.1.1. Joints	94
9.1.2. Links	95
9.1.3. Forces	96
9.2. Canvas	98
9.2.1. Graphics	98
9.2.2. Click-and-Drag	102
9.2.3. Context Menu	104
9.2.4. Mouse Scroll	106
9.3. Lower Menu	107
9.3.1. Animation Control	107
9.3.2. Canvas Control	109
9.4. Upper Menu	110
9.4.1. File	111
9.4.2. Analysis	112
9.4.3. Settings	115
9.4.4. Help	118
9.5. Applying UI Evaluation Metrics	122
9.5.1. Essential Efficiency	122
9.5.2. Task Visibility	127
9.5.3. Visual Coherence	132
10. System Design	138
10.1. Model-View-ViewModel Design Pattern	138
10.2. Model	139
10.2.1. Joints	139

10.2.2. Links	142
10.2.3. Forces	143
10.2.4. Draw Linkage	145
10.3. MainWindow	145
10.3.1. Properties	145
10.3.2. Event Handlers	147
10.3.3. Update Simulator	148
10.3.4. Draw Linkage When Incomplete	149
10.3.5. Open and Save Linkage	149
10.4. Additional Features	151
10.4.1. SolidWorks Export	151
10.4.2. Graphing	153
11. Application Testing	156
11.1. Example Linkage Testing	159
11.2. Individual UI Feature Testing	160
11.3. Response to Application Testing	163
11.3.1. Example Linkage Testing	163
11.3.2. Individual UI Feature Testing	165
12. User Evaluation Design	167
12.1. Focus Groups	167
12.2. Design of User Evaluation Survey	168
12.2.1. Demographics	170
12.2.2. PMKS+Desktop Tasks	170
12.2.4. Open Response and Follow-Up Questions	173
12.2.5. Concluding Questions	174
12.3. Design of Analysis Procedure	175
12.3.1. User Categories	176
12.3.2. Image Upload	176
12.3.3. Likert Scale	180
12.3.4. Open Response	180
13. Data Analysis	181
13.1. Focus Group	181
13.2. User Evaluation	182
13.2.1. User Categories	182
13.2.2. Image Uploads	182

13.2.3. Open Response	183
13.2.4. Rating and Follow-Up Questions	189
13.3. Evaluation of Analysis	194
13.3.1. User Categories	194
13.3.2. Image Uploads	194
13.3.3. Open Response	196
13.3.4. Rating and Follow-Up Questions	198
14. Conclusions	200
14.1. Evaluation of the Project	200
14.1.1. Achievement of Goals	200
14.1.2. User Evaluation Achievements	202
14.2. Future Work	203
14.2.1. Unfinished Features	204
14.2.2. Future Features	205
14.3. Project Experience	206
14.3.1. Applied Skills	207
14.3.2. Applied Knowledge	207
14.3.3. Gained Skills	208
14.3.4. Gained Knowledge	209
14.4. Overall Evaluation	211
15. References	212
Appendices	217
Appendix A: Notes on Linkage Simulation Applications	217
Appendix B: User Evaluation Script	227
Appendix C: Participant Responses to Open Response Questions	234
Appendix C.1.: What Was Difficult in PMKS+Desktop	234
Appendix C.2.: What Was Easy in PMKS+Desktop	238
Appendix C.3.: What Was Liked in PMKS+Desktop	241
Appendix C.4.: What Was Disliked in PMKS+Desktop	244
Appendix C.5.: Top 3 Features in PMKS+Desktop	247
Appendix C.6.: Comparison of PMKS+Desktop to Other Software	251
Appendix C.7.: Recommendations for Improving PMKS+Desktop	253

IV. List of Figures and Tables

Figure 2.1. Linkage 2 as a physical linkage (left) and simulated in PMKS (right)	20
Figure 2.2. The UI of PMKS+	23
Figure 4.1. Linkage 2 as a physical linkage (left) and simulated in PMKS (right)	28
Table 5.1. User-centered design vs. usage-centered design	38
Table 5.2. Techniques used in the user-centered design process	41
Table 5.3. Potential quantitative and qualitative data for each usability dimension	44
Table 5.4. Usability heuristics created by Jakob Nielsen (1994)	48
Figure 7.1. The same linkage with different scaling within the WPF application	73
Figure 7.2. Dayyan’s analog clock in WPF	75
Figure 7.3. A Line in mid-rotation within the WPF application	77
Figure 7.4. A rotating link in PMKS	78
Figure 7.5. A link created from composite Geometries rotating inside the WPF application	80
Figure 7.6. A circular path generated in our WPF exploration application	82
Figure 7.7. (left) A non-circular path generated in PMKS, (right) The same path generated in the WPF exploration application	83
Figure 7.8. TextBoxes updated as an Ellipse is dragged	84
Figure 7.9. (left) The default position and dimensions of the link and circle, (right) The link and circle after updating Point A and Point B	85
Figure 7.10. A blue dot before (left) and after (right) click-and-drag is performed	87

Figure 8.1. Example of a linkage in PMKS Silverlight	89
Figure 8.2. File and Analysis tabs in PMKS+	90
Figure 8.3. PMKS+'s updated lower menu	91
Figure 8.4. The final iteration of PMKS+'s UI	91
Figure 9.1. Joints table in PMKS+D containing two example joints, A and B	95
Figure 9.2. Links table in PMKS+D with two example links, 1 and 2	96
Figure 9.3. Forces table in PMKS+D with an example force, F1	97
Figure 9.4. The two types of links in PMKS and PMKS+D	99
Figure 9.5. A ground joint with a ground graphic in PMKS+D	99
Figure 9.6. A linkage with a slider in PMKS	100
Figure 9.7. A prismatic joint with a slider graphic in PMKS+D	101
Figure 9.8. A prismatic joint with a slider and track graphic in PMKS+D	101
Figure 9.9. A force graphic on a link in PMKS+D	102
Figure 9.10. Examples of hitbox graphics in PMKS+D	103
Figure 9.11. Examples of context menus in PMKS+D	106
Figure 9.12. The animation controls in PMKS+D	109
Figure 9.13. The Canvas controls in PMKS+D	110
Figure 9.14. The File tab in PMKS+D	111
Figure 9.15. The Analysis tab in PMKS+D	112
Figure 9.16. The Export Data button's context menu	113
Figure 9.17. Graph mockup for the x-velocity of an example joint	115
Figure 9.18. The Settings window in PMKS+D	115

Figure 9.19. The Properties window from PMKS adapted into PMKS+D	117
Figure 9.20. The Help tab in PMKS+D	118
Figure 9.21. The Tutorial window in PMKS+D	119
Figure 9.22. Intermediate screen used to select the wanted help document	120
Figure 9.23. The Toolbar Help window in PMKS+D	121
Figure 9.24. The final iteration of PMKS+D's UI	122
Table 9.1. Visibility and reasoning for each task of creating a 4-bar linkage in PMKS+D	128
Table 9.2. Visibility and reasoning for each task of opening an angular velocity graph in PMKS+D using the Analysis tab method	129
Table 9.3. Visibility and reasoning for each task of opening an angular velocity graph in PMKS+D using the context menu method	130
Table 9.4. Visibility and reasoning for each task of exporting a linkage to SolidWorks in PMKS+D	131
Table 9.5. Relatedness of each pair of UI elements within the Joints table	132
Table 9.6. Relatedness of each pair of UI elements within the animation control menu	133
Table 9.7. Relatedness of each pair of UI elements within the Canvas control menu	134
Table 9.8. Relatedness of each pair of UI elements within the File tab	134
Table 9.9. Relatedness of each pair of UI elements within the Analysis tab	135
Table 9.10. Relatedness of each pair of UI elements within the Settings tab	136
Table 9.11. Relatedness of each pair of UI elements within the Help tab	137
Figure 10.1. A linkage with a slider in PMKS	141
Figure 10.2. Simplified class diagram of PMKS+D's Model	144

Table 10.1. Important properties within MainWindow	146
Figure 10.3. 4-Bar linkage in PMKS+D and SolidWorks	152
Figure 10.4. The x velocity of a joint with a datapoint clicked	155
Figure 11.1. 4-Bar revolute linkage in PMKS+D	157
Figure 11.2. 6-Bar linkage with a force in PMKS+D	158
Figure 11.3. 4-Bar slider crank linkage in PMKS+D	158
Figure 12.1. Image of the linkage provided in the user survey	171
Figure 12.2. Expected angular velocity graph for user evaluation submission	172
Figure 12.3. Expected SolidWorks CAD model for user evaluation submission	173
Figure 12.4. Correct recreation of example user evaluation linkage	177
Figure 12.5. The correct angular velocity graph for the user evaluation	178
Figure 12.6. Correct CAD model from SolidWorks export	179
Figure 13.1. Percentage of users for each feature liked in PMKS+D	184
Figure 13.2. Percentage of users for each feature disliked in PMKS+D	185
Figure 13.3. Percentage of users for each easy feature in PMKS+D	186
Figure 13.4. Percentage of users for each difficult feature in PMKS+D	187
Figure 13.5. Percentage of users that placed each feature in the top 3 features of PMKS+D ...	188
Figure 13.6. Percentage of users that rated each category of the linkage creation and editing speed	189
Figure 13.7. Percentage of users that gave each Likert Scale ranking for overall satisfaction with PMKS+D	191

Figure 13.8. Percentage of users that gave each response comparing PMKS+D to Working Model, PMKS, and PMKS+ 192

Figure 13.9. Percentage of users that gave each recommendation for PMKS+D 193

Figure 13.10. A nearly-correct linkage uploaded 195

V. Authorship

Note: The listed author(s) wrote all subsections under the listed chapter headings unless stated otherwise. For example, Fabian wrote all of Chapter 6.2.: Why PMKS+ Needs a Desktop Alternative except for Chapter 6.2.2.: Incomplete Functionality, which Peter wrote.

<u>Chapter:</u>	<u>Author(s):</u>	<u>Editor(s):</u>
I. Abstract	Fabian	Peter
II. Acknowledgements	Fabian	Peter
III. Table of Contents	Fabian, Peter	Id.
IV. List of Figures	Peter	Fabian
V. Authorship	Fabian, Peter	Id.
1. Introduction	Fabian	Peter
2. What Is PMKS and PMKS+?	Fabian	Peter
2.1. What are Linkages?	Peter	Fabian
2.2. What is PMKS?	Peter	Fabian
2.3. What is PMKS+?	Fabian	Peter
3. Research Statement and Goals	Peter, Fabian	Id.
4. Methodology	Peter	Fabian
5. Literature Review	Fabian	Peter
5.1. UI Design	Fabian	Peter
5.2. Metrics for UI Evaluation	Peter	Fabian
5.3. Heuristic UI Evaluation	Peter	Fabian

5.4. User Evaluation Design	Peter	Fabian
5.5. Other Linkage Simulation Applications	Peter	Fabian
6. Desktop Application Justification	Fabian	Peter
6.1. Why PMKS Needs to Be Replaced	Fabian	Peter
6.2. Why PMKS+ Needs a Desktop Alternative	Fabian	Peter
6.2.2. Incomplete Functionality	Peter	Fabian
6.3. Why Windows?	Peter	Fabian
6.4. Why Windows Presentation Foundation?	Peter	Fabian
7. WPF Coding Explorations	Peter	Fabian
7.1. Guru99 C# Tutorial	Fabian	Peter
7.2. Window and Display Creation in WPF	Fabian	Peter
7.2.1. Draw a Coordinate Plane	Fabian	Peter
7.2.2. Start Program Maximized	Peter	Fabian
7.2.3. Create Window Subsections	Fabian	Peter
7.2.4. Zoom In and Zoom Out	Fabian	Peter
7.3. Functionality	Peter	Fabian
7.3.1. Clock	Peter	Fabian
7.3.2. Rotating a Link From Points	Peter	Fabian
7.3.3. Drawing a Link Using Composite Geometries	Peter	Fabian
7.3.4. Creating Curves	Peter	Fabian
7.3.5. TextBox Link	Fabian	Peter
7.3.6. Updating the Link and Curve From TextBoxes	Peter	Fabian

7.3.7. Click-and-Drag	Fabian	Peter
8. PMKS+ Interface	Fabian	Peter
9. Interface Design	Fabian	Peter
9.1. Tables	Fabian	Peter
9.2. Canvas	Fabian	Peter
9.3. Lower Menu	Peter	Fabian
9.4. Upper Menu	Peter	Fabian
9.5. Applying UI Evaluation Metrics	Peter	Fabian
10. System Design	Peter	Fabian
10.1. Model-View-ViewModel Design Pattern	Peter	Fabian
10.2. Model	Fabian	Peter
10.3. MainWindow	Fabian	Peter
10.3.1. Properties	Peter	Fabian
10.3.2. Event Handlers	Fabian	Peter
10.3.3. Update Simulator	Fabian	Peter
10.3.4. Drawing When Incomplete	Fabian	Peter
10.3.5. Open and Save Linkage	Peter	Fabian
10.4. Additional Features	Peter	Fabian
11. Application Testing	Robert Dutile	Peter
11.3. Response to Application Testing	Peter	Fabian

12. User Evaluation Design	Fabian	Peter
12.2.3. Likert Scale	Peter	Fabian
13. Data Analysis	Fabian, Peter	Id.
14. Conclusions	Fabian, Peter	Id.
15. References	Peter	Fabian
Appendices	Peter	Fabian

1. Introduction

Programs used in mechanical and robotics engineering courses are often multipurpose, computationally expensive, and complex. It is a necessity that these applications be efficient, have an intuitive user interface (UI), and a satisfying user experience (UX). However, as these applications age, industry-wide improvements to UI and UX might make them uncomfortable for new users. Additionally, an application might become unsafe or impossible to use if its support is dropped. In these cases, it becomes necessary for a new application to be created to replace the old one.

The Planar Mechanics Kinematic Simulator (PMKS) is a web application designed to offer a platform for users to create and analyze two dimensional, or planar, linkages (Campbell, 2014). PMKS was implemented using Silverlight, an obsolete web application format created by Microsoft (Microsoft Corporation, 2011). This application is used by the Mechanical Engineering Department at Worcester Polytechnic Institute (WPI) to aid the study of planar linkages. Due to its obsolete nature, as well as with Microsoft's support for Silverlight ending in 2021, PMKS needed to be updated to a more modern format.

A newer version of PMKS called PMKS+ was developed by a previous MQP team, which contains a new UI and mouse-based linkage creation system (Appikatla, et al., 2019). While PMKS+ partially brought the outdated software into the present day, PMKS+ had other limitations due to its web-based nature, such as ease of accessibility and incomplete functionality. In order to address these limitations and evolve PMKS further, our team was tasked

with creating a new desktop application to replace PMKS. Named PMKS+ Desktop (PMKS+D), our application contains the features of PMKS with the improved UI and UX of PMKS+ to deliver an effective learning tool for modern college students.

Chapter 2. of our report explains the background of PMKS and PMKS+ in more depth. Chapter 3. describes our team's research statement and goals created for this project. Chapter 4. discusses the methodology the team used in the various tasks performed throughout the project. The literature review given in Chapter 5. describes the introductory research conducted by the team. Chapter 6. describes the justification for the desktop application format used to create PMKS+D. Chapter 7. discusses the coding exercises that were necessary for the team to become familiar with coding in that format. Chapter 8. provides more information about the PMKS+ interface and the design decisions that were made by the PMKS+ team. The interface design of PMKS+Desktop is covered in Chapter 9., and Chapter 10. provides an overview of PMKS+D's system design. Chapter 11. describes the results of application testing. Chapter 12. discusses the design of the user evaluations performed by our team. Chapter 13. discusses the data collected from these evaluations and our analysis of the data.. Finally, our conclusions on the project and future work are discussed in Chapter 14.

2. What is PMKS and PMKS+?

The Planar Mechanical Simulator (PMKS) and PMKS+ are modeling software systems for creating and simulating two-dimensional models of linkages. To fully understand these applications, linkages and their importance must first be understood.

2.1. What are Linkages?

Linkages are assemblies of rigid bars called links connected with joints to form one or more closed chains, i.e., a collection of links with at least two joints fixed in place. Linkages are designed to transform one kind of motion into another. For example, windshield wipers are linkages that transform the circular motion of a motor into oscillating motion of the blades. Linkages do not have to be driven from rotating motors; car engines transform oscillating, linear motion of the pistons into circular motion of the wheels (Zhang, Finger, & Behrens, n.d.; Rothenhofer, 2009).

There are various kinds of linkages, but there are two main categories: planar linkages and spatial linkages (Rothenhofer, 2009). Planar linkages are linkages whose movement occurs in one plane. Therefore, planar linkages can be represented in 2D by abstracting the third dimension with only minor visual overlapping of the links. These are the kinds of linkages that PMKS simulates. In Figure 2.1. below, you can see an example of a planar linkage represented in PMKS.

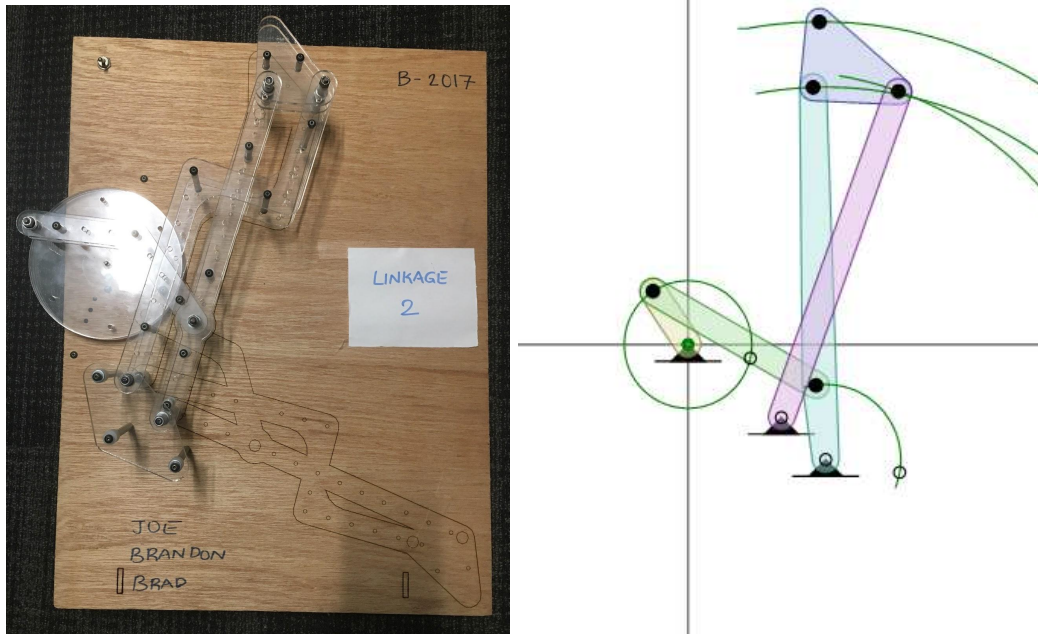


Figure 2.1.: Linkage 2 as a physical linkage (left) and simulated in PMKS (right).

Spatial linkages are linkages that occur in three dimensions; e.g., a robot's arm can move in any of the three dimensions. Therefore, one of the three dimensions cannot be abstracted without removing important information about the linkage's movement. For this reason, spatial linkages are not supported by PMKS.

Linkages are important in a variety of fields, including sports equipment, vehicles, construction, manufacturing, precision machinery, and medical devices. Users of this equipment rely on engineers to design them, keep them safe, maintain them, and ensure their effectiveness. Therefore, it is crucial for engineers to be able to understand linkages.

For this reason, applications like PMKS are important for providing an environment where planar linkages are simplified by the abstraction of the third dimension. Such systems

enable engineering students to create, view, and simulate models of planar linkages without considering the third dimension or working with physical materials.

2.2. What is PMKS?

The Planar Mechanical Kinematic Simulator (PMKS) is a Silverlight application that simulates the kinetics of planar mechanisms. Users can define joints, links between joints, and forces to apply to the links. PMKS displays a graphical representation of the assembly as well as an animation that demonstrates the range of motion. The software is open source, allowing for students and developers to freely access it and improve on it (Appikatla, et al., 2019, p. 10).

PMKS was first developed as a kinematics software system by Dr. Matthew Campbell from Oregon State University (OSU) and is currently maintained by OSU's Design Engineering Lab (Campbell, 2014). WPI students integrated kinetics functionality into PMKS as their MQP, which permitted users to apply forces to linkages for static and dynamic force analyses (Andrews, et al., 2018; Appikatla, et al., 2019, p. 10).

PMKS was originally developed in Microsoft Silverlight, a derivative of the Visual C# and Visual Basic programming languages using the .NET Framework (Microsoft Corporation, 2011). Silverlight's intent was to provide additional tools to make developing Web and mobile applications easier. Silverlight is compatible with XAML, a code format that defines the UI for .NET Core applications (De George & Victor, 2019).

Silverlight is strictly compatible with Internet Explorer 10 and 11, meaning that users can only run PMKS on Windows machines and use more modern, more advanced browsers

(Microsoft Corporation, 2011). Additionally, Microsoft is ending support for Silverlight in October, 2021, making PMKS completely inaccessible (Microsoft Corporation, 2011).

However, as explained in Chapter 2.1.: What are Linkages?, PMKS fills an important role for engineers that should not be abandoned. Currently, PMKS is used in multiple courses at WPI and OSU (Appikatla, et al., 2019, p. 10). Therefore, our advisors have identified the need to recreate PMKS in a newer language and modernize it so more students can have access to it in the future.

2.3. What is PMKS+?

Due to the reasons described above, a project team at WPI previously attempted to update PMKS as their MQP in 2018-2019. The new version was named PMKS+ and aimed at improving both the user experience and browser compatibility. PMKS+ was created in Javascript using the Angular framework, so PMKS+ is compatible with modern browsers such as Google Chrome and Safari (Angular, 2019; Appikatla, et al., 2019, p. 10). Figure 2.2. below contains an image of the PMKS+ UI.

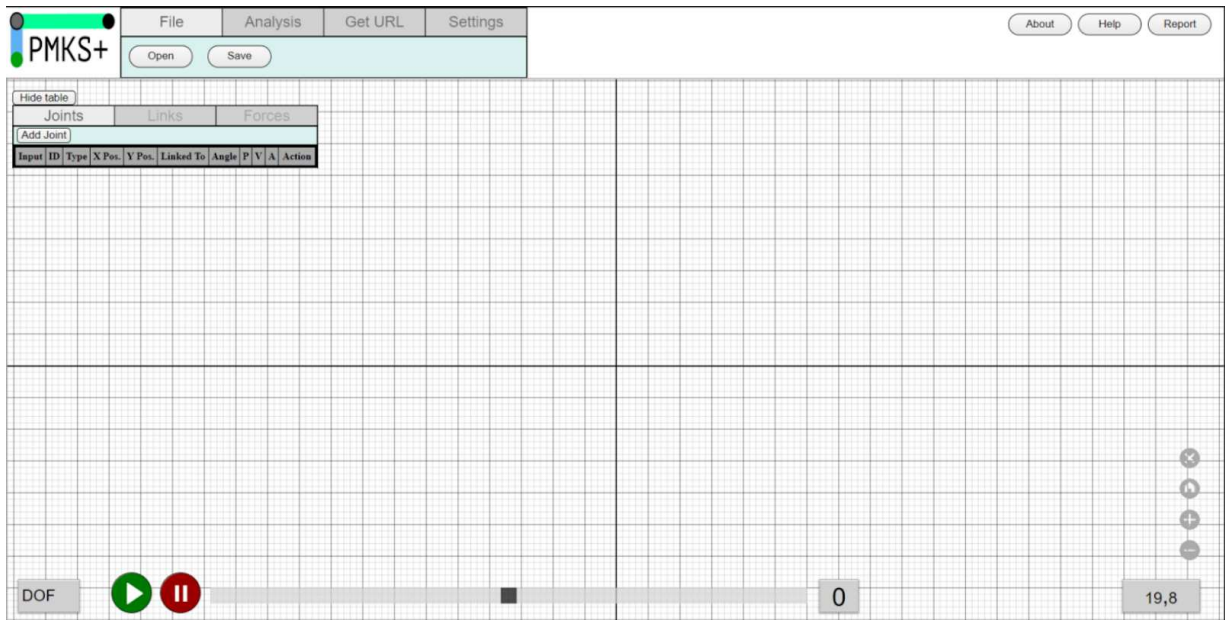


Figure 2.2.: The UI of PMKS+.

The functionality of the original PMKS was preserved in PMKS+. Led by Professor Brown and Professor Radhakrishnan, PMKS+ progressed through several iterations of UI designs and functionality (Appikatla, et al., 2019, p. 10).

Through user feedback on both PMKS and PMKS+, the team made improvements to the UI to increase ease of use and readability of the interface. The PMKS+ application is compatible with Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. Additionally, the program enhances the user experience with mouse interactions, context menus, and toolbars. Both the context menu and the mouse interactions allow the user to manipulate the visual representation of the linkage. Lastly the system architecture was redesigned around a more common web

application design, classified as Model-View-Controller framework (Appikatla, et al., 2019, p. 10).

The PMKS+ team serves as an example to our team in multiple areas. The PMKS+ team made decisions about the UI and the functionality of PMKS+ that our team will explore.

Additionally, they performed user evaluations on the PMKS and PMKS+ UIs. Our team will consider the methods that they used to formulate a new evaluation plan for PMKS+Desktop.

3. Research Statement and Goals

Our primary goal for this project was to convert the PMKS system from its implementation that uses Silverlight to one that uses C# while maintaining functionality, so that PMKS is compatible with modern Windows desktops as a standalone application. Secondly, the goal is to use the UI design from PMKS+ to create a new UI for our conversion, and then improve on the conversion by obtaining user feedback (Appikatla, et al., 2019).

In summary, our goals are to:

- Convert PMKS from a web-based Silverlight application to a C#-based, Windows Presentation Foundation desktop application.
- Implement the functionality of PMKS Silverlight using C#.
- Replicate the UI of PMKS+ and make necessary adjustments for a desktop environment.
- Conduct user evaluations and improve the UI and UX from collected user feedback.

The UI of the PMKS desktop version needs to comply with widely-accepted standards of software development. The UI should contain a well-structured layout that is simple, consistent, and follows the rules of UI design, such as the law of proximity and law of closure (Stone, et al., 2005). Additionally, metrics such as speed of performance, rate of errors, essential efficiency, and task visibility will be measured in order to evaluate the quality of the UI and UX of the desktop version (Stone, et al., 2005; Constantine & Lockwood, 1999).

The team will also conduct research on other desktop manufacturing engineering applications for aspects of their UI and UX that contribute to quality desktop software.

Finally, the team will formulate questions and procedures for the user evaluations so we can appropriately test the UI and obtain relevant feedback. We will then evaluate our collected data and compare it to the evaluation data collected by the PMKS+ team. This will allow our team to determine whether or not the desktop version has improved on the UI and UX of PMKS+, and determine how it might be improved further.

4. Methodology

Throughout the project, our team took various actions in order to achieve the goals described in Chapter 3: Research Statement and Goals. This chapter details the steps our team took in chronological order, as well as our justification for taking each step in the context of fulfilling the project's goals. The following chapters will provide more detail about these subjects.

4.1. Understanding the Project

Our team first decided that we needed to thoroughly understand the project before working on the conversion.

4.1.1. Understanding Linkages

The first subject our team found necessary to understand was linkages. PMKS, as previously explained, stands for “Planar Mechanism Kinematic Simulator”. Planar mechanisms, a.k.a. planar linkages, are a subset of the type of mechanical assembly known as a linkage. Our team decided that it was important to understand linkages because the real-world importance of linkages reflects the importance of software such as PMKS and PMKS+ to assist people studying linkages. We also needed to understand how linkages behave and how linkages are used in reality. This enabled us to understand how linkages should be represented within the simulation to best suit the users' needs and expectations.

To begin, Professor Radhakrishnan referred us to literature that explained what linkages are, how they are used, and why they are important. Then, to help us understand how linkages move, Professor Radhakrishnan provided us with a physical representation of a linkage and a TXT file that could be imported into PMKS to simulate an equivalent linkage. Figure 4.1. below contains an image of the physical linkage and the equivalent representation within PMKS.

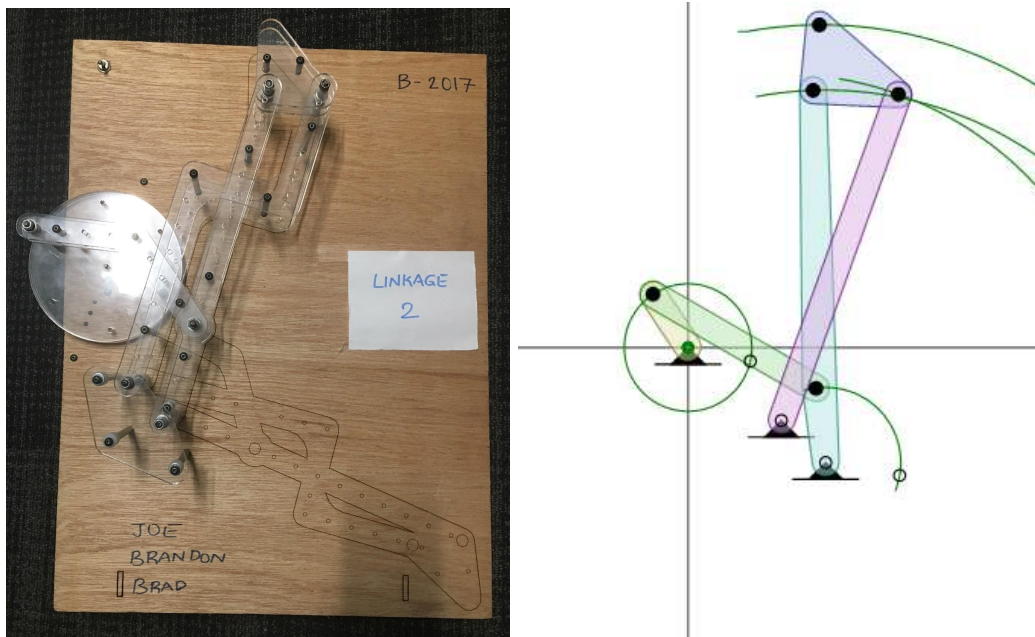


Figure 4.1.: Linkage 2 as a physical linkage (left) and simulated in PMKS (right).

More detail on this subject can be found in Chapter 2.1.: What are Linkages?

4.1.2. Understanding PMKS and PMKS+

Our team then decided that it was necessary to understand PMKS and PMKS+. Through literature research, code explorations, and discussions with our advisors, we determined the development history, application format, features, and limitations of both applications. This information allowed the team to gain a better understanding of the benefits of these applications and the need for a desktop alternative.

More information about the development history and application format of PMKS and PMKS+ can be found in Chapter 2.2.: What is PMKS? and Chapter 2.3.: What is PMKS+?. More information about the limitations of PMKS and PMKS+ can be found in Chapter 6.: Desktop Application Justification. Examples of important functionality of PMKS can be found in Chapter 7.: WPF Coding Explorations.

4.1.3. Understanding the Requirements

The team's next step was to create a set of requirements for the MQP so we could ultimately determine how successful the project was. To do this, the team created a tentative list of requirements and tasks that needed to be fulfilled to consider the MQP complete. This list was then provided to the professors, who gave the team feedback. The feedback was used to revise the list, and the revised list was approved by both parties.

With the set of requirements decided, the team decided to create a tentative schedule so that we would have a plan about how to complete our goals within the allotted time. This

schedule was shared with the advisors and revised in the same way. Once the schedule was agreed upon, the team was able to begin working on the new application.

More information regarding the goals of the project are in Chapter 3.: Research Statement and Goals.

4.2. Desktop Application Justification

The next subject our team explored were the potential formats for our desktop conversion. Previously, our team established the need for a desktop alternative to PMKS and PMKS+. Our next step was to decide on the operating systems on which our application would be supported. We performed research on the US market share of operating systems and the usage of operating systems on the WPI campus. Microsoft Windows was the most used by a wide margin, but MacOS was also found to be a potentially significant user population. However, a Windows-specific application could still be used by Mac users through WPI's remote access servers.

After our team decided that Microsoft Windows was our target operating system, we then evaluated the four application formats that Microsoft supports on Windows: Win32 API, Windows Forms, WPF, and UWP. By comparing the positive and negative attributes of each format in comparison to the requirements of the conversion and our goals for the project, our team decided that the conversion would be a WPF application.

More information on this topic can be found in Chapter 6: Desktop Application Justification.

4.3. Literature and Application Research

Next, we began research into the scholarly literature concerning UI design, evaluation, and user testing so that we could better understand the standards of professionals within these fields. Two requirements of our project are to develop a user interface for our application and to evaluate the application using metrics and user testing, so it is important that our procedures are informed by these scholarly resources.

More information on this topic can be found in Chapter 5.: Literature Review.

4.4. Exploring WPF

While performing literature research, our team also explored the WPF format to gain an understanding of the tools we would be using during the conversion process and explore different methods of implementing PMKS's functionality in WPF.

Through Microsoft's Visual Studio 2015, the team created a branch of PMKS's GitHub repository and created an empty WPF project to store the code we would then create (GitHub, 2020). We first learned how to code in the C# programming language by completing an introductory tutorial by Guru99 (Guru99, 2019).

Once we were familiar with C#, our team then began investigating ways to replicate PMKS functionality, such as scrolling through a coordinate plane using the mouse, displaying a rotating link, and changing the coordinates of the link, within a WPF application. Coding these features without using the PMKS application as inspiration allowed the team to explore solutions

to the conversion problem that did not depend on the former structure and potential uses of Silverlight

More information about this topic can be found in Chapter 7: WPF Coding Explorations.

4.5. Creating PMKS+Desktop

Once finished with exploring WPF, our team began the process of creating our conversion. We decided on the name “PMKS+Desktop” and then created a new GitHub branch to store the code.

4.5.1. C# Documentation and Naming Conventions

One subject that our team was required to understand for the conversion is the documentation and naming conventions for C#. We read Microsoft’s resources on naming and commenting practices. Therefore, we would be able to create properly formatted, readable code with appropriate, explanatory documentation the first time.

4.5.2. Reformatting the WPF Coding Explorations

The next step we took was to transfer the code we had created to replicate PMKS functionality to the new GitHub branch. However, we needed to reformat the code as we transferred it to fit the C# documentation and naming conventions that we discovered within the previous step. This improved the readability and structure of the code and provided us with an opportunity to discover bugs that we might have missed and add appropriate comments if they were missing.

4.5.3. Creating a New User Interface

After reformatting our existing code, our team began working on the new UI for the conversion. Our objective for the new UI was to incorporate the new features and layout developed by the PMKS+ Web MQP team. Several visual mockups of the UI were developed using Microsoft Visual Studio and Microsoft Blend as XAML files, allowing the team to view potential designs and gain approval from our advisors regarding certain decisions. Once the mockup reached its finalized and approved state, the team was able to use the mockup's XAML file as the foundation for the UI of the conversion. Then, the previously developed functionality was reformatted and linked to the new XAML file.

More information on this subject can be found in Chapter 8.: PMKS+ Interface and Chapter 9.: Interface Design.

4.5.4. Developing Additional Functionality

Meanwhile, our team was also developing additional functionality to add to the conversion. A portion of this functionality consisted of the features that we did not develop during our WPF explorations, such as prismatic joints from PMKS Silverlight. Another portion of this functionality included features from PMKS+, such as its new mouse-based linkage creation method. Finally, the team integrated completely new functionality to the application, such as the ability to generate a 3D model of the linkage and export it to SolidWorks.

More information on this subject can be found in Chapter 8.: PMKS+ Interface, Chapter 9.: Interface Design, and Chapter 10.: System Design.

4.6. User Evaluations

A requirement of the project was to hold a user evaluation of the software. The feedback collected from the user evaluation would allow the team to determine if PMKS Desktop's interface and features are well-designed for the intended audience.

4.6.1. Designing the Evaluation

To hold a successful user evaluation, the design of the evaluation needed to be determined. The team discussed what the evaluation was meant to inform us of so that we could then design an evaluation script that would accomplish this goal. The team decided that the evaluation was meant to determine whether college mechanical engineering students were satisfied with PMKS+D's new interface and functionality.

Due to the circumstances, it was necessary to create an evaluation script that the participants could perform remotely and without supervision. Therefore, the team created a Google Forms survey with a list of tasks that the participants would complete. These tasks allowed the user to experience the most important aspects of the interface and functionality. Then, a list of follow-up questions were created so that the team could collect data about the participants' preferences, encountered issues, and recommendations for the application.

To give the participants access to PMKS+D and SolidWorks, the team determined that a ZIP file would be created that contained the required files for PMKS+D and would be distributed

for participants to download on Slack. Then, the students would access one of WPI's Windows remote servers, download and unzip the ZIP file, and complete the evaluation.

An application for this evaluation was submitted to WPI's Institutional Review Board and was approved.

More information about this topic can be found in Chapter 12.2.: Design of User Evaluation Survey.

4.6.2. Designing the Data Analysis Script

With the design of the evaluation complete, the team knew what types of data that the evaluation would provide. The team then needed to establish the process that would be used to examine the collected data. For each kind of question, the team determined the ways that we would organize and analyze the data. For Likert scales, the mode and median responses would be found. For open response questions, the number of users that mentioned a specific topic would be counted. Then, all questions would be graphed as percentage column charts to determine the shape of the response distribution.

More information on this topic can be found in Chapter 12.3.: Design of Analysis Procedure.

4.6.3. Performing the Evaluation

With the evaluation and analysis designs finalized, the team then performed the evaluation. Professor Radhakrishnan recruited students from his ME 4320 Advanced Engineering Design course to participate in the evaluation. He then added the participants to a

Slack channel and provided the ZIP file, the link to the Google Form, and instructions (Slack, 2020). The participants were required to complete the evaluation before 11:59 PM on April 28, 2020, at which point responses to the survey were closed. Throughout the evaluation process, the team was available to assist students with issues that prevented them from completing the evaluation.

4.6.4. Performing the Analysis

Once all responses were received, the team began performing analysis on the data according to the analysis plan. First, the team found the mode and median of the Likert scale question. Then, the team counted the number of users that mentioned each topic for each open response question. Then, the team created a graph for each question to plot the percentage of participants that gave a certain response. Once this was complete, the team analyzed the processed data and drew conclusions about how satisfying PMKS+D's interface and functionality were to our participants.

More information about this topic can be found in Chapter 13.: Data Analysis.

5. Literature Review

In order to begin work on creating a user interface the team needed to conduct research into user interfaces. Important fields the team looked into were UI design, UI evaluation, user evaluation design, and other desktop engineering applications. In-depth research into these areas provided the team with a solid knowledge base for application design.

5.1. UI Design

User interface design plays an important role in creating a usable desktop application. By understanding the key components of professional UI design, the team could follow guidelines to aid in the creation of the application.

5.1.1. Usage-Centered Design

The team looked at two techniques in designing a UI, usage-centered design and user-centered design. The comparison in Table 5.1. below between the two by Constantine and Lockwood (2002) is a good representation of the difference between the two concepts.

Table 5.1.: User-centered design vs. usage-centered design.

User-Centered Design	Usage-Centered Design
<ul style="list-style-type: none"> ● Focus is on users: user experience and user satisfaction ● Driven by user input ● Substantial user involvement <ul style="list-style-type: none"> ○ User studies ○ Participatory design ○ User feedback ○ User testing ● Design by iterative prototyping ● Highly varied, informal, or unspecified processes ● Design by trial-and-error, evolution 	<ul style="list-style-type: none"> ● Focus is on usage: improved tools supporting task accomplishment ● Driven by models and modeling ● Selective user involvement <ul style="list-style-type: none"> ○ Explorative modeling ○ Model validation ○ Usability inspections ● Design by modeling ● Systematic, fully specified process ● Design by engineering

The team used both of these design concepts to iteratively create and improve the UI design.

The first process of UI design is usage-centered design (Constantine & Lockwood, 2002). This design process is based on the tools users interact with. By focusing on the tools required to complete a task, usage-centered design can improve task accomplishment rates. In order to increase efficiency, models of the tasks are created and analyzed. This is a systematic process that requires minimal input from users and accurate representation of tasks and user roles through models.

In order to create accurate models the iterative modeling process itself must be understood. The first step in model creation is the creation of components of the model (Constantine & Lockwood, 2002). Whether the components are the different tasks needed, or the

different user roles, it is important to begin creatively to produce a large volume of components of the model. The next step is to organize the components of the model by understanding the relationship between each part of the model. This may mean the action the user needs to take to proceed to the next part of a task. Finally once an organized model is created, the model must be refined and detailed to ensure an in-depth, accurate model. These processes are repeated constantly throughout all stages of the design process and provide a map of the benefits and drawbacks of every version of the user interface.

The modeling process is a creative process that can get complicated and unhelpful quickly. To avoid this, there are also a few guidelines to follow when creating models. Ensuring that the creative process is separated from criticisms will allow rudimentary progress to be made in a short amount of time (Constantine & Lockwood, 2002). Criticism can occur in the third step of the modeling process, refinement. Similarly a general model outline can be created and finer details can be added later. However, the opposite approach, where specific details aid in the creation of general models, can also apply. The final guideline for creating models is to not immediately connect new concepts to the model. Rather, time should be allowed for reflection between the creation and connection of model parts to the model. By following these rules the team can create accurate and helpful models.

Task Model:

One type of model used during UI design is a task model (Constantine & Lockwood, 2002). Task models are a map to the processes and steps a user needs to take to perform a certain task. Task models aid in breaking down a large task into its individual components, making it

easier for a designer to isolate individual steps. These individual steps will directly correlate to the potential user actions that must be programmed to allow a user to progress through a task in a UI. Task models will allow the team to analyze the different options and determine if there are more optimal solutions to task procedures.

5.1.2. User-Centered Design

The second process of UI design is user centered design. User centered design functions under the cycle of user feedback generating design choices and the reevaluation of the new design. The first component is to gather user data including discrete numbers, preferences, and comments. There are several different methods of gathering data from users to guide the design of a UI. Table 5.2. below displays the different techniques used in the user-centered design process (Abrams, et al., 2004).

Table 5.2.: Techniques used in the user-centered design process.

Technique	Purpose	Stage of the Design Cycle
Background Interviews and questions	Collecting data related to the needs and expectations of users; evaluation of design alternatives, prototypes and the final artifact	At the beginning of the design project
Sequence of work interviews and questionnaires	Collecting data related to the sequence of work to be performed with the artifact	Early in the design cycle
Focus groups	Include a wide range of stakeholders to discuss issues and requirements	Early in the design cycle
On-site observation	Collecting information concerning the environment in which the artifact will be used	Early in the design cycle
Role Playing, walkthroughs, and simulations	Evaluation of alternative designs and gaining additional information about user needs and expectations; prototype evaluation	Early and mid-point in the design cycle
Usability testing	Collecting qualitative data related to measurable usability criteria	Final stage of the design cycle
Interviews and questionnaires	Collecting qualitative data related to user satisfaction with the artifact	Final stage of the design cycle

The team decided to conduct focus groups, walkthroughs, and usability testing. However due to circumstances the team had to cancel in person walkthroughs. The team did conduct focus groups and walkthroughs.

5.1.3. UI Design Rules

UI design rules are important guidelines for creating a cohesive user interface. These rules include **Access**, **Progression**, **Support**, and **Context** (Constantine & Lockwood, 1999).

The first rule is **Access**. It is important for the user to have access to tools that pertain to them. Since users can be a range of skill levels, the UI should present initially simple features but contain other more advanced features that are also easily accessible.

Progression is the second usability rule to follow when creating a user interface. As a user gains experience with the user interface, initial or beginner level tasks become second nature to the user. Progression allows the user to continue to develop their user experience by adding advanced options to the interface.

Additionally, **Support** will be needed for engaged users. Support provides the users with a road map of how to progress through the different tasks. Sections of the program should be dedicated to helping the user navigate the user interface because users often have a low long term memory retention.

Context is the final major rule for greater usability development. This rule recommends the separation of information throughout the UI. By partitioning components of the user interface, the user can infer the relation of certain functions and can reinforce certain activities for a better UX. Furthermore, sectioning the UI allows the development team to implement a fairly large level of complexity without impeding on other design rules and usability metrics.

The team used these rules to help guide the development of the user interface. By designing the application following the rules of access, progression, support, and context, the team optimized our ability to implement an intuitive user interface.

5.2. Metrics for UI Evaluation

For our team to determine whether or not we have successfully created a good UI, it is necessary to decide on how we will evaluate our UI. Evaluating a UI is important in the development process because it allows you to measure how close you are to achieving a set of established requirements for the project. One method to evaluate a UI is to compare it to a number of performance metrics. These metrics are decided on beforehand and goal values are established so the team can measure their progress as the UI is developed.

5.2.1. The Most Common Metrics

The “five Es” are five dimensions that can be used to evaluate the usability of an interface. The five Es are: Effective, Efficient, Engaging, Error tolerant, and Easy to learn (Stone, et al., 2005, p. 108).

If an interface is Effective, users can achieve their goals completely and accurately. An Efficient interface allows users to accurately complete a task quickly. Engaging interfaces have a style and tone that pleases or satisfies the user. Error tolerant interfaces prevent errors and helps the user quickly recover from errors that occur. Finally, Easy to learn interfaces are designed so

that new users can understand the basic features quickly and users can intuitively learn more complex functionality (Stone, et al., 2005, p. 108).

Each of the five Es can have both quantitative and qualitative data collected about them during user evaluations. The effectiveness can be measured quantitatively by determining whether the user completed a task accurately, and can be measured qualitatively by asking the user whether they felt the task was finished correctly (Stone, et al., 2005, p. 445). Table 5.3. below describes quantitative and qualitative data that can be collected for each of the five Es, as suggested by Stone et al. (2005).

Table 5.3.: Potential quantitative and qualitative data for each usability dimension (Stone, et al., 2005, p. 445).

Dimension:	Possible Quantitative Data:	Possible Qualitative Data:
Effective	Whether the task was completed accurately or not	The users' views of whether the task was completed accurately or not
Efficient	<ul style="list-style-type: none"> ● Number of clicks/keystrokes ● Elapsed time 	The users' views on the difficulty of the task
Engaging	Numeric measures of satisfaction (such as Likert scales)	Qualitative surveys/interviews to gauge user attitudes towards the interface
Error tolerant	Level of accuracy achieved versus time spent making errors	User reports of confidence in the interface even after making mistakes
Easy to learn	<ul style="list-style-type: none"> ● Number of errors ● Time spent in incorrect routes ● Time spent by a novice versus time spent by an experienced user 	Novice users' feelings of confidence in using the interface

5.2.2. Additional Metrics

According to Constantine and Lockwood (1999), there exist other potential metrics to measure the quality of a UI. The three that our team identified as important are: essential efficiency (EE), task visibility (TV), and visual coherence (VC).

EE measures the ratio of the ideal number of steps that need to be taken to complete a task to the actual number of steps needed (Constantine & Lockwood, 1999, p. 427). An EE of 100% means that a task takes exactly the ideal number of steps. In order to accurately measure EE, it is important that the use cases of the application be as simple and general as possible so the minimum number of essential steps is found (Constantine & Lockwood, 1999, p. 429). This is especially important for short, frequent tasks because they need to be straight-forward and take no more actions than required (Constantine & Lockwood, 1999, p. 427). The formula below calculates EE for a certain task, where $S_{essential}$ is ideal number of steps to complete a task and $S_{enacted}$ is actual number of steps required.

$$EE = 100 \times \frac{S_{essential}}{S_{enacted}}$$

TV measures the percentage of a task that is visible to the user (Constantine & Lockwood, 1999, p. 433). Each step in the completion of a task is ranked between 0 and 1, with 0 signifying that the UI feature required to complete the step is extremely difficult to find and 1 signifying that the UI feature is immediately obvious (Constantine & Lockwood, 1999, p. 434). It is important that TV be high for frequent use cases, especially when certain steps are not

optional (Constantine & Lockwood, 1999, p. 433). The formula below calculates TV for a certain task, where S_{total} is the total number of steps taken to complete a task and V_i is the feature visibility (between 0 and 1) of step i .

$$TV = \frac{100}{S_{total}} \times \sum_{\forall i} V_i$$

VC determines how well a UI keeps related features together and unrelated features separate (Constantine & Lockwood, 1999, p. 438). To calculate total VC, the VC of the UI components within each subgroup can be calculated, and recursively evaluate higher levels of UI groups until you evaluate the entire UI. In practise, the relatedness of two features can be simplified to two values: 0 or 1 (Constantine & Lockwood, 1999, p. 440). A value of 1 means that the two features are substantially related; otherwise, the value is 0. VC for a group of components can be calculated using the following formula, where N_k is the number of visual components in group k , and $R_{i,j}$ is the relatedness of two components i and j .

$$VC = \frac{100 \sum_{\forall k} G_k}{\sum_{\forall k} N_k \times (N_k - 1) / 2} \text{ with } G_k = \sum_{\forall i,j | i \neq j} R_{i,j}$$

5.3. Heuristic UI Evaluation

Another form of UI evaluation that the team used was heuristic evaluation. A heuristic evaluation is a process where a set of evaluators inspect an interface with regards to an established set of usability principles known as “heuristics” (Nielsen, 1994). These evaluators then report aspects of the interface that violate the heuristics.

There are many sets of heuristics that can be used. One set of heuristics was created by Jakob Nielsen (1994) to provide the broadest explanatory coverage of actual usability problems. Table 5.4. below describes each heuristic.

Table 5.4.: Usability heuristics created by Jakob Nielsen (1994).

Heuristic:	Description:
Visibility of system status	Reveal what is happening in the system
Match between system and real world	Use language familiar to the user and ensure real world metaphors are used correctly
User control and freedom	Minimize the extent to which the system traps the user in an inescapable state
Consistency and standards	Keep a consistent visual style and display similar information similarly
Error prevention	Design the system to prevent errors in the first place
Recognition rather than recall	Make the user's future options salient
Flexibility and efficiency of use	Allow the user additional options to sidestep regular interaction techniques
Aesthetic and minimalist design	Satisfy the user aesthetically but do not overwhelm them
Helping users recognize, diagnose, and recover from errors	Provide error messages and help information

Our advisors will perform heuristic evaluations on each iteration of our UI to discover usability problems. The team will then use this feedback to fix the usability problems and develop new iterations of the UI.

5.4. User Evaluation Design

Another method to evaluate the quality of a UI is to have a sample of the user demographic perform an evaluation of the UI through a guided experiment. Rubin and Chisnell (2008) describe three types of user testing: exploratory testing, assessment testing, and validation testing. The most relevant category for this project is assessment testing.

Assessment testing is used after the backbone of the UI has been completed. This form of testing measures the intuitiveness of the UI as well as how well users can perform full-sized, realistic tasks to identify areas of improvement (Rubin & Chisnell, 2008, p. 34–35). Acting as a mid-point between exploratory testing and validation testing, validation testing has users performing tasks under the supervision of a moderator. The moderator collects quantitative data and occasionally asks the user about their behaviors and thought processes (Rubin & Chisnell, 2008, p. 35).

5.4.1. Requirements for an Evaluation

To conduct a user evaluation, various requirements must be fulfilled. Rubin and Chisnell (2008) describe these requirements as components of a test plan.

First of all, the test plan should contain the research questions. These describe the issues that need to be resolved and the foci of the research (Rubin & Chisnell, 2008, p. 69). This section is described as the most important section, and therefore it is important that the questions be “as precise, accurate, clear, and measurable... as possible” (Rubin & Chisnell, 2008, p. 69).

Secondly, the participant characteristics should be detailed. These are the qualities that describe the various users of the product, because these are the kinds of people that should be sampled for the user evaluation. If there are multiple categories of users, e.g. experts and novices, these should be described, as well as how many subjects to be sampled from each category (Rubin & Chisnell, 2008, p. 72).

Next, the methodology should provide a detailed description of how the team is going to carry out the evaluation. It details each part of the test from the arrival of the subjects to their departure in detail (Rubin & Chisnell, 2008, p. 74). The design of the test should be focused on fulfilling the test objectives. For example, it may be important to collect data on users performing multiple actions with similar processes. This is called within-subjects design, and it is recommended to use “counterbalancing”, i.e. randomizing the order of the tasks for each subject, to mitigate the transferred learning between tasks (Rubin & Chisnell, 2008, p. 76). However, if the task order would normally be sequential, then it is more important to simulate an actual use environment than to eliminate learning transfer.

After creating the methodology, a task list should be created. Task lists contain brief descriptions of each task that the user will complete throughout the evaluation (Rubin & Chisnell, 2008, p. 79). These descriptions should include the materials required by or provided to the user, the screens or program states that the user will be taken through, and a description of successful completion of the task (Rubin & Chisnell, 2008, p. 79-80). Successful completion does not simply mean finishing a list of instructions. You may wish to have users complete the task within a certain amount of time or limit the number of errors they can make.

Testing time is always limited, so it is important to ensure that the task list contains the most important tasks. Rubin and Chisnell (2008) suggest four different methods of task prioritization: *frequency*, *criticality*, *vulnerability*, and *readiness*. Frequency prioritizes the tasks that are performed most often by the majority of the user population (Rubin & Chisnell, 2008, p. 85). Criticality prioritizes tasks that have significant consequences for the end user if performed incorrectly, whereas vulnerability prioritizes tasks that are believed to be the hardest to perform or have design flaws (Rubin & Chisnell, 2008, p. 86). Finally, readiness prioritizes the tasks that are ready to be tested. Usually, reserved as a last resort, the philosophy of readiness is that “it is always better to test something than nothing” (Rubin & Chisnell, 2008, p. 86-87).

Then, the test environment describes the environment that the test is seeking to emulate. If the application is expected to be used in noisy, crowded sales offices, then the test environment should have elements that simulate that environment, such as phones ringing in the background (Rubin & Chisnell, 2008, p. 87). The description also includes any equipment that the subjects will be using, such as computers or printers.

The test moderator role describes what the test moderator will be doing during the experiment. This is especially important if the moderator will perform confusing actions (Rubin & Chisnell, 2008, p. 87). For this reason, it should detail when or why the moderator will probe, and how they will interact with the subject. For example, the moderator might be acting in a particular role or playing devil’s advocate (Rubin & Chisnell, 2008, p. 88).

Finally, a description of the data to be collected during the test is included. Both performance and preference data needs to be detailed, and should be focused on answering the research questions (Rubin & Chisnell, 2008, p. 88). Whether the data will be used quantitatively or qualitatively is

important because that will affect how the data will be collected during the evaluation (Rubin & Chisnell, 2008, p. 88).

5.4.2. Moderating the Evaluation

A significant aspect of user evaluation is the moderation. It is important for moderators to keep their preconceptions in check and perform correctly, as they have the ability to misinterpret what they see the user doing and to affect how the user performs (Rubin & Chisnell, 2008, p. 202).

Firstly, it is key that moderators act impartially to ensure that the user's opinion is not affected by what the moderator seems to think. Moderators should present the product neutrally, as if they have no vested interest whether the product is good or not (Rubin & Chisnell, 2008, p. 202). They should react to the subject's "mistakes" exactly the same as "correct" actions, and react identically to all participants and comments. Never make the subjects feel wrong for making a "mistake"; if the user is having problems, that is the fault of the product (Rubin & Chisnell, 2008, p. 203). This will encourage users to act freely and not be concerned with "looking good". Moderators must also be aware to keep their voice and body language in mind. Approaching someone or raising your tone indicates approval, while moving away from someone or lowering your tone can indicate rejection (Rubin & Chisnell, 2008, p. 203). Evaluation is important, so be serious and respect the user's time. However, you should also be relaxed. Humor is okay when used to relax the subject, but keep in mind to laugh with the subject, not at them (Rubin & Chisnell, 2008, p. 209). Lastly, treat each participant as a new person. Everyone will have different skill sets, assumptions, and perspectives, so it is important

not to let the previous subject influence how you treat the next. For this reason, the moderator should take breaks between sessions to clear their mind (Rubin & Chisnell, 2008, p. 204).

Secondly, it is important to interact with the subject as appropriate to the testing context. If the testing is later in development, probing for the thought processes of the user may be less important. However, interacting with the user correctly and at the right time is always important (Rubin & Chisnell, 2008, p. 206). A moderator should limit their interruptions to short, noninvasive discussions, as probing too frequently will disturb the user's thought process or annoy them (Rubin & Chisnell, 2008, p. 208). Never show surprise at unexpected actions by the user; instead, focus on learning what the *user* expected to happen and whether what just occurred matched with that expectation (Rubin & Chisnell, 2008, p. 207). A moderator should help users express their thoughts in a helpful way. Verbal and nonverbal cues will often provide a good opportunity for the moderator to probe the user's thoughts. If the subject seems lost or troubled, ask *if* there is a problem, but do not betray what you think the problem is. Subtle questioning allows users to answer honestly, without feeling judged or as if there is a correct answer (Rubin & Chisnell, 2008, p. 207-208).

Thirdly, moderators should not "rescue" struggling participants (Rubin & Chisnell, 2008, p. 209). Noting how users feel and act when they are struggling is just as important as when they are succeeding. Moderators should assist only as a last resort, such as when the user is obviously frustrated or about to give up, when bugs are preventing correct progression, or if there is information missing that would normally be there (Rubin & Chisnell, 2008, p. 211-212). If the issue is either of the former two examples, try to gradually reveal information over time to allow the user to figure out the solution, rather than all at once (Rubin & Chisnell, 2008, p. 213).

Finally, a moderator should adapt the test plan appropriately. If you make a mistake, such as revealing information at the wrong time, proceed as if nothing had occurred (Rubin & Chisnell, 2008, p. 210). At worst, you will only invalidate the one section you were on, while at best the user will not notice the mistake. If the moderator is instructing the user, they should make sure that the user feels finished before moving on (Rubin & Chisnell, 2008, p. 211). Once again, the moderator should not signal to the user whether or not they have “succeeded”. It is helpful to instruct the user to signal when they feel they are finished so the moderator can proceed or take notes without it seeming out of the ordinary to the participant.

Do not be afraid to deviate from the test plan (Rubin & Chisnell, 2008, p. 226). If, after multiple participants, the moderator notices that participants generally don’t understand a task, the moderator should provide more information in the future. If you discover additional, important areas of exploration that you missed on the original test plan, or if the original tasks aren’t exploring the research questions correctly, you should adapt the plan during the evaluation rather than miss your chance to collect this data (Rubin & Chisnell, 2008, p. 226). If the participant does not have the experience that you expected, or they are taking more time than allocated on an important task, then it is up to the moderator to provide additional information or extend the timeline. The moderator is there to guide the subjects and obtain the correct types of information to successfully answer the research questions, not to strictly adhere to the test plan.

5.4.3. Data Collection

Before performing an experiment, it is important that the method for collecting and compiling data is established. The format that you store data in should allow you to notice patterns which inform you on some aspect of the research questions (Rubin & Chisnell, 2008, p. 247).

Data compilation should be a continuous process throughout the test sessions to save time during analysis (Rubin & Chisnell, 2008, p. 247). After each day of testing, all results for that day should be collected. This allows the testers to understand what was collected on each day, ensure all data is legible and in the correct format, and determine if important information is missing (Rubin & Chisnell, 2008, p. 247). Afterwards, it may be appropriate to update the test plan.

Written notes should be transferred into a digital format, and all quantitative data should be placed into a master sheet. Spreadsheet software is useful for organizing both quantitative and qualitative data (Rubin & Chisnell, 2008, p. 248). Spreadsheets can automatically update the running summaries of the quantitative data, and laying out text within a spreadsheet will allow the team to see related responses and patterns (Rubin & Chisnell, 2008, p. 248). Using formatting rules, such as key words, to color related cells will allow you to see other patterns you might have missed otherwise.

5.4.4. Data Analysis

To derive conclusions from the data, it must first be analyzed. The first step of analysis is to perform a summarization of the data. After all data has been collected and compiled, summarization allows the team to determine general trends within the data, such as differences in performance between different user groups (Rubin & Chisnell, 2008, p. 249).

Metrics such as speed of performance and rate of errors can be calculated for each task, keeping in mind the definition of “error” and the success criteria that the team agreed on. The team can also summarize the number of errors per type, such as omission (leaving something out) or commission (doing something unneeded) (Rubin & Chisnell, 2008, p. 249). It may also be helpful to generate statistics with different success criteria, such as the percentage of participants who performed a task successfully, including or excluding those who required assistance, and tracking the percentage of participants who performed successfully within a specified time benchmark (Rubin & Chisnell, 2008, p. 250). Quantitative data such as these should have their mean, median, range, and standard deviation calculated (Rubin & Chisnell, 2008, p. 251-253).

Preference data from users should be summarized differently depending on the form of the response (Rubin & Chisnell, 2008, p. 254). Limited-choice responses should have the answers to each question summed. Free-form responses and comments should have each question and their responses listed, with similar answers grouped together. Comments and observations from testing should be transcribed, and critical quotations should be compiled (Rubin & Chisnell, 2008, p. 254).

Other examples of quantitative data to summarize include the number of times returning to the main window or site map unnecessarily and the number and type of hints given for each task (Rubin & Chisnell, 2008, p. 256). The places in tasks where users had issues, and how long they needed to figure out the solution can be noted as well. If there are different kinds of users participating in the project, data for each group should be summarized separately to note differences between them (Rubin & Chisnell, 2008, p. 256). However, the team should be careful using percentages to describe small sample sizes, as they can sound misleading; 100% of 4 people is less impressive than 85% of 2000 (Rubin & Chisnell, 2008, p. 258).

After summarizing the data, the next step of analysis is to determine the tasks where users had the most trouble (Rubin & Chisnell, 2008, p. 258). First of all, focus on tasks that did not meet the success criteria. A 70% success rate is typical; any less and the task should be considered a “difficult” or “problematic” task (Rubin & Chisnell, 2008, p. 258). After determining which tasks are the most problematic, it is useful to divide the tasks into different levels of performance (Rubin & Chisnell, 2008, p. 259).

The team should then identify the errors that lead to incorrect performance for each task (Rubin & Chisnell, 2008, p. 260). For example, the user was supposed to enter certain information before continuing, but they did not. Afterwards, the team should perform a source of error analysis. You should think about how the users’ errors resulted from the design of the product, and then decide on the reason or reasons for these errors (Rubin & Chisnell, 2008, p. 260). Perhaps the user was supposed to enter 20 characters into a field, but the field was only large enough to display 11, or perhaps the navigation between submenus was too confusing and the labels were too vague (Rubin & Chisnell, 2008, p. 260). Often, there are multiple causes for

an error of differing severity. Therefore, the team should be careful about making conclusions before reviewing all the data and considering all potential error sources (Rubin & Chisnell, 2008, p. 261). If time allows, a source of error analysis is recommended to be performed separately for every task performed by every user (Rubin & Chisnell, 2008, p. 261).

Once the source of error analysis is complete, the discovered problems should be prioritized by criticality (Rubin & Chisnell, 2008, p. 261). For smaller tests, it is easier to simply ask the subjects what they felt the biggest problems were, but criticality can still reinforce the user's feelings (Rubin & Chisnell, 2008, p. 263). Criticality is a measure that combines the severity of the problem and probability that the problem occurs. To calculate criticality, all problems should first be categorized on a 4-point scale by severity, i.e. from making the program unusable to only being slightly irritating but having no effect on task completion (Rubin & Chisnell, 2008, p. 262). Then, the problems should be categorized on a 4-point scale by an estimate of their frequency of occurrence (Rubin & Chisnell, 2008, p. 263). A problem's frequency score should take into account the percentage of users that can be affected by the problem, and the probability that someone within that group will experience the problem. Finally, add the two rankings together to determine the criticalities.

5.5. Other Linkage Simulation Applications

The final stage of our review was to investigate other applications used to simulate or design planar linkages. The 5 applications investigated were:

- MechAnalyzer
- Linkage Mechanism Designer and Simulator (LMDS)
- SAM
- Working Model
- MotionGen

These applications were chosen by Professor Radhakrishnan for having a similar purpose to PMKS.

The purpose of the investigation was threefold: to assess how often certain features appeared in linkage applications, how these features worked, and to determine positive and negative aspects of their UI designs. This data provided the team with information about existing linkage simulation solutions that we would use to develop PMKS+D.

The team began the investigation with a list of features that either existed in PMKS already or were being considered for PMKS+D. For each application, the presence of each feature was noted. If a feature was present, how the feature functions was noted as well. Finally, features or aspects of the UI that the team liked or disliked as a new user were noted. Appendix A: Notes on Linkage Simulation Applications contains the notes taken during the investigation.

All of the features that the team looked for were present in at least one of the applications. This demonstrated to the team that the features we were considering for PMKS+D

have precedent in other similar applications. The data allowed us to determine which features were common, i.e., supported by 3 or more applications, and which were uncommon. The team then decided to prioritize the more common features over the uncommon ones because their prevalence suggested that they would be useful to a linkage designer in more situations.

The team used how these features functioned in the different applications as inspiration for potential ways we could implement these features in PMKS+D. For example, 3 out of 5 applications allowed the user to toggle the display of the movement paths for each joint by right-clicking the joint.

Finally, the team used the positive and negative aspects of these applications to influence the design of our user interface. The positive aspects were considered for our application, such as the informative hover text used in SAM and Working Model. The negative aspects were remembered and avoided to the best of our ability, such as the lack of a tutorial and the overly complex submenus within Working Model and LMDS.

6. Desktop Application Justification

PMKS+ offers a modernized, Javascript-based version of the original PMKS application. However, there are multiple benefits that having a desktop version of PMKS could offer in addition to the web-based versions. To develop a desktop version, a choice must be made between the application formats available. This chapter explores the decisions our team made that resulted in us deciding to convert PMKS to a standalone, desktop, Windows-compatible Windows Presentation Foundation application.

6.1. Why PMKS Needs to Be Replaced

As stated in Chapter 2.: What is PMKS and PMKS+?, the original PMKS application was created with a development library called Microsoft Silverlight. Silverlight is currently strictly compatible with Internet Explorer 10 and 11, meaning that PMKS can only be run on Windows machines, and that users cannot use more modern, more advanced browsers (Microsoft Corporation, 2011). Additionally, a Windows-exclusive web-based application such as PMKS has the accessibility limitations of a Windows-exclusive desktop application without the positives that a desktop application has, which will be explored in the next section. In addition to these restrictions, support for Silverlight is ending in October, 2021, which will make PMKS unusable (Microsoft Corporation, 2011). Therefore, PMKS has inherent, unfixable flaws and needs to be replaced.

6.2. Why PMKS+ Needs a Desktop Alternative

PMKS+ successfully corrected many of the issues the original application had. It is compatible with modern PCs, and has a more modern UI and a new mouse-based linkage creation method. However, PMKS+ also faces multiple shortcomings that make it a non-ideal version of PMKS.

6.2.1. Ease of Accessibility

Similar to the original PMKS application, PMKS+ is dependent on a modern development tool called Angular CLI. Angular CLI depends on Node.js, an asynchronous event-driven JavaScript runtime (Node.js Foundation, 2019a). Node.js is currently compatible with Windows, Mac OS, and Linux, and Angular CLI is compatible with all major browsers, including Google Chrome, Safari, and Firefox (Angular, 2019; Node.js Foundation, 2019b). Therefore, PMKS+ has far greater accessibility than PMKS. However, Node.js, Angular CLI, and a web browser are packaged individually and each requires its own installation. For new users, this could be a major source of confusion. A standalone desktop application that requires fewer installation steps and contains all necessary packages would be more accessible to new users.

6.2.2. Incomplete Functionality

The PMKS+ team had to rebuild PMKS in Javascript from the ground up. This took a large amount of work, and the team was not able to totally complete the conversion, leaving some functionality of PMKS unimplemented. Users are not able to perform static, dynamic, and stress analysis on the linkage, simulate forces, or save and share linkages (Appikatla, et al., 2019, p. 115–116). The unimplemented non-dyadic solver, which allows the simulator to track the position of joints that are not attached to the ground or the input joint, was especially important because certain valid linkages are unable to be solved currently (Appikatla, et al., 2019, p. 115–116). Other functionality, like collision detection between links and an interactive tutorial, were suggested as potential improvements in the future (Appikatla, et al., 2019, p. 118–119).

6.2.3. Integration with Other Desktop Applications

Users of PMKS at WPI must also work closely with other engineering software. SolidWorks is a computer-aided design (CAD) system that is used by engineers at WPI, and Professor Radhakrishnan plans to implement some of its functionality into PMKS in the future. Integration with SolidWorks or AutoCAD is currently not possible with a web-based application like PMKS and PMKS+. However, a desktop application would make this integration possible, allowing users to interact with other software through PMKS.

6.3. Why Windows?

After it was decided that a desktop application would be necessary, the team had to decide which operating systems (OS) would support our desktop application.

According to NetMarketShare (2019a), the three most common OS platforms are Windows at 87.50% of the market, MacOS at 9.74%, and Linux at 2.14%. Evidently, Windows is a supermajority of OS on the market.

At WPI, the majority of computers that WPI buys for the library, classrooms, and teachers are Windows computers. WPI Information Technology Services (2019) recommends that new students purchase Windows or Mac OS computers. Furthermore, the majority of new students purchase Windows machines because they are cost-effective and are compatible with most academic software (WPI Information Technology Services, 2019). An important example of such an academic software system is SolidWorks. At this time, Solid Works is only officially compatible with Windows (Dassault Systèmes, 2019).

For these reasons, a new PMKS desktop conversion must be compatible with Windows. However, if Mac OS is left unsupported, we risk leaving a potentially significant portion of students unable to use the conversion. Thankfully, WPI supports the use of a remote access server to use WPI software, which is also compatible with Mac OS (WPI Information Technology Services, 2019). Therefore, it made the most sense for our team to pursue a Windows-compatible solution without attempting to support multiple platforms.

6.4. Why Windows Presentation Foundation?

After deciding to create a Windows-compatible desktop conversion, our team needed to decide on the type of application to create. Microsoft officially supports four application formats: Win32 API, Windows Forms, Windows Presentation Foundation (WPF), and Universal Windows Platform (UWP) (Schofield, 2019). To determine the format to proceed with, our team discussed the pros and cons of each application format.

6.4.1. Why not Win32 API?

Win32 API is a low-level C++-based framework. Due to allowing unmanaged code to run, Win32 allows for developers to achieve higher levels of performance and efficiency (Schofield, 2019). The other three options we considered use a managed runtime environment like .NET Framework (Schofield, 2019). However, it is not common to use the Win32 API for modern applications, but is more suited for applications where the developers do not want users to see the Windows UI (Schofield, 2019). Our PMKS conversion would likely not utilize the low-level functionality nor benefit much from the increased performance.

Additionally, Win32 API does not support C# or XAML (Schofield, 2019). Due to the fact that PMKS was developed for Silverlight, there exists a significant amount of regular C# and XAML code that could be reused. Choosing a format that is compatible with C# and XAML would allow our team to reuse the relevant code. Recreating the application from scratch with a different programming language would add a significant amount of development time.

For these reasons, we decided that Win32 API would be a significant time investment that is not outweighed by the low-level functionality or increased performance. For these reasons, we decided not to use Win32 API for creating the PMKS desktop conversion.

6.4.2. Why not Windows Forms?

Windows Forms was the original platform developed for Windows to use the .NET Framework. Windows Forms is compatible with Windows 7 and all later versions, and has support for C# development. However, Windows Forms is not compatible with XAML, and focuses on a more “lightweight” UI model. For this reason, we decided not to focus on Windows Forms when WPF and UWP both support C# and XAML (Schofield, 2019).

6.4.3. WPF or UWP?

The last two of Microsoft’s format options remaining were Windows Presentation Foundation (WPF) and Universal Windows Platform (UWP). At first glance, WPF and UWP seem quite similar. WPF and UWP both support C# and XAML, have similar “development skills”, and are designed for desktop applications that “require a sophisticated UI, styles customization, and graphics-intensive scenarios” (Schofield, 2019). However, both formats have specific traits that we had to weigh when deciding on which format to implement.

WPF was Microsoft’s first format to support the use of XAML to provide a “rich and customizable UX”. It is currently supported by Windows 7 and up, and may be deployed standalone through a self-contained executable or pushed to users’ machines from a server through ClickOnce deployment (NextTurn, et al., 2017; Schofield, 2019).

UWP is Microsoft's newest format for Windows applications. UWP has access to many additional features, including accessibility features like screen readers, adaptive controls and input, multi-window display, and the ability to run on Xbox and Windows 10 mobile (Schofield, 2019; Whitney, et al., 2018). However, UWP apps on desktop are only supported on Windows 10, and UWP apps must be distributed through the Windows Store (Schofield, 2019; Whitney, et al., 2018).

Between WPF and UWP, our team decided to focus on WPF moving forward. While UWP is the most recent format, it has two significant drawbacks.

The first is that UWP applications must be distributed through the Windows Store. In order to be allowed onto the store, the UWP app must complete a certification process from Microsoft (JnHs, et al., 2018). WPF apps may be distributed freely as an executable, and do not require a certification process or the use of the Windows Store.

Second and more importantly, UWP only supports Windows 10 desktops, and this is a major detriment. According to NetMarketShare (2019b), Windows 10 makes up nearly 49% of Windows installations, while Windows 7, 8, and 8.1 comprise nearly 48% of Windows installations as well. Not supporting Windows 7, 8, and 8.1 would leave nearly half of Windows users unable to use our conversion, and the additional features of UWP are irrelevant to our conversion and do not outweigh that cost.

For all of the reasons detailed above, our team decided to convert PMKS to a standalone, desktop Microsoft WPF application.

7. WPF Coding Explorations

To begin the process of programming, it was important that the team understand how to code in C# using WPF. We achieved this goal by completing a series of C# tutorials and recreating important functionality of PMKS in a WPF application. The following chapter details these coding explorations and how they relate to the completion of the project.

7.1. Guru99 C# Tutorial

The first exploration the group performed was working through the tutorials by Guru99 (Guru99, 2019). These tutorials were recommended by Professor Radhakrishnan to become more familiar with the WPF format. The group worked through all sections of the tutorial by reviewing the provided information and by following the instructions in a WPF project in Visual Studio. The important tutorials that later became useful are creating basic controls, such as `TextBoxes` and `Buttons`, data binding containing `DataContext`, and responding to changes.

7.2. Window and Display Creation in WPF

The next investigation the group performed was to work on recreating the different aspects of the PMKS UI in WPF. This section will discuss the visible elements the team explored in detail. By experimenting and creating examples of these different components, the team was able to gather more information about WPF UI elements that can be used to create the final UI.

7.2.1. Draw a Coordinate Plane

There are two parts to drawing a coordinate plane in WPF. The first part is creating the tiled background and the second part is creating the x and y axes.

In order to create a grid to efficiently display a coordinate plane, the team had to determine a way to draw both horizontal and vertical lines behind all of the other objects on the grid. The `Background` property is a useful tool for this feature. The `Background` is a property of both the `Canvas` and `Grid` classes, and have certain properties that are desirable for creating coordinate planes.

Hierarchy:

The first benefit of the `Background` is that it doubles as a child of the `Canvas` you are setting it for. Because scaling and translating a specific `Canvas` will apply the transformation to all of the `Children` of the `Canvas`, including the `Background`.

Depth:

The `Background` also allows for objects that are drawn to that `Canvas` to always be displayed on top. This is important because the plane should never be shown in front of the other `Children` of the `Canvas`.

Opacity:

The third and final benefit of the `Background` that we identified is that you are able to set the `Opacity` of the `Background`, which is useful for adding the axes, which we will explain later.

In order to create the axes of the coordinate plane, the team created two `Rectangles`. The x-axis has a height of 0.25 pixels and an undefined width, so the axis would automatically reach the edge of the coordinate plane. The y-axis was similar; it has a width of 0.25 pixels and an undefined height. The two axes initially displayed in front of all other elements of the `Canvas`. To solve this issue, the team discovered the `ZIndex` property.

An important aspect to note is that the `ZIndex`, or the overlap layer, is automatically set to zero. The `ZIndex` is important to consider because the higher its value, the more priority the object receives in being shown to the user. If two elements occupy the same x and y space on a canvas, the object with the higher `ZIndex` will be shown as an object overlapping the other object. By setting the `ZIndex` of the axes to a negative number, we were able to solve the overlapping issues.

The final component to displaying the axis stated previously was to set the `Opacity` of the `Background`. With all of these values explicitly set, the team was able to display a coordinate plane that displayed the x and y axes and did not overlap with other important elements.

7.2.2. Start Program Maximized

In Visual Studio, programs automatically start in a window when run. It is possible for users to manually maximize the window, but Professor Radhakrishnan tasked our team with determining how to start a WPF program maximized. The ability for our team to create a maximized window on start-up is relevant to PMKS+Desktop because PMKS begins maximized, and our team should have the ability to replicate this behavior.

By default, a new WPF application defines a subclass of `Window` named “`MainWindow`”. This constructor is called when the program first launches.

During our research, we discovered the `WindowState` property of the `Window` class, which specifies which of three possible `WindowStates` the window is in: `Restored`, `Minimized`, or `Maximized` (Microsoft Corporation, 2019i). Adding the line

```
this.WindowState = WindowState.Maximized;
```

to the file sets the `WindowState` for the current window to `Maximized`. Therefore, placing this code within the `MainWindow` constructor allows the program to start maximized.

7.2.3. Create Window Subsections

Another important UI design feature we explored was how to create multiple separate window subsections that do not overlap when the window is resized. Creating subsections that behave this way is important because this is how the menus in PMKS, PMKS+, and most other applications perform. This investigation also created further insight into stationary components and movable components.

Our team began by creating separate `Canvases` and `Grids` within the initial `Window`. By doing this, we could separate the coordinate plane from the stationary menus. This allowed the coordinate plane to be transformed without affecting the menu. Additionally, the menu also displays over the grid, which is useful because it allows the user to always view necessary tools and information.

This exploration allowed the team to manipulate different user interaction tools into a menu. The following elements are common in both standard UI and PMKS+.

TextBox:

`TextBoxes` are primarily used to record user typed input. They accept input from users as a string. Using the `TextChanged` event, text boxes are also able to call event handlers to update other parts of the application.

Label:

`Labels` are used to guide the user to certain valuable tools by displaying text..

Slider:

`Sliders` are used to hold variable information. The slider allows the user to visualize a scale of some type. The slider can also have functionality in `C#` code.

Button:

`Buttons` are the simplest form of user input. They simply perform a straightforward function when manipulated. Text and icons can be easily displayed to signify the functionality to the user.

7.2.4. Zoom In and Zoom Out

The criteria for zooming in and out applies to only certain elements in the window. For example, the menu should not be affected, as it would either make the menu unusable or cover too much of the screen. Similar to the PMKS+ version, the team created a scalable `Grid` element. This `Grid` is able to scale all the elements that are drawn on it as well as its

Background. The team achieved this by adding a `ScaleTransform` to a specified grid. The XAML syntax is written below:

```
<Grid.LayoutTransform>
    <ScaleTransform ScaleX="{Binding ElementName=uiScaleSlider,
                                     Path=Value}"
                   ScaleY="{Binding ElementName=uiScaleSlider,
                                     Path=Value}"/>
</Grid.LayoutTransform>
```

`LayoutTransform` is a framework element that offers different transformations (Microsoft, 2019). Contained in the framework is `ScaleTransform`, which applies directly to the `Grid` that the `LayoutTransform` extends. Finally, the scaling is bound to the `Slider` element named `uiScaleSlider` and is bound to the value of that element. Binding the value of the UI slider to the scaling of the grid ensures that any change in the slider will result in a change in the UI. The zooming functionality is demonstrated below in Figure 7.1.

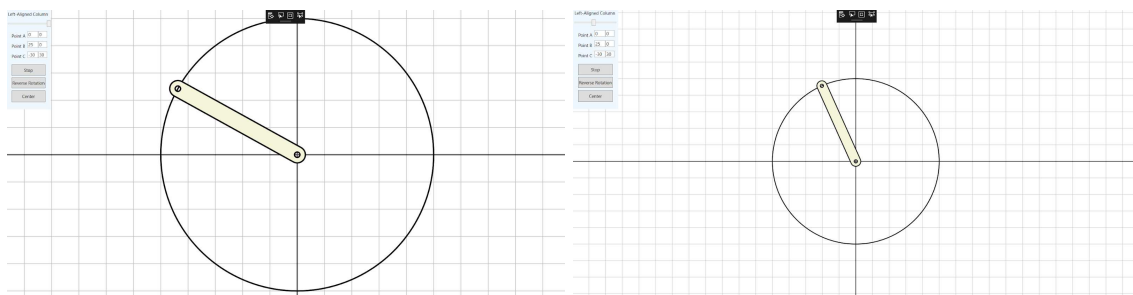


Figure 7.1.: The same linkage with different scaling within the WPF application.

7.3. Functionality

During our explorations, our team began to understand how we could potentially convert the functionality of PMKS into a WPF application. This section details the tasks that our team completed in WPF that relate to the functionality of PMKS.

7.3.1. Clock

The first task that our team explored was creating an analog clock application within WPF. Analog clocks have hands that move in a circle around the central input. We felt that this was similar enough to a rotating link and it could provide us some insight into how to rotate a shape in a circle.

Our team explored three different guides on creating an analog clock, but the most helpful of the three was by Mohammad Dayyan. Dayyan (2008) provides XAML code and C# code that defines an analog clock. The XAML file defines a `Window` containing a `Grid`, which contains three vertically-oriented `Rectangles`: `rectangleSecond`, `rectangleMinute`, and `rectangleHour`. These will be the respective hands of the clock. For each of these `Rectangles`, a `RotateTransform` is defined. The `RotateTransform` class contains `CenterX`, `CenterY`, and `Angle` properties (Microsoft Corporation, 2019g). `CenterX` and `CenterY` define the x and y coordinates of the center of rotation, and the `Angle` property sets the angle of the object in degrees (Microsoft Corporation, 2019g). Dayyan (2008) sets the center of rotation at the center of the `Window`.

The C# file first defines a `Timer` with a recurring interval of 1000 milliseconds, or one second. Within the `Window`'s constructor, the C# file adds a function `timer_Elapsed()` as an `ElapsedEventHandler` to the `Timer`'s `Elapsed` event. The `timer_Elapsed()` function uses the `Window.Dispatcher.Invoke()` function to synchronously set the `Angle` properties of the `RotateTransform`s defined above, causing the angles of the hands to update each second. Figure 7.2. below demonstrates how the clock looked when completed.

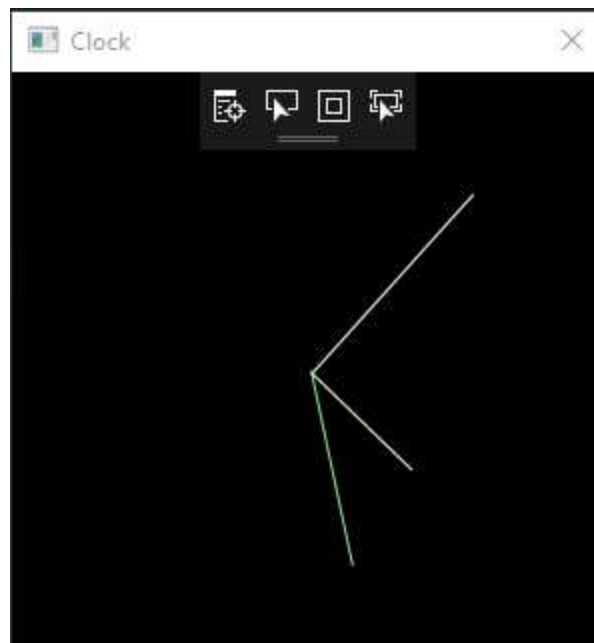


Figure 7.2.: Dayyan's analog clock in WPF.

7.3.2. Rotating a Link From Points

Dayyan's analog clock was our first attempt at considering how to rotate a link. However, most links in a linkage do not move along a completely circular path. Therefore, simply changing the angle of the link does not work. The PMKS simulator instead generates a series of coordinates for each joint in the linkage. Then, PMKS draws the links at those points in sequence to simulate motion.

In order to replicate the functionality the team first stored the x and y coordinates of a point at each degree of a unit circle within a 2D array of doubles called `circle[,]`. Then, a double `length` was defined, which would store the length of the link to be drawn. The coordinates within `circle[,]` could be scaled by `length` to reach the correct path of motion for a link of that length. Finally, an integer `counter` was defined, which counts from 0 to 359 to determine which entries in `circle[,]` to retrieve from.

Our team also used the `Timer` and `timer_Elapsed()` structure from Dayyan's (2008) analog clock to increment the counter. An interval of 1000/60 milliseconds (60 Hz) was initially chosen because that is the refresh rate of our computer screens and because it is a quick enough interval to simulate smooth motion. For comparison, movies are filmed at 24 Hz. Additionally, PMKS allows for the user to pause the rotation of the linkage. Pausing the rotation will set the `Timer`'s `Enabled` property to `False` pauses the `Timer`, which stops incrementing the counter and thus prevents the link from rotating (Microsoft Corporation, 2019h).

Rotation begins with a counter that starts at 0. Then, it is repeatedly incremented and has modulo 360 applied. Every time `counter` is incremented, a `Line` is recreated. The `Line`'s `X1` and `Y1` properties control the coordinates of the starting point of the `Line`, which our team set at the center of the screen (Microsoft Corporation, 2019d). The `X2` and `Y2` properties control the coordinates of the end point of the `Line`, which is set to the coordinates in `circle[,]` at the index of `counter` (Microsoft Corporation, 2019d). Then, the `Line`'s `Stroke` property is set to `Brushes.Black` to make the line visible on the white background. Finally, the new `Line` is added to the current `Canvas`'s `Children` property, which contains all of the UI elements for the current `Canvas` (Microsoft Corporation, 2019a). However, in order for the `Line` to animate properly the previous `Line` must be removed from the `Children` before the next `Line` is added. Figure 7.3. below contains an image of the `Line` mid-rotation within our WPF application.

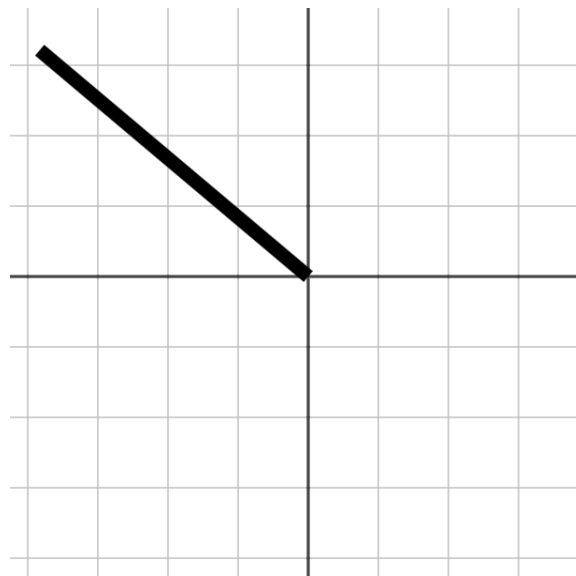


Figure 7.3: A Line in mid-rotation within the WPF application.

7.3.3. Drawing a Link Using Composite Geometries

In PMKS, links are not represented visually as simple lines. Instead, they are displayed as a long, rounded shape with circles at both ends to represent the joints, as seen in Figure 7.4. below.

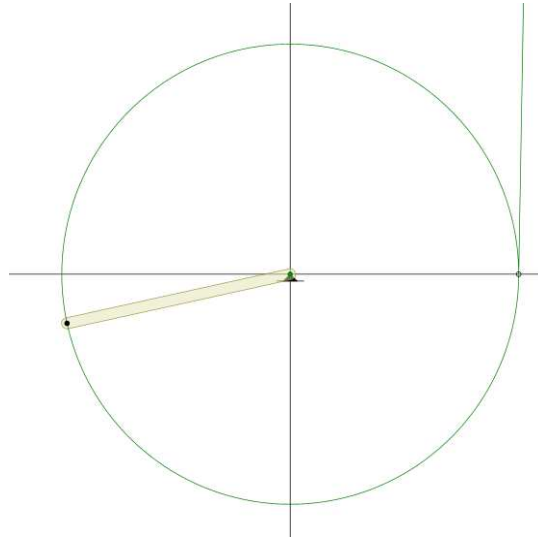


Figure 7.4.: A rotating link in PMKS.

Professor Radhakrishnan suggested to our team to use composite `Geometries` to recreate this shape. The `Geometry` class is an abstract class from which more specific `Classes` such as `LineGeometry`, `EllipseGeometry`, and `PathGeometry` inherit (Microsoft Corporation, 2019c). These `Geometries` can be added to a `GeometryGroup` and eventually displayed together on the screen as a single shape.

First, we created two `Points`: `Point1` at the center of the screen and `Point2` located at a position determined by `circle[,]`. An `EllipseGeometry` was then created at both

points by setting their `Center` property to either `Point1` or `Point2` (Microsoft Corporation, 2019b). These ellipses represent the joints at either end of the link.

To create the shape around the joints, four additional points needed to be defined. These are the two points at each end of the linkage where the semi-circle meets the straight line. We can think of each pair as the two points that are perpendicular to the direction of the link along a circle. This circle is centered around the joint and has a diameter of the width of the link. Therefore, we defined an additional double for the width of a link named `width`.

We can find these coordinates by taking the appropriate values from `circle[,]` and scaling by half the width of the link. A perpendicular line crosses another line at 90 degrees, and the angle of the direction of the link is `counter`. Therefore, the correct indexes in `circle[,]` are at $(\text{counter} + 90) \% 360$ and $(\text{counter} - 90) \% 360$.

Using these four `Points`, we can create a `PathFigure` to contain the four segments necessary to create the shape: two `LineSegments` and two `ArcSegments`. A `PathFigure` contains a `StartPoint` property, which controls where the start of the figure is (Microsoft Corporation, 2019f). The next `LineSegment` or `ArcSegment` to be added to a `PathFigure` only needs to define an `EndPoint`, using the `StartPoint` of the `PathFigure` or the `EndPoint` of the previous `PathSegment` as its `StartPoint`.

Finally, the `PathFigure` needed to be added to a `PathFigureCollection`, which then needed to be added to a `PathGeometry`. The `PathGeometry` and the two `EllipseGeometry` defined above can now be added to a `GeometryGroup`, which then is added to a `Path`. This `Path` can now be added to the `Canvas's Children` and displayed. As with the `Line`, every iteration of the `counter` causes the previous `Path` to be removed before a

new one can be added. Figure 7.5. below is an image of the link as it rotates in the WPF application.

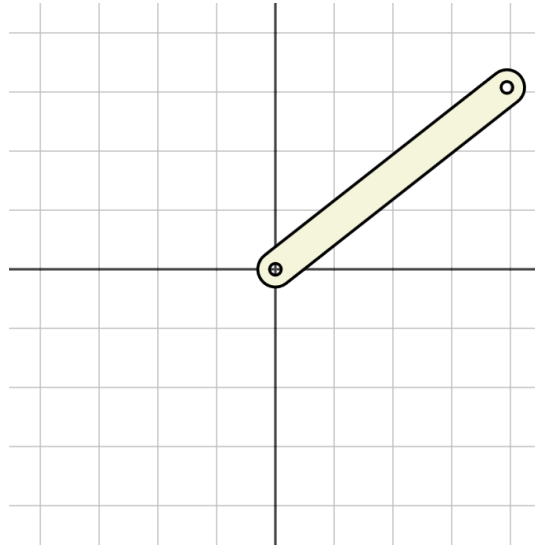


Figure 7.5.: A link created from composite Geometries rotating inside the WPF application.

7.3.4. Creating Curves

One feature of the original PMKS application is that the path of a joint's movement is displayed on the screen, as seen in Figure 7.4. above. The PMKS simulator generates a set of points that the joint moves through, and then PMKS creates a composite geometry of curves connecting each of these points to display the path.

To recreate this feature, we first focused on creating a standard circle from a set of points. The `circle[,]` array defined in Chapter 7.3.2.: Rotating a Link From Points was reused, as it already contained the points of a circular path.

First, a new `PathFigure` was created with a `StartPoint` at the first point of the circle. Then, `QuadraticBezierSegments` were used to create a curve between points and added to the `PathFigure`, as recommended to us by Professor Radhakrishan.

`QuadraticBezierSegments` contain the `Point1` and `Point2` properties (Microsoft Corporation, 2019e). The latter determines the end point of the curve, and the former determines the control point of the curve, which controls the steepness and direction of the curve (Microsoft Corporation, 2019e).

These kinds of curves have no defined starting point, and instead use the end point of the previous segment or the `StartPoint` property of the `PathFigure` if no other segments exist when added (Microsoft Corporation, 2019e). Each curve has its `Point2` set to the next x and y values in the `circle[,]` array scaled by the length of the linkage, and its `Point1` is set as the midpoint between the last point in the array and `Point2`. The last curve instead sets its end point to the `StartPoint` of the `PathFigure`, finishing the circle. Finally, a `Path` containing this `PathFigure` is added to the `Children` of the `Canvas`, and the circle is displayed.

Figure 7.6. below contains an image of the circular path displayed with a rotating link inside of our WPF application.

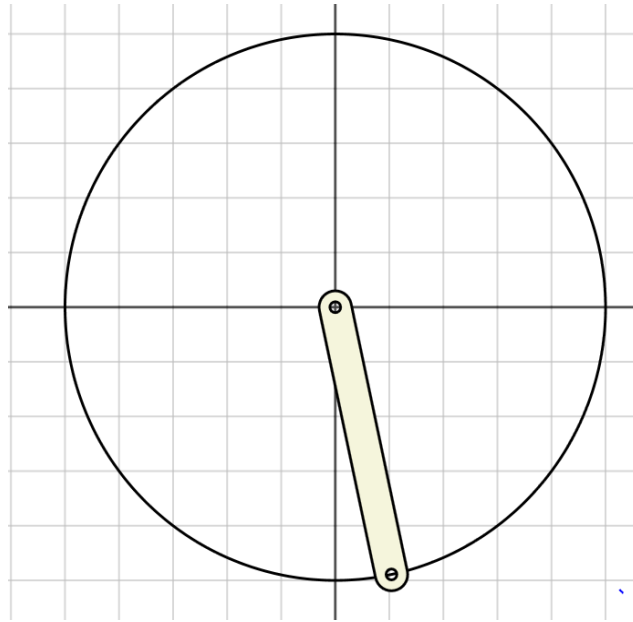


Figure 7.6.: A circular path generated in our WPF exploration application.

After the circle was completed, it was important that we then attempt to display a non-circular path. Links that aren't the input link often move in non-circular paths, and the conversion will need to display these paths as well.

Using PMKS, the team created a linkage and exported the kinematic data. From the generated TXT file, we copied the x and y values of the motion of one joint into a 2D array in our program. Using these points, a similar method using the `QuadraticBezierSegments` was employed to display the non-circular path. Figure 7.7. below shows a comparison of the path generated in PMKS and the path generated in our WPF exploration application.

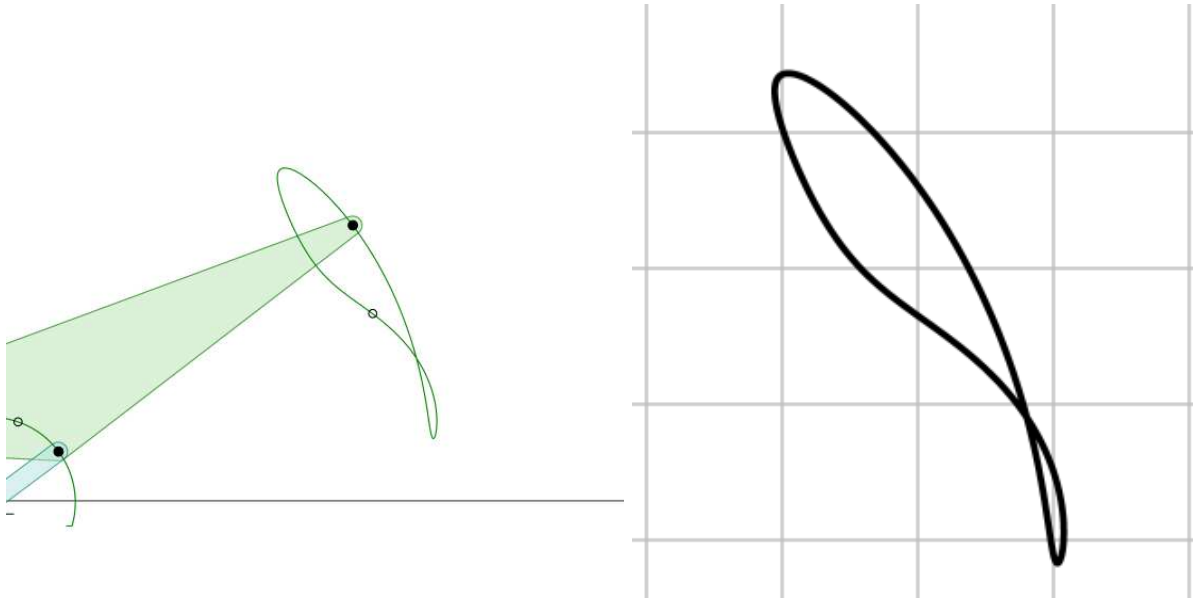


Figure 7.7.: (left) A non-circular path generated in PMKS, (right) The same path generated in the WPF exploration application.

7.3.5. TextBox Link

The UIs of PMKS and PMKS+ contain a table that displays the different properties of the joints, links, and forces of the linkage. There are two important features that the team explored to understand how to recreate a similar function in WPF: editing the joint values to translate the joint location and dragging the joint via mouse events.

Dragging the elements is discussed in more depth in Chapter 7.3.7.: Click and Drag. Using the mouse button events, the team was able to implement `TextBox` updates. The team set the value displayed in the `TextBox` to be equal to the new location of the moved element.

To create the inverse functionality the team inserted `TextChanged` fields to the specified `TextBox` declaration in the XAML file. `TextChanged` fields allow a function to be

created in C# that will be called when the `TextBox` is edited. The functions directly altered the position of the `Ellipse` elements.

Figure 7.8. below shows the `TextBox` displaying the coordinates that correlate to the blue dot element as it is dragged across the coordinate plane.

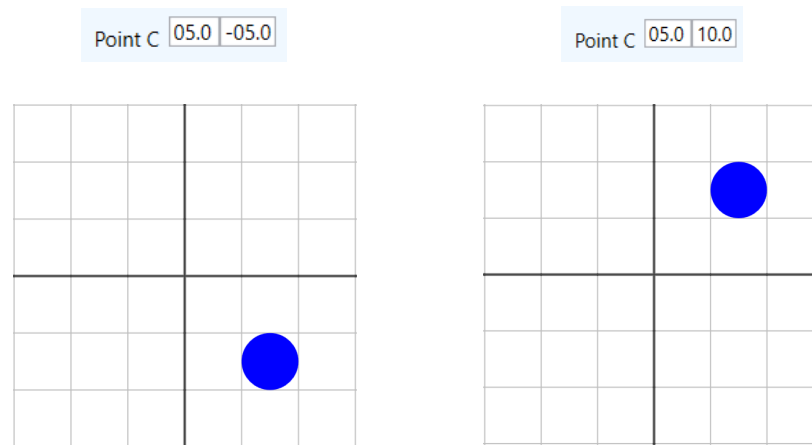


Figure 7.8.: TextBoxes updated as an Ellipse is dragged.

7.3.6. Updating the Link and Curve From TextBoxes

After the text boxes were created and linked to the variables, it was then possible to adjust the position of the link, and the curve tracing its path, using the values from the `TextBoxes`, just as it is possible in PMKS.

As mentioned in the previous section, a function is linked to the action of typing for each of the four `TextBoxes`. Within each of these functions, the team added calls to two functions that we created: `updateLength()` and `updateCircle()`.

To move the link and circle to the new location, the center point of both can simply be made equal to a new point `PointA`. However, simply updating the center point will not draw the correct link or the correct circle. To solve this issue, `PointB` is treated as one point along the circumference of a circle around `PointA`. Then, `updateLength()` calculates the radius of this circle using Pythagorean's Theorem between the two points, and this radius is the new length of the link. This will affect the link when it is redrawn next, but the circle is not updated regularly like the link is. The `updateCircle()` function fills this role by removing the previous `Path` containing the circle and adding a new one centered on `PointA`'s new location with a radius of the length of the link. The call for `updateCircle()` occurs after the call to `updateLength()`, so the circle will draw in the correct position with the correct size. Figure 7.9. below contains a comparison between the default link displayed when the application is launched and a link after moving `PointA` to $(-5, 10)$ and `PointB` to $(0, 0)$.

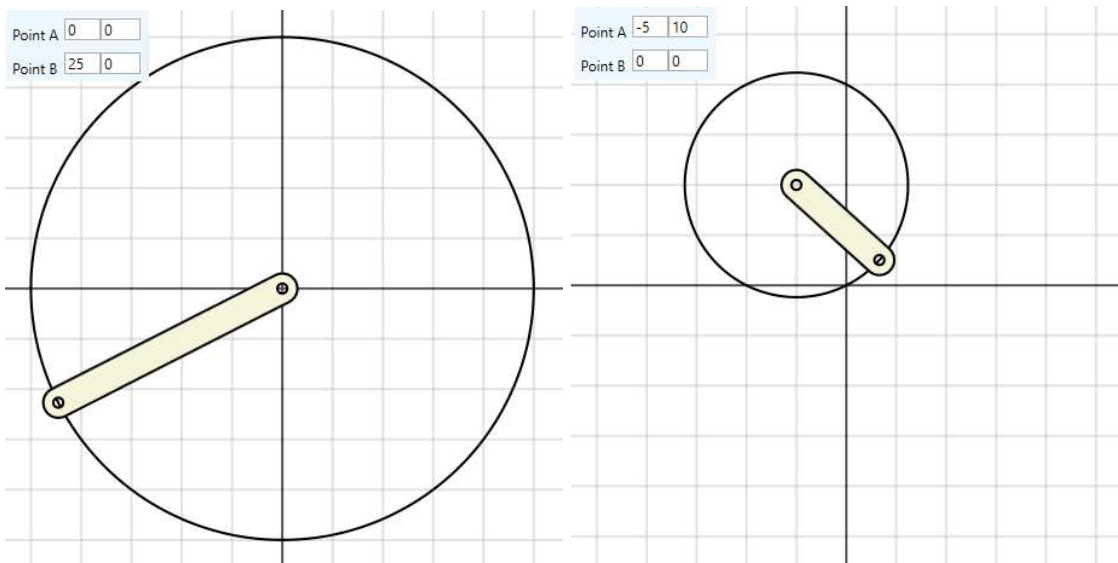


Figure 7.9.: (left) The default position and dimensions of the link and circle, (right) The link and circle after updating `Point A` and `Point B`.

7.3.7. Click-and-Drag

Click and drag is a major user interactive tool that is very helpful to learning. In many engineering modeling software systems it is important to be able to manipulate both the model and the viewpoint of the user. The team accomplished both manipulating an individual object as well as moving the canvas to where the various elements are attached.

The team began with understanding mouse click events in WPF. These mouse events require three mouse components: `MouseDown`, `MouseUp` and `MouseMove`. These three elements can be defined in XAML as three separate fields of the `Canvas` containing the elements that are being manipulated by the mouse. The team created functions that corresponded to the `MouseLeftButtonDown`, `MouseLeftButtonUp`, and `MouseMove` fields in the XAML file.

By creating the functions that correspond to these fields, the team was able to add functionality to these different mouse events in C#. The function name in C# will be connected to the mouse event declared in the XAML. The group discovered that the program can compare the mouse's location to the location of UI elements to determine which element is being clicked. The elements and location of the mouse are stored in global variables that can be accessed by the other mouse functions. The mouse up function simply resets the global variable that records the element to null. Finally mouse move uses the information of the dragged element and sets the position of the element to the position of the mouse.

Understanding the mouse click events allowed the group to use mouse left and mouse right events to distinguish between different dragged elements. The left mouse click was used for

dragging the whole coordinate plane to view certain parts of the linkage. The right mouse click was used to move different elements along the coordinate plane.

An example of the two different transformations are shown below in Figure 7.10. The axis which is initially on the right was moved to the left using the left mouse button events. Separately, the blue dot was dragged up to the first quadrant from the fourth quadrant in relation to the axis, as well as moved left to remain at the same portrayed coordinates.

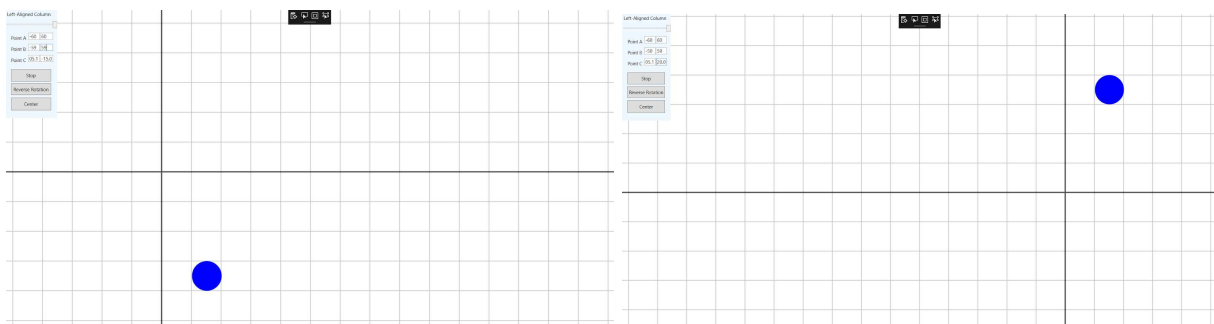


Figure 7.10.: A blue dot before (left) and after (right) click-and-drag is performed.

8. PMKS+ Interface

The team that developed PMKS+ made several changes to the interface design and controls from PMKS, the original Silverlight version (Appikatla, et al., 2019). The improved interface and linkage creation methods of PMKS+ are what our team used as a basis for our new PMKS+D user interface. This section will explain the changes made for PMKS+ and the development team's justification for these changes.

For later reference, Figure 8.1. below is an image of the original PMKS Silverlight application. Pay attention to the visual style and the organization of the user interface, as these elements will be contrasted with PMKS+'s UI.

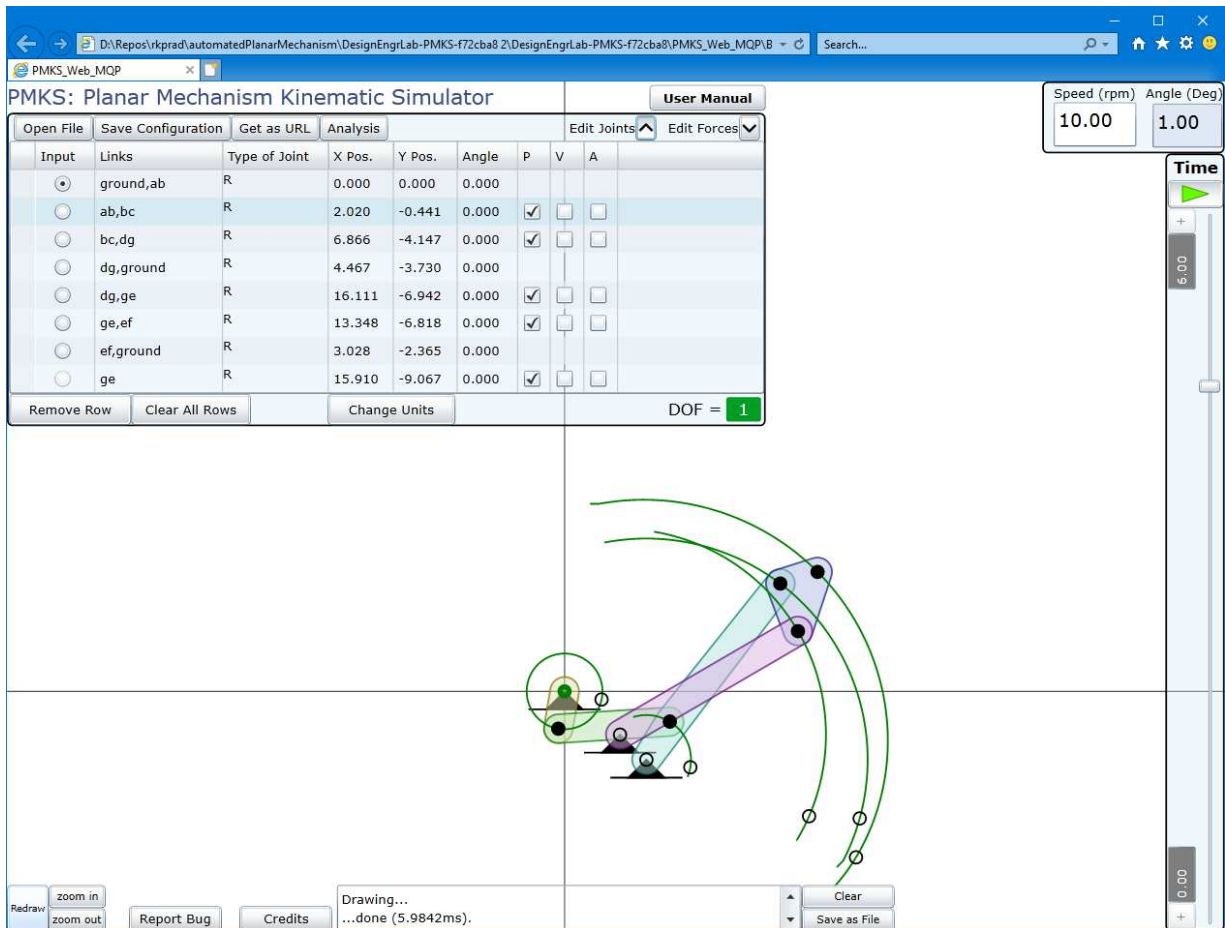


Figure 8.1.: Example of a linkage in PMKS Silverlight.

8.1. PMKS+ Layout Justification

The PMKS+ team conducted multiple surveys and collected feedback to justify the implementation of their interface changes. PMKS+ performed surveys early on in the design process to evaluate initial interface design choices. This feedback, coupled with feedback from the advisors, allowed the team to create seven iterations of their interface and improve the state of their application into the final product.

One major change was that the team moved the Open File, Save File, Get URL, and Analysis buttons from the top of the table into a tab menu at the top of the screen while removing redundant copies of these buttons. This allowed them to group buttons more effectively under separate, labelled tabs while keeping the UI compact. For example, the Open and Save buttons were grouped together under the File tab, while the different analysis options were grouped under the Analysis tab. Figure 8.2. below demonstrates these two tabs within PMKS+'s UI.

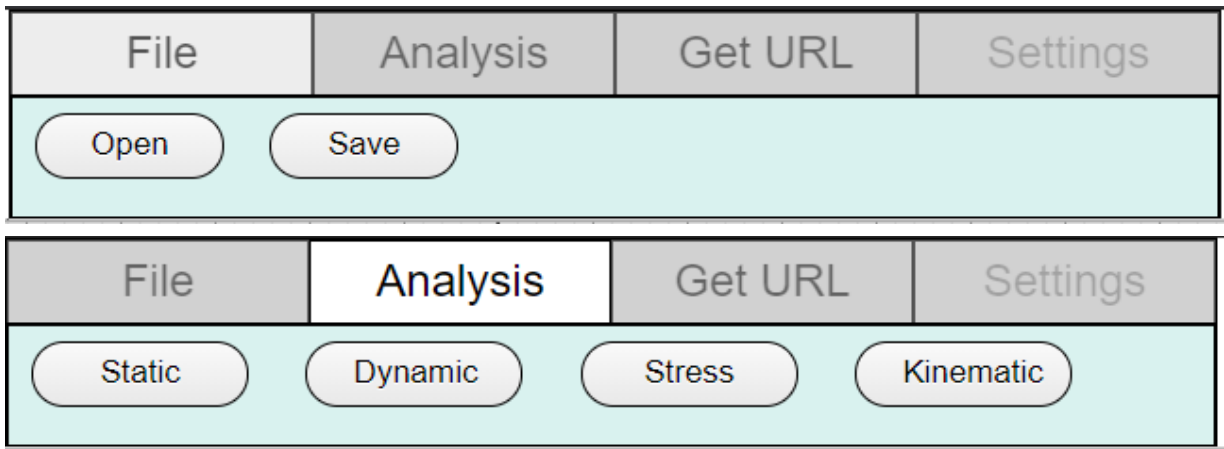


Figure 8.2.: File and Analysis tabs in PMKS+.

A second major change was the redesign of the controls for the linkage animation and the canvas. The controls were given a graphical update to appear more modern, while the buttons had their text exchanged for symbols that were easier to understand. The linkage animation controls were moved from the right side of the window to the bottom to have the progress slider more closely imitate a left-right timeline. This also allowed the play button and progress slider to be enlarged while allowing for additional components like the pause button. The controls for the

canvas, such as the zoom in and zoom out buttons, Clear button, and Redraw button were grouped together and repositioned to the bottom right. PMKS+'s updated lower menu is shown in Figure 8.3. below.

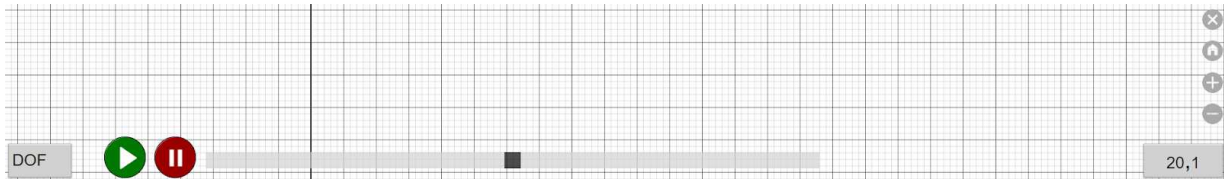


Figure 8.3.: PMKS+'s updated lower menu.

Finally, the Credits, User Manual, and Report Bug buttons were renamed to About, Help, and Report and grouped in the top right.

Figure 8.4. below is a screenshot of the final UI iteration designed by the PMKS+ team.

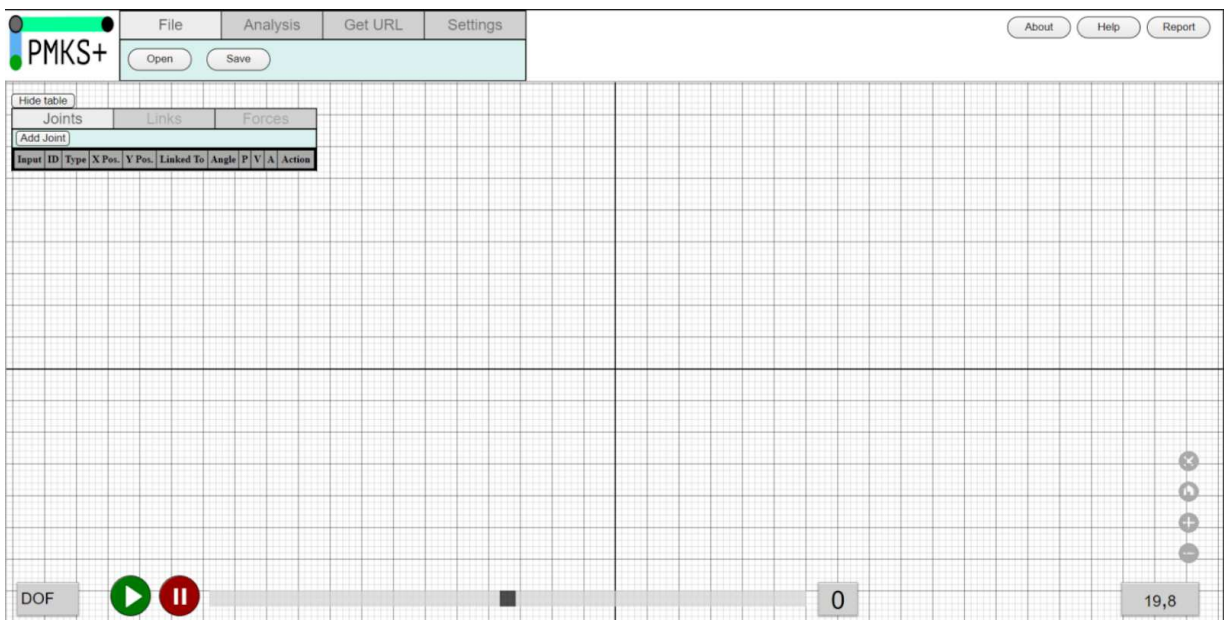


Figure 8.4.: The final iteration of PMKS+'s UI.

8.2. Linkage Creation Justification

The PMKS+ team also reengineered the linkage creation method. In PMKS Silverlight, the user created the linkage by typing into tables to add new joints, links, and forces. PMKS+ created a mouse-based method involving context menus and mouse clicks. The team conducted an in-depth evaluation of three potential linkage creation methods:

- Click-Drag: the user clicks to create a joint and releases the click at the second location.
- Click-Click-Connect: the user clicks two joints to connect them with a link.
- Click-Click: the user clicks once to create a joint and clicks a second time to place it.

The team concluded that the Click-Click method would be the most effective because it was similar to hand-drawing, efficient, and easy to use on trackpads.

The PMKS+ team also evaluated the advantages and disadvantages of a context menu-based implementation using right clicks. While these context menus added mouse steps and some complexity to the creation process, they are straightforward, descriptive, and intuitive. The team ultimately concluded that a context menu click-click method of linkage creation was a good substitute for the original method.

9. Interface Design

The team implemented the following interface design features to improve the user experience while maintaining PMKS's functionality. Using combinations of XAML elements, the team elemented the four main components that were consistent through the previous applications: the tables, canvas, lower menu, and upper menu.

9.1. Tables

The interface has three tables that can be selected using tabs, which provide information about the three components of a linkage: Joints, Links, and Forces. The order was chosen due to the order that components can be added to the linkage. A joint is the first component that a user can create, followed by a link connecting two joints. Finally, a force can be added to the link. The tables can be shown or hidden at the user's discretion through the **Show Table** button to view the table and **Hide Table** button to hide the table.

The tables serve multiple roles, including user feedback and user input. Feedback is provided to the user by cells in the table being updated with new information as the linkage is directly manipulated. The user also has the opportunity to manipulate the linkage from the table to more precisely create linkages, e.g., by placing a joint in an exact location. The additional information and precision the tables provide improves the user experience.

These tables were implemented using a `TabControl` embedded in a `Grid` element to allocate the correct space for the tables. The `TabControl` allows the user to switch between

the three different DataGrids. Each DataGrid has several DataColumnns for the different fields, and new DataRowns are created to display the data of each Joint, Link, and Force in the table.

9.1.1. Joints

The Joints tab provides information on the current joints of the linkage. The user has the ability to edit the initial x- and y-coordinates of the joint. Once the user presses the enter key after editing a cell, the joint will be updated to the location inputted. Furthermore, as the user drags a joint, the x- and y-position cells for that joint will be updated on the table. A list of the fields of the table with a short description is displayed below:

- **Joint ID:** a capital letter assigned as the ID of the joint;
- **Input:** whether the link is set as the input joint or not. Note that checkboxes are used to indicate Boolean values;
- **Type:** the type of joint: revolute (R) or prismatic (P);
- **Linked To:** a list containing the IDs of the links that the joint is connecting;
- **X Pos:** the x-position of the Joint when the linkage is stationary;
- **Y Pos:** the y-position of the Joint when the linkage is stationary.

The order of these fields were kept mostly consistent from PMKS+. However, we decided to move the x and y position columns to the right so that users would not have to move their mouse as far away from the linkage to click on the table and begin typing. Figure 9.1. below is an image of the Joints table in PMKS+D containing two example joints, A and B.

Joints		Links		Forces	
Joint ID	Input	Type	Linked To	X Pos	Y Pos
A	<input checked="" type="checkbox"/>	R	1,2	-2.853	-1.147
B	<input type="checkbox"/>	R	2	-1.187	-2.107
	<input type="checkbox"/>				

Figure 9.1.: Joints table in PMKS+D containing two example joints, A and B.

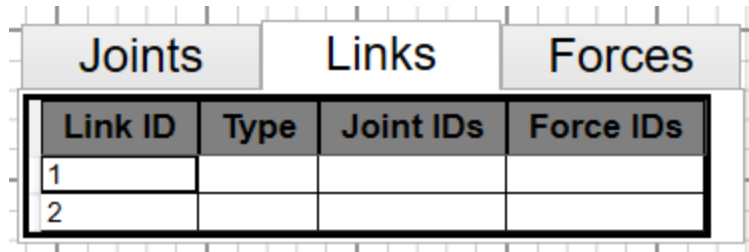
9.1.2. Links

The Links tab provides information about the current links of the linkage. Due to time constraints, full user input and interaction was not implemented beyond displaying the ID of each link. However, the following fields are present in the column headers and their future function is described below:

- Link ID: a positive integer assigned as the ID of the link;
- Type: the type of link (Polygonal link or Bar link);
- Joint IDs: a list containing the IDs of the joints connected to the link;
- Force IDs: a list containing the IDs of the forces applied to the link.

The order of these fields was chosen to be consistent with the joints table. In the joints table, the columns display the joint ID, the type of joint, and then the IDs of the links the joint is on. Therefore, we chose to have the columns display the link ID, the type of link, and then the IDs of the joints and forces attached to the link. Figure 9.2. below shows the links table in

PMKS+D with two example links, 1 and 2. Unfortunately, due to time constraints, only the Link ID column has been implemented.



Joints		Links		Forces	
Link ID	Type	Joint IDs	Force IDs		
1					
2					

Figure 9.2.: Links table in PMKS+D with two example links, 1 and 2.

9.1.3. Forces

The Forces tab provides information about any forces present in the linkage. A user can edit the Angle, Magnitude, and Fixed Force columns and the force will be updated in the linkage. Also, as the user manipulates the force using click-and-drag, the respective information within the table will be updated to the correct values. All of the field headers are described below:

- **Force ID:** a positive integer assigned as the ID of the force. An “F” is prefixed (e.g. F1) to separate force IDs from link IDs;
- **On Link ID:** the ID of the link to which the force is connected;
- **Angle:** the angle in degrees of the force’s application. If the force is global, the angle is relative to the x-axis. Otherwise, it is relative to the link’s angle;
- **Magnitude:** the magnitude of the link in Newtons;
- **X Pos** - the x-position of the force’s point of application;

- Y Pos - the y-position of the force's point of application;
- Fixed Force - true (checked) if the force is a local force, meaning that the angle is constant relative to the link's angle. False (unchecked) if the force is a global force, meaning that the angle is relative to the x-axis.

The order of these fields is consistent with the forces table in PMKS, with two changes. Firstly, the columns for inputting the x and y components of the force's magnitude were replaced with columns for inputting the angle and magnitude of the force because the team believed that this was more intuitive. Secondly, the columns for x position and y position were repositioned to the right for quicker access, similarly to the Joints table. Figure 9.3. below shows the forces table in PMKS+D with an example force, F1.

Joints		Links		Forces		
Force ID	On Link ID	Angle	Magnitude	X Pos	Y Pos	Fixed Force
F1	2	45	7.071	2.009	-1.415	<input type="checkbox"/>
						<input type="checkbox"/>

Figure 9.3.: Forces table in PMKS+D with an example force, F1.

9.2. Canvas

The Canvas is the primary interface of the application. The Canvas displays a Cartesian plane with the linkage overlaid on top. The linkage's visual representation might consist of graphics for links, joints, paths, forces, and others. The graphical components used to display the linkage can be manipulated with three user interface techniques: click-and-drag, context menu, and mouse scroll. These three interactions allow the user to create and customize various linkage configurations and will be discussed in the following section in more depth.

9.2.1. Graphics

Multiple graphics were implemented to visually represent the various different linkages allowed by this application. These graphics add to the user's ability to comprehend the structure and state of the linkage.

Joints and links are the primary graphics used to display the linkage. All joints are represented as circles with a white center to provide more contrast to the link compared to the original PMKS version. There are two types of links that the user can create: a bar link connecting two joints and a poly link connecting three or more joints. The outlines of the links and joints are dark and solid to provide contrast with the white background. The interior of the link is a partially transparent color to allow the user to see any components that are behind the link. In Figure 9.4. below, a bar link and poly link with their joints are shown in both PMKS and PMKS+D.

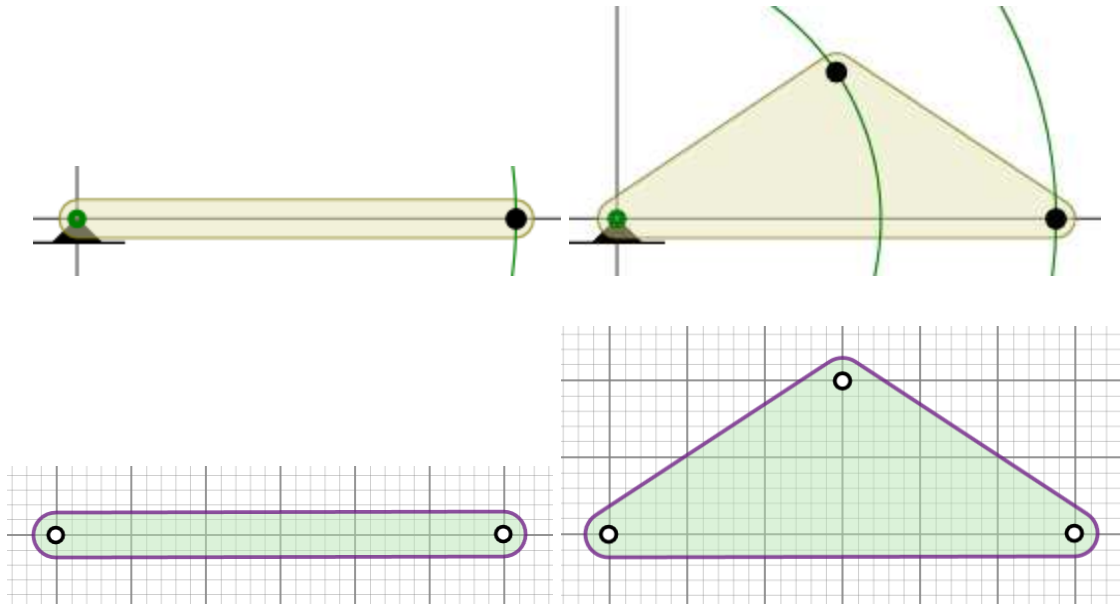


Figure 9.4.: The two types of links in PMKS and PMKS+D: PMKS (top) and PMKS+D (bottom); Bar links (left) and poly links (right).

The ground graphic is used to display to the user which joints are connected to the ground link. This is important because the grounded joints determine the movement for the rest of the linkage. For revolute joints, a ground joint is stationary and is displayed in Figure 9.5. below:

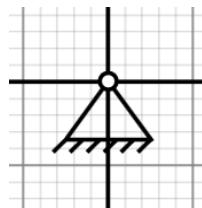


Figure 9.5.: A ground joint with a ground graphic in PMKS+D.

The slider graphic is implemented to display a prismatic joint. A prismatic or slider joint is a joint that is fixed to a linear path rather than rotating, representing a slider on a “track”. The team changed the slider graphic from PMKS because the team felt that the old graphic was less intuitive. In PMKS, a slider was shown as a slotted bar sliding around a fixed point, as seen in Figure 9.6. below. Professor Radhakrishnan advised us that linkages in the real world would be better represented by a peg sliding within a track. The team designed two new graphics to convey this idea.

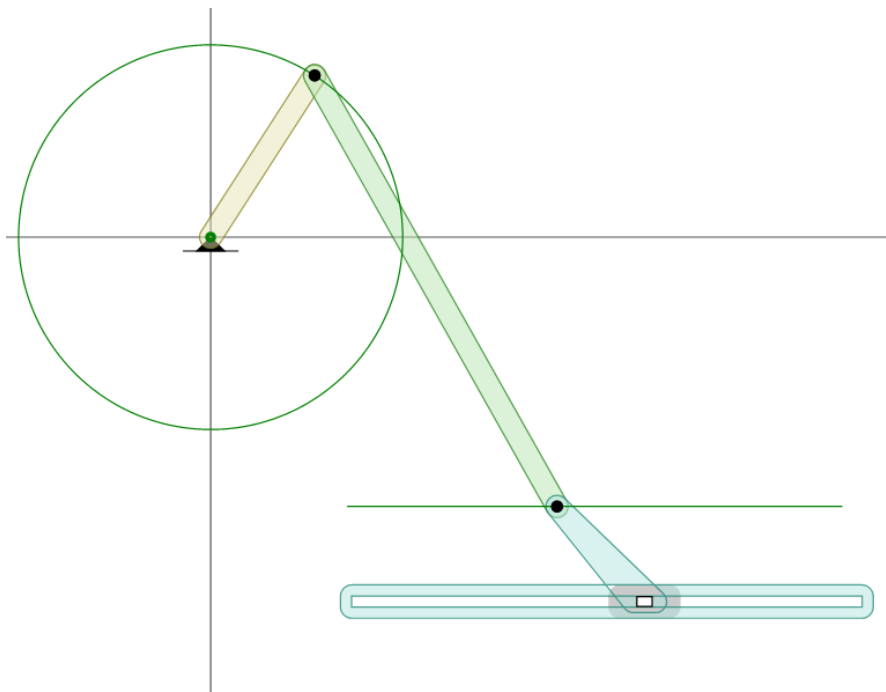


Figure 9.6.: A linkage with a slider in PMKS.

The new slider graphic is represented as a square around the specified joint. It is meant to represent a solid piece sliding within a track, so it is styled similarly to a link in color, outline, and size. Figure 9.7. below is an image example of the prismatic joint graphic.

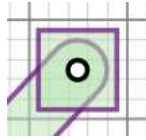


Figure 9.7.: A prismatic joint with a slider graphic in PMKS+D.

The track graphic for prismatic joints is similar to the ground graphic for revolute joints to inform the user that the joint is limited to the specified movement. The track graphic displays the outer limits of the track as well as the entire length in between. Figure 9.8. below is an example of the grounded prismatic joint.

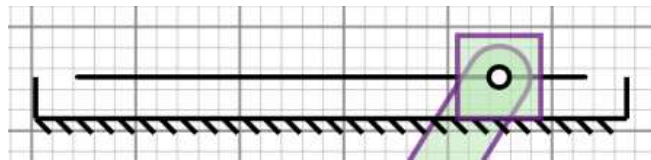


Figure 9.8.: A prismatic joint with a slider and track graphic in PMKS+D.

The final graphic implemented was a force graphic. This graphic is used to display to the user all of the forces that act on the linkage. The tail of the force is located where the force is applied, and the angle and length of the force's head correspond to the angle and magnitude of the respective force. This provides visual feedback to the user about the status of a force. Figure 9.9. below is an image of the force graphic.

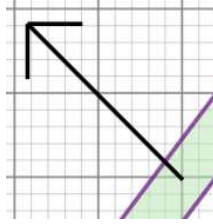


Figure 9.9.: A force graphic on a link in PMKS+D.

The force has two modes that are presented to the user: local and global. A local force has a fixed angle related to the link it is connected to, while a global force's angle is relative to the center of the Canvas. During rotation, a local force will continue to act on the link in the same relative direction as the link moves, while a global force will act in the same absolute direction no matter how the link moves or rotates.

9.2.2. Click-and-Drag

Click-and-Drag is the first primary feature of the Canvas. Click-and-drag functionality has two components: editing the linkage and panning the view of the grid. This control scheme gives the user an intuitive way to edit the linkage while receiving visual feedback, which enhances the user experience.

In order to provide feedback to the user that the execution of an action was successful, there are some visual components the team implemented as part of the interface. First of all, two TextBoxes at the bottom-left of the screen display the current location of the mouse relative to the Cartesian plane of the Canvas. Secondly, the user receives feedback when editing the linkage through the links and joint paths changing in real time as the user manipulates the linkage to a

desired outcome. Finally, when panning the view, the view is updated in real time so the user knows where the linkage will be when they release the mouse button.

To ensure that the manipulation is intuitive, the team implemented colored shapes that appear when the mouse is over a linkage element. These “hitboxes” inform the user of which element is currently being manipulated. The team implemented three different `Shape` elements for the components of a linkage: `Ellipse` for joints and forces, `Rectangle` for bar links, and `Polygon` for poly links. A combination of these three elements are added to the `Canvas` when the linkage is created, and remain invisible until the user's cursor is moved over the shape. Using `MouseMoveEvents` to detect the state of the mouse, only one hitbox element at a time may remain visible. Examples of these hitboxes are displayed below in Figure 9.10.

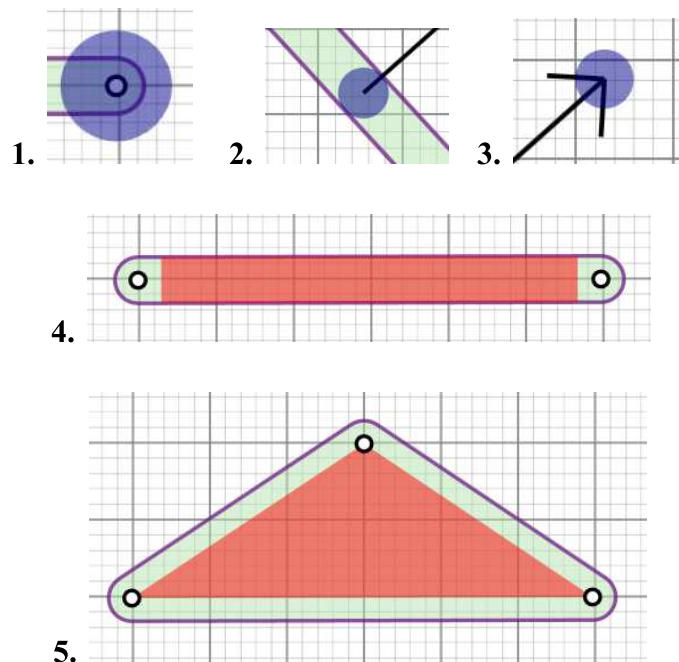


Figure 9.10.: Examples of hitbox graphics in PMKS+D: 1. A joint hitbox, 2. A hitbox for a force's tail, 3. A hitbox for a force's head, 4. A bar link hitbox, 5. A poly link hitbox.

Furthermore, the Ellipse hitbox contains additional functions to track the click-and-drag events. If a `MouseEvent` is triggered when the mouse cursor is over a Joint or Force's Ellipse hitbox, the program will capture the mouse cursor's position and allow the user to move the joint or force's position with the mouse cursor. A `MouseEvent` event triggers updates to any links, table cells, or movement path graphics as needed. Finally, a `MouseEvent` disables the editing.

The same strategy is used to pan the Canvas. If a `MouseEvent` is triggered when the mouse cursor is over an empty area of the Canvas, `MouseEvent`s will pan the view, and a `MouseEvent` disables the panning.

9.2.3. Context Menu

The team added context menus as an additional form of user interaction, following PMKS+'s recommendations. The context menus provide straightforward lists of options that the user can intuitively select by using the mouse. The text of the options were chosen to be descriptive and maintain continuity with existing design elements.

The context menus are displayed when the user right-clicks an element and are separated into three categories — link, joint, and Canvas — depending on where the user's cursor is at the time of clicking. The contents of the context menus vary depending on the state of the link, joint, or Canvas when the user clicks. Below are lists of what each context menu may contain, as well as a visual example of the context menu in a specific condition.

Link:

- **Add Force:** adds a force to the specified link.
- **Create Joint:** adds a joint to the specified link.
- **Show Link Kinematics:** opens a window that allows the user to view different graphs of kinematic information for the specified link.

Joint:

- **Create Link:** attaches a new bar link with a revolute joint to the specified joint.
- **Toggle Ground:** toggles the specified joint's connection to the ground link on or off.
- **Delete Joint:** removes the specified joint. If the joint is on a bar link, the bar link will be deleted as well.
- **Add Slider:** attaches a new bar link with a prismatic joint to the specified joint.
- **Hide or Show Position Path:** toggles the display of the specified joint's movement path.
- **Show Joint Kinematics:** opens a window that allows the user to view different graphs of kinematic information for the specified joint.

Canvas:

- **Create Joint:** will create a new grounded input joint if no linkage exists on the grid.

Figure 9.11. below shows an example for each category of context menu in PMKS+D.

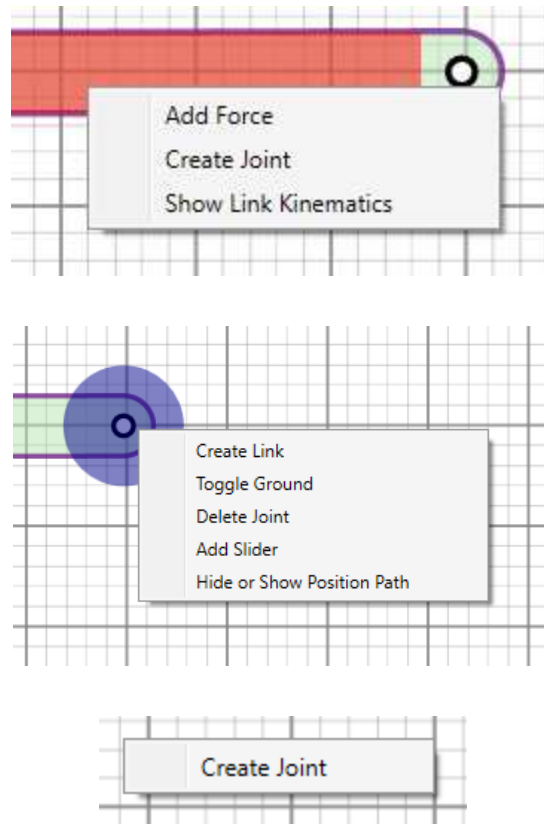


Figure 9.11.: Examples of context menus in PMKS+D: (top) A link context menu, (middle) a joint context menu, (bottom) a Canvas context menu.

9.2.4. Mouse Scroll

The final form of interaction with the Canvas is the use of the scroll wheel on the mouse. An event handler detects when a `MouseWheel` event triggers on the Canvas, which allows the user to adjust the zoom level of the Canvas and all linkage elements on it. This allows the user to fit the view to linkages of various sizes. As the user scrolls the wheel toward themselves, the

Canvas will shrink in size and display more of the linkage. When the user scrolls away from themselves, the Canvas will grow in size, causing the view to display less of the linkage by making it look larger.

9.3. Lower Menu

The PMKS+D UI has a collection of controls along the bottom of the screen that affect the display of the linkage. This menu is divided into two major sections: animation control and canvas control.

9.3.1. Animation Control

The animation control section allows the user to control how the linkage is animated on screen. A list of controls and their functions are given below:

- **Degrees of Freedom (DoF):** A `Label` that shows the current DoF of the linkage. The linkage can only animate if there is 1 degree of freedom;
- **Input Direction:** A `Button` used to toggle the direction of the input link's rotation between clockwise and counterclockwise;
- **Input Speed:** A `Button` used to toggle the speed of the input link's rotation between three different settings;
- **Play/Pause:** A `Button` used to begin the animation if the linkage can animate or pause the animation if the linkage is animating. This button disables linkage editing;

- **Stop:** A `Button` used to stop the animation and reset the linkage to its initial position. This button enables linkage editing;
- **Timeline:** A `Slider` used to scan the linkage's animation timeline.

These controls are similar to PMKS+'s animation controls with some modifications. The `Input Direction` and `Input Speed` buttons are new, and meant to give the user more control over how they would like the linkage's animation to look. The `Play` and `Pause` buttons were combined and turned into a toggle button to match how common media players function, such as YouTube. Finally, the `Stop` button was added to reset the linkage back to its initial position. Due to the way PMKS's simulator works, the linkage is required to be in its initial position when being edited, so the `Stop` button was added as a shortcut.

One design decision our team made was to have the animation controls always display the next available action rather than the current state of the animation. For example, the `Play` button shows when the animation is paused to indicate that the user can play the animation, and the `Input Speed` button displays "x2" when the linkage is animating at "x1" speed to indicate that the user can toggle the speed to "x2" speed. The team made this decision because we felt that it was consistent with the way that buttons are normally used; namely, that a button is pressed to perform the indicated action.

Figure 9.12. below displays the animation control section, as well as button variants for the toggleable buttons.

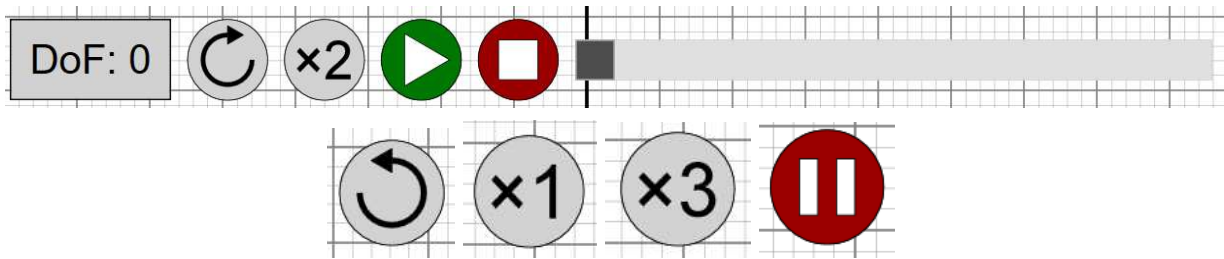


Figure 9.12.: The animation controls in PMKS+D: (top) The default animation controls, (bottom, left-to-right) Counterclockwise variant of Input Direction button, x1 and x3 speed variants of Input Speed button, Pause button

9.3.2. Canvas Control

The Canvas control section provides the user with button shortcuts to actions that affect the entire Canvas. A list of controls and their functions are given below:

- **Display IDs:** A Button used to display the IDs of the linkage's joints, links, and forces on the Canvas;
- **Zoom In:** A Button used to zoom in on the Canvas. Functions identically to using the mouse wheel;
- **Zoom Out:** A Button used to zoom out of the Canvas. Functions identically to using the mouse wheel;
- **Center:** A Button used to reset the center of the Canvas to the center of the screen;
- **Clear:** A Button used to delete the current linkage from the Canvas.

These controls are similar to PMKS+'s Canvas controls, with some modifications. First of all, rather than the controls being arranged in a vertical line like in PMKS+, they were made horizontal and inline with the animation controls to be more visually consistent. Secondly, the icon for the Center button was redesigned from a house to a set of arrows pointing towards the center of the icon. This was done to make the purpose of the button more immediately apparent. Finally, the Display IDs button was added to provide the user with a way to view the linkage with the IDs from the tables displayed beside the respective components. Therefore, the user can better understand which joint, link, and force corresponds to each entry in the table.

Figure 9.13. below displays the Canvas controls in PMKS+D.

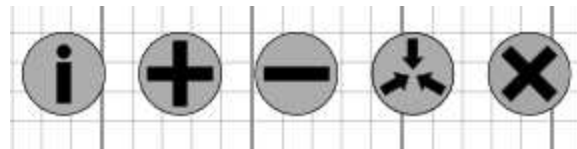


Figure 9.13.: The Canvas controls in PMKS+D: (left-to-right) Display IDs button, Zoom In button, Zoom Out button, Center button, and Clear button.

9.4. Upper Menu

The last major collection of controls within the PMKS+D UI is the upper menu. This menu has four tabs that each contain controls related to a certain topic: **File**, **Analysis**, **Settings**, and **Help**. As explained within Chapter 8.: PMKS+ Interface, the tab menu was chosen by the PMKS+ team to group related controls under labelled sections while keeping the interface compact.

9.4.1. File

The **File** tab contains controls that handle the input/output (I/O) of linkage configuration files for PMKS+D. Figure 9.14. below displays the File tab in PMKS+D.

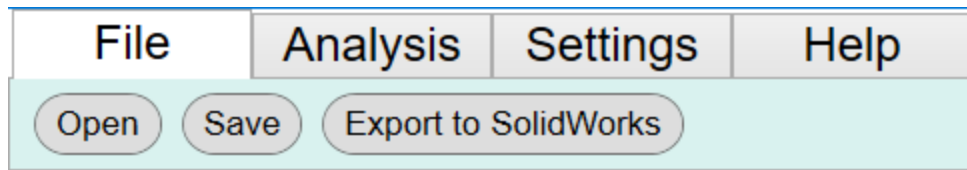


Figure 9.14.: The File tab in PMKS+D.

The **File** tab contains three **Buttons**:

- **Open**: A **Button** used to select a saved linkage configuration Comma-Separated Values (CSV) file. Opens a File Explorer window;
- **Save**: A **Button** used to save the current linkage configuration to a CSV file. Opens a File Explorer window;
- **Export to SolidWorks**: A **Button** used to export the current linkage configuration into SolidWorks, generating 3D model files that construct the linkage.

The PMKS+D File tab resembles PMKS+, with some modifications. First of all, the CSV file format was chosen for its human readability and compatibility with both PMKS+D and PMKS+. Readability to humans allows users to edit the linkage configuration within the CSV file without requiring access to PMKS+D or PMKS+, and the mutual compatibility allowed us to implement saving that is compatible with both softwares. The shared save files made URL

generation redundant, so the **Get URL** tab was removed. More information about how the opening and saving functions is found in Chapter 10.3.5.: Open and Save Linkage.

Secondly, the ability to export the linkage to SolidWorks was added because it was a feature that the team felt would improve the UX. PMKS+D allows users to create and simulate linkages digitally. However, if a user wanted to recreate that linkage as a Computer-Aided Design (CAD) 3D model so they could physically build it, they would have to do the entire process by hand. More information about how the SolidWorks export functions is found in Chapter 10.4.1.: SolidWorks Export.

9.4.2. Analysis

The **Analysis** tab contains controls that handle the display and I/O of analysis data on the current linkage. Figure 9.15. below displays the **Analysis** tab in PMKS+D.

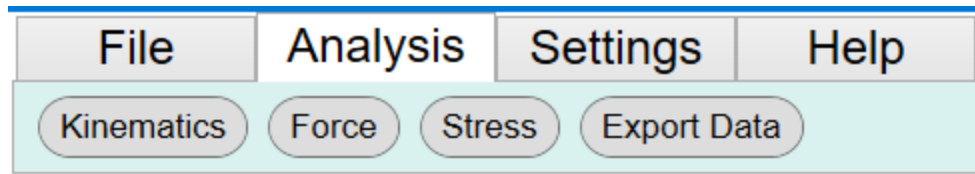


Figure 9.15.: The Analysis tab in PMKS+D.

The **Analysis** tab contains the following controls:

- **Kinematics:** A `Button` used to open a graph displaying kinematic data, such as position, velocity, or acceleration over time, vector loops, or instant centers;
- **Force:** A `Button` used to open a graph displaying static or dynamic force data;

- **Stress:** A Button used to open a graph displaying stress data;
- **Export Data:** A Button used to export a file containing analysis data for PVA, vector loops, instant centers, static and dynamic forces, and stress.

The layout of the **Analysis** tab is consistent with PMKS+, except for the addition of the **Export Data** button. The **Export Data** button was added to replace the multiple buttons from the PMKS interface and make the interface more quick to read. Now, when the **Export Data** button is clicked, a context menu appears that allows the user to select which data they would like to export. This can be seen in Figure 9.16. below.

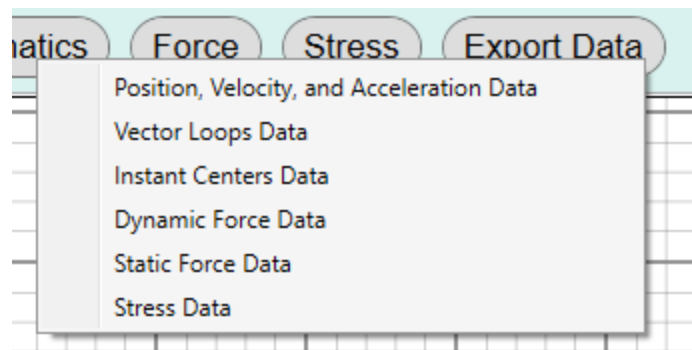


Figure 9.16.: The Export Data button's context menu.

The **Kinematics**, **Force**, and **Stress** buttons have new functionality, which is the ability to open a graph displaying the linkage's analysis data. Previously, the user would have to export the data and create these graphs themselves. Similarly to the SolidWorks export, this functionality improves the UX by automating this process.

These three buttons open context menus styled similarly to the **Export Data** button's menu in Figure 9.16., allowing the user to select which graph they would like to open. Unfortunately, due to time constraints, only mockup graphs for position, velocity, and acceleration (PVA) data were implemented.

The graph is centered and large in the window to provide the best visibility to the user. Along the top of the window is a set of controls. The team decided to implement a dropdown menu to allow the user to select the position, velocity, or acceleration data that they would like to display on the graph. The **Export Selected Data** button exports the analysis data of the graph displayed on screen. The **Save Selected Data as Image** button allows the user to save the graph as an image. Finally, the **Export All Data for this...** button allows the user to export all analysis data for the currently selected joint, link, or force.

The mockup layout used for the PVA graph can be seen Figure 9.17. below. This layout will be used to implement the other graphs in the future. For more information about the implementation of these graphs, see Chapter 10.4.2.: Graphing.

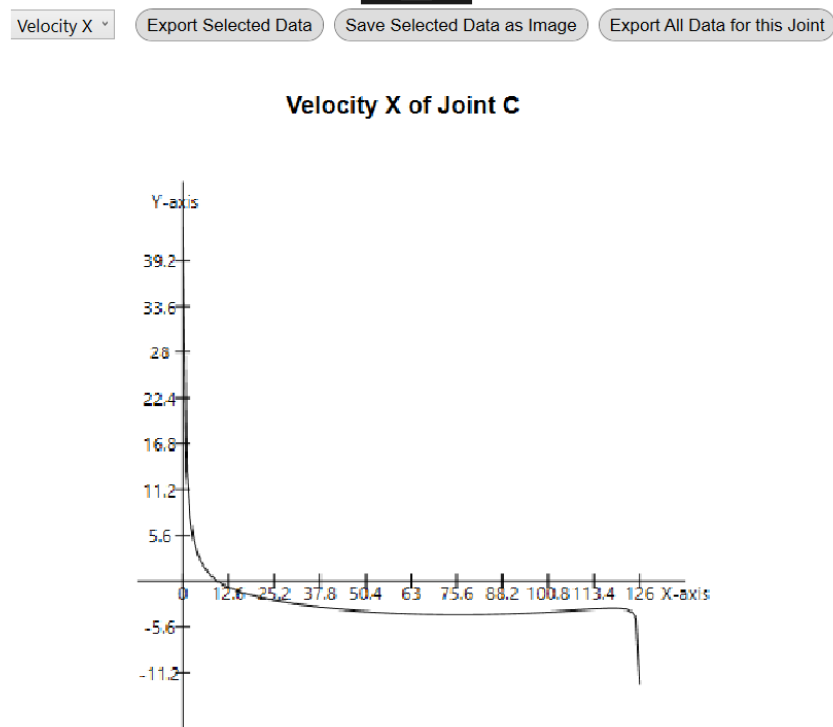


Figure 9.17.: Graph mockup for the x-velocity of an example joint.

9.4.3. Settings

The Settings tab contains controls that allow the user to customize how the simulator performs the analysis. The Settings tab is shown in Figure 9.18. below.

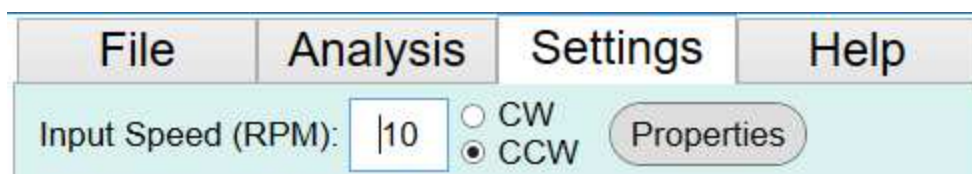


Figure 9.18.: The Settings window in PMKS+D.

The **Settings** tab contains the following controls:

- **Input Speed:** A `TextBox` used to change the rotations per minute (RPM) of the input link. Used for analysis on the linkage;
- **CW/CCW:** `RadioButton`s used to select if the input link's rotation is clockwise/counterclockwise. Used for analysis on the linkage;
- **Properties:** A `Button` used to open a window containing material and dimensional properties. Used for analysis on the linkage.

The **Settings** tab in PMKS+ was unimplemented, so our team needed to decide which controls to put within this section. In PMKS, the user was able to customize how the analysis was performed. They could type the RPM of the input link into a textbox, and open a window to select the material used to construct the linkage and the size of the links and joints. The team decided to preserve this functionality.

For the input speed, we decided that a textbox was most efficient at allowing the user to set the RPM they desired. For clarity, we also added the **CW/CCW** radio buttons so the user knows whether the input is rotating clockwise or counterclockwise. Finally, we adapted the **Properties** window from PMKS into WPF. The team decided that this was the best way to display all of the options available to the user without cluttering the upper menu.

The **Properties** window is shown in Figure 9.19. below.

Unit Properties Set Units: Metric ▾

Length: Centimeter (cm)
Mass: Gram (g)
Force: Newton (kg·m/s ²)
Density: Grams per Cubic Centimeter (g/cm ³)
Pressure: Megapascal (MPa)

Material Properties Custom Material

Link Material: Aluminum 6061 ▾
Density: 2.7 g/cm ³
Yield Strength: 276 MPa
Pin Material: Carbon Steel 1065 ▾
Density: 7.85 g/cm ³
Yield Strength: 490 MPa

Link Dimensions Dimension Synthesis

Link Width: 1.905	cm
Link Depth: 0.635	cm
Pin Diameter: 0.635	cm

Cancel Save

Figure 9.19.: The Properties window from PMKS adapted into PMKS+D.

9.4.4. Help

The **Help** tab contains controls that allow the user to access help materials. Figure 9.20. below displays the new **Help** tab in PMKS+D.



Figure 9.20.: The Help tab in PMKS+D.

The **Help** tab contains the following controls:

- **About:** A `Button` used to open a window displaying information about PMKS+D;
- **Tutorial:** A `Button` used to open a window displaying a step-by-step tutorial of how to create a 4-bar linkage;
- **Help:** A `Button` used to display help information for different aspects of PMKS+D;
- **Report Issue:** A `Button` used to open a browser window to a Google Forms survey, which allows the user to report issues with PMKS+D;
- **Linkages:** A `Button` used to open a window that allows the user to open a linkage configuration from a set of example linkages.

The **Help** tab is new to PMKS+D. With the removal of the **Get URL** tab and the desire to add more help materials, the team felt that it was appropriate to move the **About**, **Help**, and **Report** buttons into a new tab. The team also changed the **Report** button to a **Report Issue**

button because this made the purpose of the button more clear. With the **Help** tab created, the team decided to add the **Tutorial** and the **Linkages** buttons.

The idea to add a step-by-step tutorial was inspired by the lack of a tutorial within the other linkage simulation applications explored. For a new user, it is difficult to learn new applications, and a tutorial provides a way to learn about the functionality of the application in a guided environment.

The idea to add sample linkages to the application was similarly inspired by the presence of sample linkages in the linkage simulation applications we explored. The team found it useful to be able to see common linkage examples recreated in the application's visual style, which allowed us to relate our real world experience to the application.

Figure 9.21. below displays the **Tutorial** window within PMKS+D. It is a document containing step-by-step instructions with images so the user can be introduced to the functionality, visual style, and terminology used by the application.

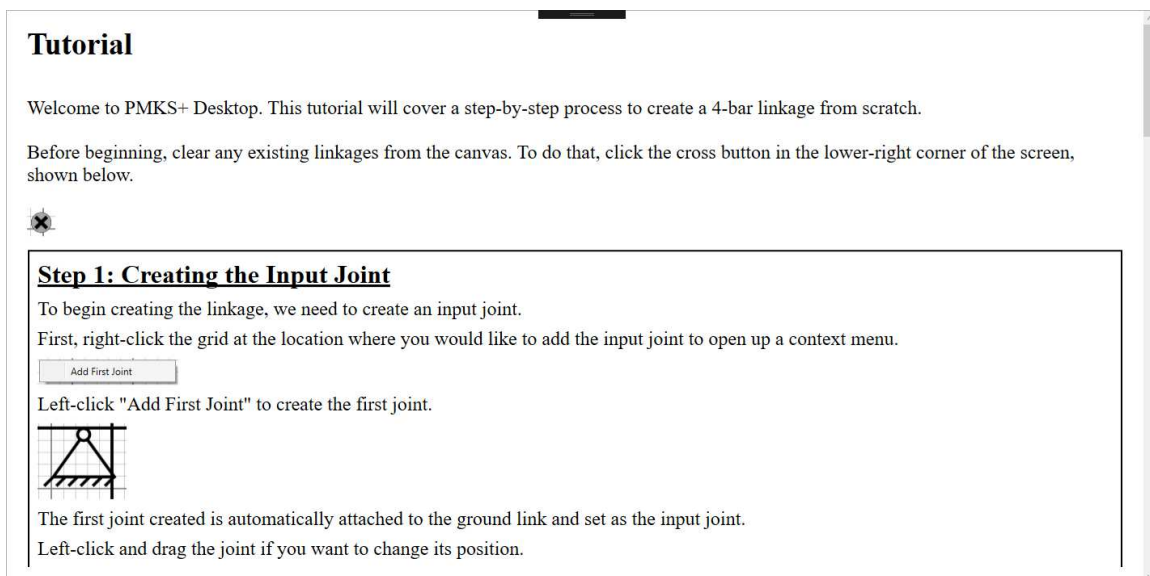


Figure 9.21.: The Tutorial window in PMKS+D.

The **Help** button had its functionality updated from PMKS+. Previously, the **Help** button would open a document containing help information for the entire application. The team felt that this style of help was overwhelming in its size and too difficult to find the information about the desired topics. To solve this issue, the team split the help text into multiple documents grouped by functionality and created an intermediate screen where the user selects which document they would like to open. This screen is shown in Figure 9.22. below.

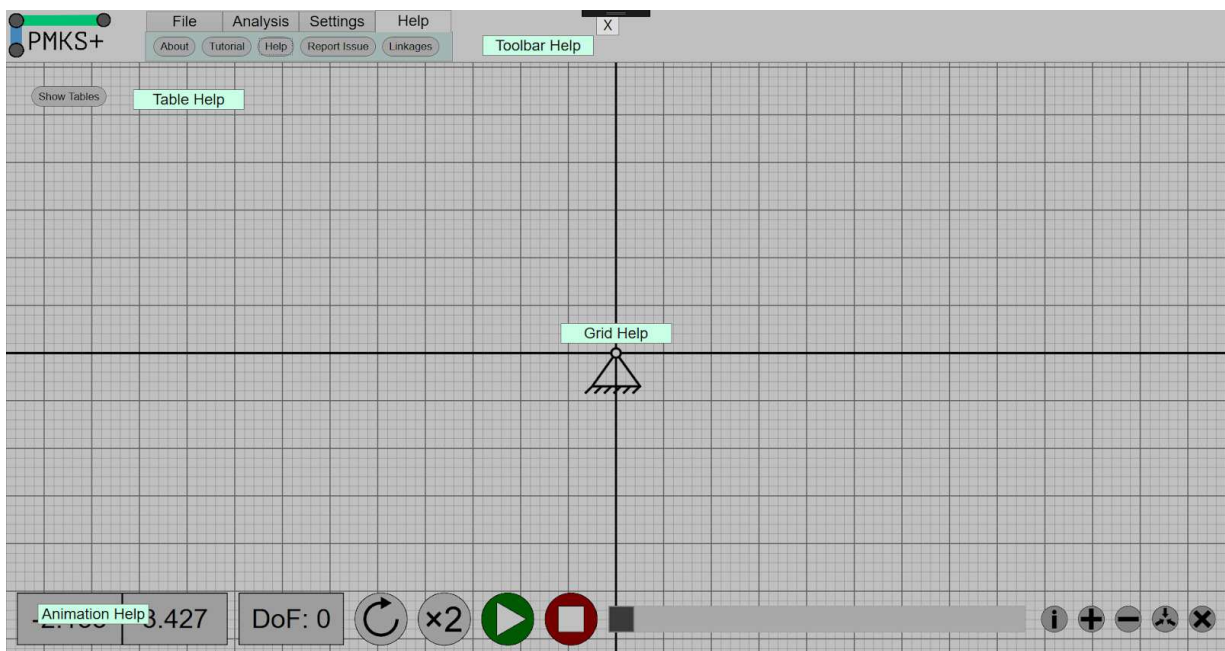


Figure 9.22.: Intermediate screen used to select the wanted help document.

The team added a button for each help document: **Toolbar Help**, **Table Help**, **Grid Help**, and **Animation Help**. Then, the background UI was darkened to indicate that input is not allowed while the intermediate screen is shown and to draw attention to the new buttons. Finally, a close button with an X was added to allow the user to exit the intermediate screen.

If a button is clicked, the appropriate help document is opened. The help documents contain text that explain the purpose of each UI element and pictures that correspond with the text. For example, Figure 9.23. below displays the **Toolbar Help** window.

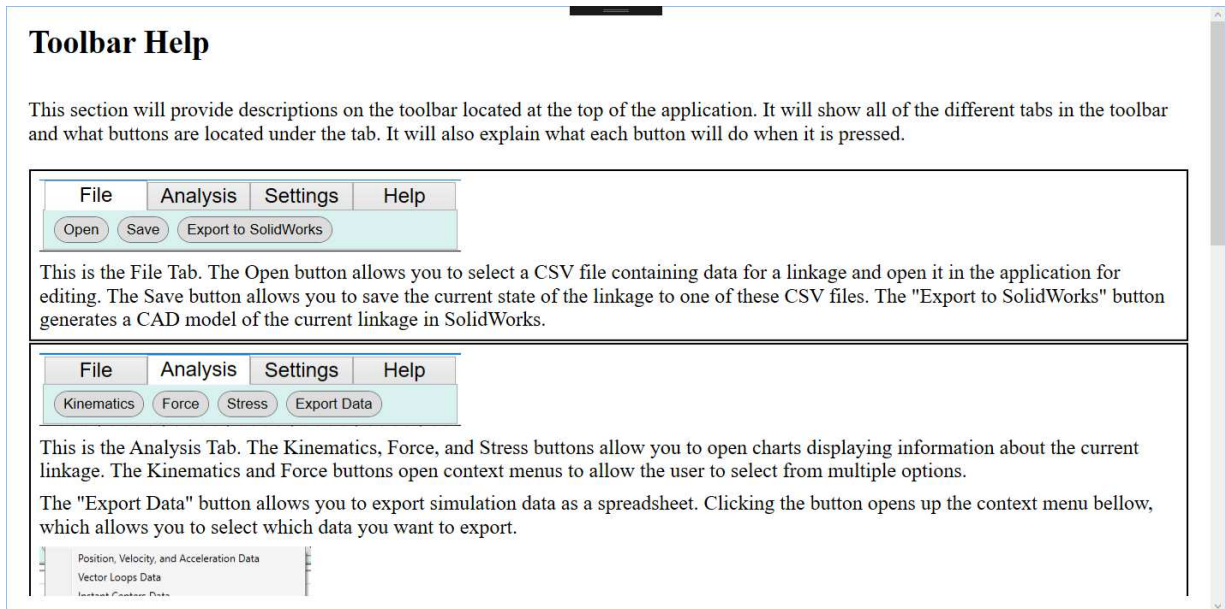


Figure 9.23.: The Toolbar Help window in PMKS+D.

With the design of these major sections of the interface complete, the team combined everything into a complete model of the interface. The final iteration of the PMKS+D UI is shown in Figure 9.24. below.

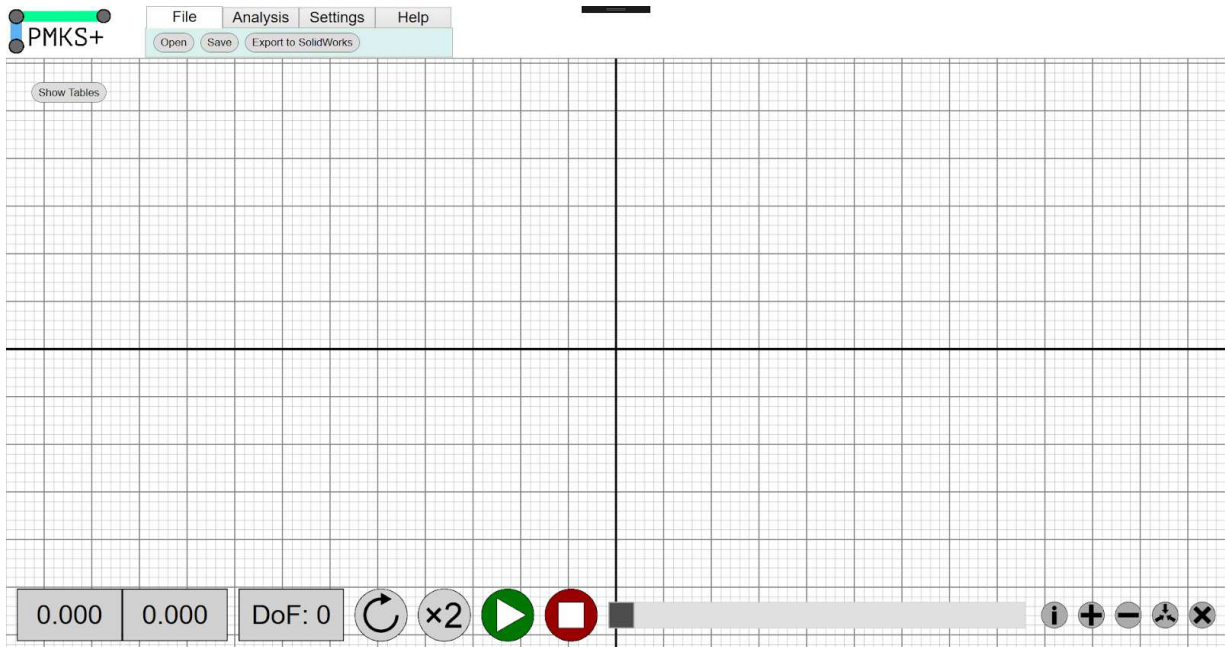


Figure 9.24.: The final iteration of PMKS+D's UI.

9.5. Applying UI Evaluation Metrics

To evaluate the success of PMKS+D's UI design, our team calculated the values for each UI evaluation metrics on different components of our interface. These metrics are described in Chapter 5.2.2.: Additional Metrics. The findings of this chapter are compared to the analyzed results of the user evaluation in Chapter 13.3.: Evaluation of Analysis.

9.5.1. Essential Efficiency

The first metric that the team calculated was the EE of different tasks within PMKS+D. These tasks are: creating a 4-bar linkage using the mouse-based creation method, opening an

angular velocity analysis graph, and exporting a linkage to SolidWorks. These tasks were chosen because the team felt these tasks were the most important and exciting features of our system.

Creating a 4-Bar Linkage:

The **essential steps** for creating a 4-bar linkage were determined to be the following:

1. Create an input joint in the desired location
2. Create a bar link attached to the input joint in the desired location
3. Create a bar link attached to the previous link in the desired location
4. Create a bar link attached to the previous link in the desired location
5. Designate the final joint as a ground joint.

The **required steps** for creating a 4-bar linkage within PMKS+D are:

1. Right-click to open Canvas context menu
2. Create an input joint
3. Drag the joint to the desired location
4. Right-click the joint to open a Joint context menu
5. Add a link
6. Drag the new joint to the desired location
7. Right-click the joint to open a Joint context menu
8. Add a link
9. Drag the new joint to the desired location
10. Right-click the joint to open a Joint context menu

11. Add a link
12. Drag the new joint to the desired location
13. Right-click the joint to open a Joint context menu
14. Toggle the joint to a ground joint

Using the formula for EE given in Chapter 5.2.2.: Additional Metrics, the EE of creating a 4-bar link is 36%. This result suggests that the mouse-based linkage creation method that our team adapted from PMKS+ is not as efficient as possible. The context menus especially appear to be reducing the EE, as it requires the user to repeatedly right-click and select an option.

Opening an Angular Velocity Analysis Graph:

The **essential steps** for opening an angular velocity graph were determined to be the following:

1. Open a graphing window
2. Select angular velocity as the type of graph
3. Select the component you want to graph

There are two methods for opening an angular velocity graph in PMKS+D. The first uses the **Analysis** tab, and the second uses the context menus.

The **required steps** for opening an angular velocity graph within PMKS+D using the Analysis tab are:

1. Click the Analysis tab
2. Click the Kinematics button
3. Click the Position, Velocity, and Acceleration option of the context menu
4. Select angular velocity as the type of graph from the dropdown menu
5. Select the component you want to graph from the dropdown menu

The **required steps** for opening an angular velocity graph within PMKS+D using the context menus are:

1. Right-click the desired component
2. Click the Show... Kinematics option of the context menu
3. Select angular velocity as the type of graph from the dropdown menu

Using the formula for EE given in Chapter 5.2.2.: Additional Metrics, the EE of the Analysis tab method is 60% and the EE of the context menu method is 100%. This result suggests that the context menu method is as efficient as possible, which is an interesting discovery. This result suggests that context menus reduce efficiency in tasks that require repetition but increase efficiency in tasks that do not. It appears that the Analysis tab method could be made more efficient by not requiring the user to switch to the Analysis tab. This could be achieved by removing the tab menu from the upper menu. Instead, buttons could run across the entire upper menu, with titles to designate subsections.

Exporting a Linkage to SolidWorks:

The **essential steps** for exporting a linkage to SolidWorks were determined to be the following:

1. Select a directory to save files within
2. Begin the export process

The **required steps** for exporting a linkage to SolidWorks within PMKS+D are:

1. Click the File tab
2. Click the Export to SolidWorks button
3. Click the Yes button to confirm export
4. Select the directory where the SolidWorks template files are located
5. Select a directory to save files within

Using the formula for EE given in Chapter 5.2.2.: Additional Metrics, the EE of exporting to SolidWorks is 40%.

First of all, the EE could be improved by removing the need to confirm the export. The reason the confirmation exists is to warn the user of the types of linkages that are not supported by the export process. If these limitations were removed, the confirmation window would become unnecessary.

Secondly, the EE could be improved if the user did not have to select the location of the SolidWorks template files. The team was not successful at including these files within the

executable, but finding another method for removing this requirement in the future would be beneficial.

9.5.2. Task Visibility

The second metric that the team calculated was the TV of the tasks from the previous section within PMKS+D.

Creating a 4-Bar Linkage:

In Table 9.1. below, the visibility rating for each required step of creating a 4-bar linkage is given, as well as a short justification for the rating.

Table 9.1.: Visibility and reasoning for each task of creating a 4-bar linkage in PMKS+D.

Task #:	Visibility:	Reasoning:
1	0.25	Context menu is invisible until the user right-clicks, but a right-click is a common method to bring up context menus in desktop applications
2	1	The correct option in the context menu is immediately apparent
3	0.6	It is not explicit that click-and-drag is required, but click-and-drag is a commonly-used method to move things in desktop applications
4	0.5	Context menu is invisible until the user right-clicks, but it is feasible to assume that right-clicking a joint will also bring up a context menu
5	1	The correct option in the context menu is immediately apparent
6	1	Click-and-drag on a joint has been used to move a joint previously
7	1	A joint context menu has been used to add a link previously
8	1	The correct option in the context menu is immediately apparent
9	1	Click-and-drag on a joint has been used to move a joint previously
10	1	A joint context menu has been used to add a link previously
11	1	The correct option in the context menu is immediately apparent
12	1	Click-and-drag on a joint has been used to move a joint previously
13	0.6	Context menu is invisible until the user right-clicks, but it is feasible to assume that the user might have seen the “toggle ground” option in the context menu previously.
14	1	The correct option in the context menu is immediately apparent

Using the formula for TV given in Chapter 5.2.2.: Additional Metrics, the TV of creating a 4-bar linkage is 85%. This is a pretty good result, but users may have an issue with figuring out

how to open context menus or how to move joints. This might be improved through somehow indicating that opening a context menu and dragging joints are possible on UI.

Opening an Angular Velocity Analysis Graph:

In Table 9.2. below, the visibility rating for each required step of opening an analysis graph using the Analysis tab is given, as well as a short justification for the rating.

Table 9.2.: Visibility and reasoning for each task of opening an angular velocity graph in PMKS+D using the Analysis tab method.

Task #:	Visibility:	Reasoning:
1	0.65	Analysis tab is on the top edge of the screen, but is in a large font. Relevance to graphing is not stated, but is not difficult to assume.
2	0.7	Kinematics button is directly below the Analysis tab. Angular velocity is not stated, but angular velocity is a form of kinematic analysis.
3	0.9	Context menu appears directly below the Kinematics button. Angular velocity is not stated, but is indicated from the word “velocity”
4	0.8	Angular velocity is not listed as an option for graph type until the dropdown menu is clicked, but is obvious once the user does so
5	0.8	The desired component might not be listed as an option until the dropdown menu is clicked, but is obvious once the user does so

In Table 9.3. below, the visibility rating for each required step of opening an analysis graph using the context menu is given, as well as a short justification for the rating.

Table 9.3.: Visibility and reasoning for each task of opening an angular velocity graph in PMKS+D using the context menu method.

Task #:	Visibility:	Reasoning:
1	0.5	Context menu is invisible until the user right-clicks, but it is feasible to assume that right-clicking a joint/link will also bring up a context menu
2	0.7	Show... Kinematics option is directly where the user right-clicked. Angular velocity is not stated, but angular velocity is a form of kinematic analysis.
3	0.8	Angular velocity is not listed as an option for graph type until the dropdown menu is clicked, but is obvious once the user does so

Using the formula for TV given in Chapter 5.2.2.: Additional Metrics, the TV of opening an analysis graph is 77% for the **Analysis** tab method and 67% for the context menu method.

First of all, it appears that the tab menu reduces TV. This again suggests that the tab menu should be replaced, as described in the previous section, as this would improve both EE and TV.

Secondly, this suggests that context menus provide a tradeoff between TV and EE. Context menus may be harmful for TV due to being invisible before the user opens one, but are better for EE because it allows the program to preload certain information, such as which component the user would like to graph.

Exporting a Linkage to SolidWorks:

In Table 9.4. below, the visibility rating for each required step of exporting a linkage to SolidWorks is given, as well as a short justification for the rating.

Table 9.4.: Visibility and reasoning for each task of exporting a linkage to SolidWorks in PMKS+D.

Task #:	Visibility:	Reasoning:
1	0.45	File tab is on the top edge of the screen, but is in a large font. Relevance to SolidWorks is not stated
2	0.95	Export to SolidWorks button is directly below the File tab.
3	0.85	Yes button is located below some text, but is large and obviously the correct option
4	0.7	Directory where the SolidWorks template files are located may not be initially visible, but the user should have previous experience using a File Explorer window.
5	0.9	The user should have previous experience using a File Explorer window to save a file.

Using the formula for TV given in Chapter 5.2.2.: Additional Metrics, the TV of exporting a linkage to SolidWorks is 77%. This result once again suggests that the tab menu should be replaced, as described in the previous section, as this would improve both EE and TV. This result also suggests that the need to confirm the export should be removed, as well as the requirement to select the location of the SolidWorks template files. Both of these improvements would improve the EE and TV of the SolidWorks export, and should be considered.

9.5.3. Visual Coherence

The third metric that the team calculated was the VC of the PMKS+D UI. The UI groups used to calculate the VC were the three tables, the animation control menu, the Canvas control menu, and the three tabs. These are the most important groups within the UI besides the Canvas. The Canvas was not evaluated due to the relatedness of the Canvas being affected by the size and complexity of the linkage on screen.

In Table 9.5. below, the relatedness of each pair of UI elements within the Joints table is given, as determined by the process outlined in Chapter 5.2.2.: Additional Metrics. The Joints table was chosen because it is the default table displayed. However, the relatedness scores for the Links and Forces tables are identical.

Table 9.5.: Relatedness of each pair of UI elements within the Joints table.

	Show/Hide Tables Button	Joints Tab Button	Data Table
Show/Hide Tables Button	X	1	1
Joints Tab Button	1	X	1
Data Table	1	1	X

In Table 9.6. below, the relatedness of each pair of UI elements within the animation control menu is given, as determined by the process outlined in Chapter 5.2.2.: Additional Metrics.

Table 9.6.: Relatedness of each pair of UI elements within the animation control menu.

	DoF Label	Input Direction Button	Input Speed Button	Play/Pause Button	Stop Button	Timeline Slider
DoF Label	X	0	0	1	0	0
Input Direction Button	0	X	1	1	0	1
Input Speed Button	0	1	X	1	0	1
Play/Pause Button	1	1	1	X	1	1
Stop Button	0	0	0	1	X	1
Timeline Slider	0	1	1	1	1	X

In Table 9.7. below, the relatedness of each pair of UI elements within the Canvas control menu is given, as determined by the process outlined in Chapter 5.2.2.: Additional Metrics. It is worth noting that the relatedness scores for this menu appears to be low. It is unclear how to fix this, besides moving the Display IDs and Clear buttons to a more appropriate location.

Table 9.7.: Relatedness of each pair of UI elements within the Canvas control menu.

	Display IDs Button	Zoom In Button	Zoom Out Button	Center Button	Clear Button
Display IDs Button	X	0	0	0	0
Zoom In Button	0	X	1	1	0
Zoom Out Button	0	1	X	1	0
Center Button	0	1	1	X	0
Clear Button	0	0	0	0	X

In Table 9.8. below, the relatedness of each pair of UI elements within the File tab is given, as determined by the process outlined in Chapter 5.2.2.: Additional Metrics.

Table 9.8.: Relatedness of each pair of UI elements within the File tab.

	File Tab Button	Open Button	Save Button	Export to SolidWorks Button
File Tab Button	X	1	1	1
Open Button	1	X	1	0
Save Button	1	1	X	0
Export to SolidWorks Button	1	0	0	X

In Table 9.9. below, the relatedness of each pair of UI elements within the **Analysis** tab is given, as determined by the process outlined in Chapter 5.2.2.: Additional Metrics.

Table 9.9.: Relatedness of each pair of UI elements within the Analysis tab.

	Analysis Tab Button	Kinematics Button	Force Button	Stress Button	Export Data Button
Analysis Tab Button	X	1	1	1	1
Kinematics Button	1	X	1	1	1
Force Button	1	1	X	1	1
Stress Button	1	1	1	X	1
Export Data Button	1	1	1	1	X

In Table 9.10. below, the relatedness of each pair of UI elements within the **Settings** tab is given, as determined by the process outlined in Chapter 5.2.2.: Additional Metrics.

Table 9.10.: Relatedness of each pair of UI elements within the Settings tab.

	Settings Tab Button	Input Speed Label	Input Speed TextBox	CW RadioButton	CCW RadioButton	Properties Button
Settings Tab Button	X	1	1	1	1	1
Input Speed Label	1	X	1	0	0	0
Input Speed TextBox	1	1	X	1	1	0
CW RadioButton	1	1	1	X	1	0
CCW RadioButton	1	1	1	1	X	0
Properties Button	1	0	0	0	0	X

In Table 9.11. below, the relatedness of each pair of UI elements within the **Help** tab is given, as determined by the process outlined in Chapter 5.2.2.: Additional Metrics. It is worth noting that the relatedness scores for the **Help** tab appears to be quite low. It is unclear how to fix this, other than perhaps moving the **Linkages** button to the **File** tab, as its purpose seems to relate more closely to opening a linkage configuration rather than help information.

Table 9.11.: Relatedness of each pair of UI elements within the Help tab.

	Help Tab Button	About Button	Tutorial Button	Help Button	Report Issue Button	Linkages Button
Settings Tab Button	X	1	1	1	1	1
About Button	1	X	0	0	0	0
Tutorial Button	1	0	X	1	0	0
Help Button	1	0	1	X	0	0
Report Issue Button	1	0	0	0	X	0
Linkages Button	1	0	0	0	0	X

With the relatedness scores determined for each UI group, the VC of the entire interface could be calculated using the formula provided in Chapter 5.2.2.: Additional Metrics. The VC of the entire PMKS+D UI is 63%. This result suggests that the relatedness of the UI groups could be improved somewhat. It seems that smaller UI subgroups need to be created to separate UI objects that are dissimilar.

10. System Design

In addition to designing the UI of PMKS+D, the team needed to design the internals of the system to implement the functionality of PMKS while allowing for new functionality. This chapter describes aspects of the system design that the team implemented after our coding explorations in Chapter 7.: WPF Coding Explorations.

10.1. Model-View-ViewModel Design Pattern

To begin the design of PMKS+D, the team decided to follow an established design pattern to keep our code organized and efficient. The design pattern officially supported by Microsoft in WPF is the Model-View-ViewModel design pattern. This design pattern has three features:

- the Model, which stores information about the application's state;
- the View, which defines the UI displayed to the user;
- the ViewModel, which implements the program, handles interactions with the View, and updates the Model (Microsoft Corporation, 2017).

In WPF, the View is stored within XAML files, which contain declarations for UI elements and fields that the ViewModel interacts with. The design of the View is explored in Chapter 9: Interface Design. The Model and ViewModel classes are contained in C# files and define properties and methods that control the flow of the program and storage of information.

The design of the Model is explored in Chapter 10.2.: Model, and the design of the ViewModel is explored in Chapter 10.3.: MainWindow.

10.2. Model

The model is a set of classes that store information about the components of a linkage so that the information can be passed into the simulator, edited by the user, and displayed on screen. The team created abstract classes for joints and links, subclasses for different kinds of joints and links, and a class for forces. These classes contain properties and functions that are specific to each component and allow the program to track, edit, and draw components of the linkage. More information about the visual design of the joint, link, and force graphics can be found in Chapter 9.2.1.: Graphics.

10.2.1. Joints

The team created an abstract class called **Joint** to represent a generic joint. The following is a list of the properties of the `Joint` class that will be inherited by all subclasses.

Joint Abstract Class:

- **InitialX:** the initial x-position of the Joint.
- **InitialY:** the initial y-position of the Joint.
- **Links:** the list of Links that the Joint is attached to.
- **IsGrounded:** a boolean whether the `Joint` is grounded or not.
- **JointType:** an enum that stores the type of `Joint` (Revolute or Prismatic).

- **ID**: a string that is the ID of the `Joint`.
- **IsInput**: a boolean of whether the `Joint` is the input `Joint`.
- **HitboxEllipse**: an `Ellipse` instance representing the hitbox of the `Joint`.
- **MainWindow**: value to store the state of the `MainWindow`.
- **IsSelected**: a boolean storing if the hitbox is selected or not.
- **Angle**: the angle in degrees of the `Joint`'s movement, if prismatic.
- **HideOrShowPath**: boolean storing if the `Joint`'s movement path is hidden or not.

The abstract class is inherited by two subclasses: `RevoluteJoint` and `PrismaticJoint`. Both contain unique methods for drawing the joint as well as additional functions to aid the calculations of the position.

The **`RevoluteJoint`** subclass gathers information directly from the simulator for animation. This is because the graphic for a revolute joint does not differ from the original PMKS version.

`PrismaticJoint` is used specifically for sliders, and uses data from the simulator in a unique way relative to the other methods of drawing. The simulator from the Silverlight version had a different method to display the link where the joint is stationary and a slider is a large open slot that is the length of the range of motion. An example of a linkage with a slider in PMKS is shown in Figure 10.1. below.

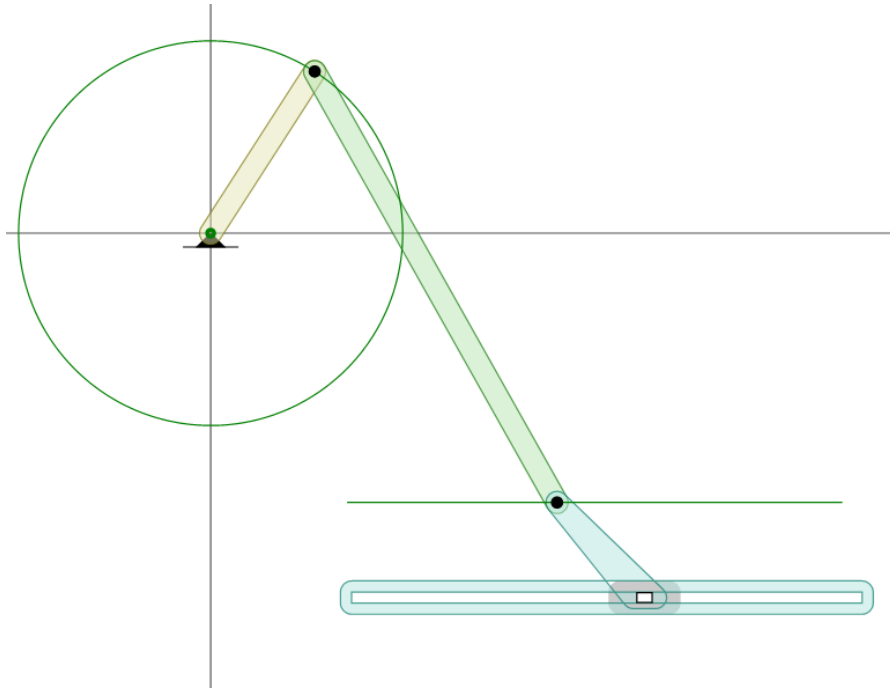


Figure 10.1.: A linkage with a slider in PMKS.

In order to create a new method of presenting the joint, the display of the joints needed to change. The `DrawJoint` method uniquely takes the position of the other joints on the current joint's link and calculates the difference in position to display a prismatic joint that appears to slide on a stationary track. Also, similar to PMKS, the track needed to display the limits of motion. The limits are determined by offsetting the extreme positions of the other joints on the current joint's link. These limits and tracks make up the unique ground graphic for a prismatic joint.

10.2.2. Links

The **Link** abstract class consists of several properties to outline the parameters of a link.

The following is a list of these properties that apply to all links in the model.

Link Abstract Class:

- **Joints**: a list of `Joints` on the `Link`.
- **IsGround**: a boolean to determine if the `Link` is the ground `Link`.
- **ID**: a numerical value that is the ID of the `Link`.
- **IsInput**: a boolean value if the `Link` is the input `Link`.
- **HitboxShape**: a `Shape` for a hitbox that pertains to the `Link`.
- **MainWindow**: a value to store the current state of the `MainWindow`.
- **Forces**: a list of `Forces` attached to the `Link`.
- **IsSelected**: a boolean to determine the transparency of the hitbox

Similar to the `Joint` abstract class, there are two subclasses that inherit `Link`:

`BarLink` and `PolyLink`.

The **BarLink** represents the basic form of a link and includes only two `Joints` in the `Joints` property. This model representation gathers information on the two `Joints`' positions to draw a rounded bar graphic. This subclass also contains a function to draw the `HitboxShape` as a `Rectangle` with the width and length of the `BarLink` graphic.

The **PolyLink** subclass represents a link with 3 or more `Joints` attached to it. When creating the graphic for the `PolyLink`, the positions of each joint are used to create a rounded

polygonal shape as opposed to a rounded bar. The `HitboxShape` is also set to a `Polygon` connecting each `Joint`.

10.2.3. Forces

Finally, there is a single **Force** class because there is currently only one way to graphically display a force. The following is a list of the `Force` properties

- `InitialX`: the initial x-position of the `Force`'s point of application.
- `InitialY`: the initial y-position of the `Force`'s point of application.
- `MagnitudeX`: the magnitude of the `Force` in the x direction.
- `MagnitudeY`: the magnitude of the `Force` in the y direction.
- `Magnitude`: the scalar total magnitude of the `Force`.
- `IsRelativeToLink`: boolean storing whether the `Force`'s angle is relative to its `Link` (local) or not (global).
- `ID`: a string storing the `Force`'s ID.
- `OnLink`: the `Link` that the `Force` is on.
- `MainWindow`: the `MainWindow`'s current state.
- `HitboxEllipseInitial`: the `Ellipse` for the hitbox of the `Force`'s point of application.
- `HitboxEllipseMagnitude`: the `Ellipse` for the hitbox at the `Force`'s head.
- `IsInitialSelected`: a boolean storing if the initial hitbox is selected.
- `IsMagnitudeSelected`: a boolean storing if the magnitude hitbox is selected.
- `Angle`: the angle of the `Force`'s magnitude.
- `LinkAngle`: the angle of the `Force`'s magnitude relative to its `Link`'s angle.

In order to use all of these properties, the `Force` class also contains several functions specifically related to forces. The force information is gathered directly from the simulator for displaying the animation. The team created several trigonometric functions that perform the necessary calculations to display the force. Depending on `IsRelativeToLink`, the `Force` is drawn as either a local force that has a constant angle relative to its `Link` or a global force that has a constant angle relative to the origin.

Figure 10.2. below contains a simplified class diagram of the Model.

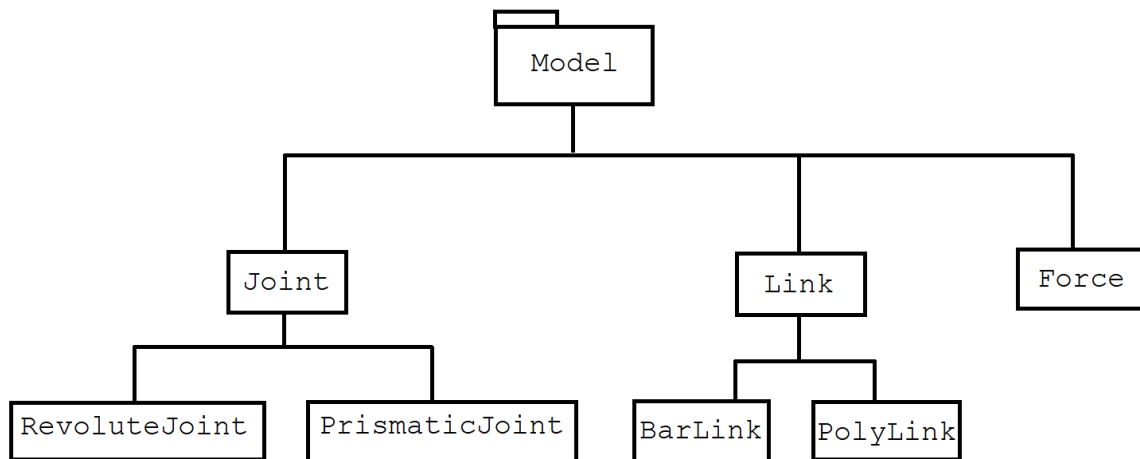


Figure 10.2.: Simplified class diagram of PMKS+D's Model.

10.2.4. Draw Linkage

In order to keep the old simulator functionality but maintain the new aesthetic and additional features, the team implemented the `Draw` functions within the `Model` classes. In the `MainWindow`, a function runs through the list of `Joints`, `Links`, and `Forces` currently stored in the `Model` and calls each `Draw` method to display each component. This also had the unique benefit of allowing flexibility of which components to draw first so lower priority components, such as ground graphics and path lines, are added first and higher priority components, such as joints and links, are drawn last.

10.3. MainWindow

`MainWindow.xaml.cs` is the main class where most of the application functionality is placed and represents the `ViewModel` of the program. `MainWindow` initializes the program, renders animations, and handles events. The `MainWindow` contains properties and event handlers, and updates the simulator, draws the linkage, and opens and saves files.

10.3.1. Properties

The `MainWindow`, because of its control over the program's execution, needs to store important information about the state of the program. In WPF, a member of a `Class` that stores data is known as a `Property`. Table 10.1. below contains the especially important properties and a description of their purpose:

Table 10.1.: Important properties within MainWindow.

Name:	Class:	Description:
PMKSD	Simulator	Stores the current state of the Simulator, which performs the simulation.
IsModelReady	bool	True if the Model is ready to be passed to the simulator, false otherwise.
Joints	ObservableCollection <Model.Joint>	Stores the Joints that make up the current linkage.
Links	ObservableCollection <Model.Link>	Stores the Links that make up the current linkage.
Forces	ObservableCollection <Model.Force>	Stores the Forces that are applied to the current linkage.
Index	int	The current index into the array of positions within the simulator. Updated to control the linkage' animation. Used by the Model classes to draw the linkage.
RPM	InputSpeed	Used to change the speed of the linkage's animation.
RefreshRate	double	The calculated refresh rate of the computer's monitor. Used to animate the linkage at the correct RPM.
InputRotation	RotationDirection	Used to change the rotation direction of the linkage's animation.
IsMovementPaused	bool	Used to pause or unpause the animation.
LinkWidth	double	Set the displayed width of a Link graphic.
JointRadius	double	Set the displayed radius of a Joint graphic.
SliderJointWidth	double	Set the displayed width of a PrismaticJoint slider graphic.
SliderJointHeight	double	Set the displayed height of a PrismaticJoint slider graphic.
Scale	double	The visual scale of the Canvas. Used to scale the Canvas while zooming.
LinkagePath	Path	The movement paths of the linkage's Joints.

10.3.2. Event Handlers

The event handlers make up the largest section of the `MainWindow`. Event handlers handle mouse events, button click events, and context menus. Using all of these techniques, the team was able to handle all user interactions that were registered from the `View` class `MainWindow.xaml`.

Mouse Events:

The mouse event handlers are separated into four parts: `MouseMove`, `MouseLeftButtonDown`, `MouseLeftButtonUp`, and `MouseScroll`. `MouseMove` aids in the detection of the hitboxes described earlier, as well as handling click-and-drag. Dragging a joint begins and ends with the `MouseLeftButtonDown` and the `MouseLeftButtonUp` event handlers, respectively. Finally, the `MouseScroll` is used to handle scaling the view.

Button Click Events:

The team implemented several unique `Clicked` events to handle the user's interaction with the buttons throughout the interface. These event handlers control a wealth of functionality, from saving or opening files to exporting to SolidWorks and controlling the grid and the linkage's animation. By dividing the functionality into discrete event handlers, the team was able to keep the structure of the program organized and consistent with the user interface.

Context Menus:

The remaining event handlers were implemented to process any context menu actions performed by the user. The context menus on the grid allow the user to manipulate the existing model and therefore change the linkage's visual representation on the interface. Other context menu options allow the user to open new `Windows`. For example, the buttons in the `Analysis` tab open context menus with options for the types of graphs that can be opened.

10.3.3. Update Simulator

Another major component to the `MainWindow` system design was implementing a method to update the simulator based on the state of the `Model`. The `UpdateSimulator()` method handles two processes: transforming `Model` data into a form that the simulator can use, and inspecting the transformed data. `UpdateSimulator()` only allows data to be passed to the simulator when the `Model` contains the correct information.

First, the method takes the data from the `Joints`, `Links`, and `Forces` stored in the `Model` and organizes it into data structures that the simulator will accept. The simulator only accepts joint position information and implicitly defines the links based on the order of the link ID names that it receives for each joint.

Once the parsing is complete, the method further manipulates the link IDs. All links have integer IDs in the `Model` for user convenience. However, the simulator expects a link named "ground" and a link labeled "input". `UpdateSimulator()` corrects the link IDs to fit the ground and input links in the `Model` to the simulator's expectations.

The final process of updating the simulator is checking whether all the necessary components of the linkage are present. If all the components the simulator needs are not present, then the data is not passed to the simulator and a flag is set that informs the rest of the program that the simulator is not in a readable state.

10.3.4. Draw Linkage When Incomplete

Due to the fact that the simulator must have certain components of the linkage to run properly, the team needed to create methods of drawing the linkage when the simulator is incomplete. In the previous section, the team discussed the complications with passing information to the simulator. If the Model is not passable, the program will draw the primitive state of the linkage using the existing data in the Model. This function allows the linkage to still be displayed while system components are still incomplete.

10.3.5. Open and Save Linkage

In order to save the current linkage to a CSV file or open a saved linkage configuration into the program, it was necessary to create a way to communicate between the Model and the program's I/O stream. The team created a pair of methods: `ParseCSV` to handle opening a linkage configuration file and `SaveCSV` to handle saving the linkage configuration to a CSV file. The format for the CSV file was created by the PMKS+ team, and our team designed these methods to conform to that format.

Both methods begin by showing a `FileDialog` to allow the user to select the file they want to open, or name and choose the location of the file they want to save. Once the

FileDialog's FileOK event triggers, the program knows that the selected file or given file name is valid and proceeds onward.

ParseCSV calls the FileOpen method on the FileDialog to open the file as a Stream, which is then passed into a StreamReader. The StreamReader uses ReadLine to advance line-by-line through the CSV file.

The CSV file has headers to designate the fields that the program should check for. If the format of these headers does not match the expected format, then the program will display an error message and exit the method without changing the Model. For example, the code below is checking the header declaring the fields of joints.

```
if (text != "id,x,y,type,ground,input,angle")
{
    MessageBox.Show("Error: File is malformed.\n\n
                    Check the format of the file and
                    try again.");
    e.Cancel = true;
    return;
}
```

When a valid header is detected, all of the lines until the next header are treated as an instance of a Model class. For example, once the above header is validated, every line in the CSV file until the header for the links is treated as a new joint. Each line is split up by commas into an attributes[] array, and each entry of the array is fed into the TryParse method for the expected type. For example, the angle of a Joint is expected to be a double, so attributes[6] is passed into double.TryParse(). If all the attributes are successfully parsed, the attributes are passed into the constructor for a new Joint, Link, or Force and

added to a list of `Joints`, `Links`, or `Forces`, respectively. Otherwise, the program prints an error message and exits without updating the `Model`.

When the entire CSV file is parsed, the lists of `Joints`, `Links`, or `Forces` are optimized for the simulator and the `Joints`, `Links`, or `Forces` properties of the `MainWindow` are overwritten with the new lists and the simulator is updated.

`SaveCSV` is quite similar. First, it calls the `FileOpen` method on the `FileDialog` to open the file as a `Stream`. Then, a `string` is constructed section by section and line by line. First, the headers for a section are added. Then, depending on the section, each `Joint`, `Link`, or `Force` in the `Model` have its attributes added to a new line. Once all three sections are added to the `string`, the `string` is encoded into an ASCII-encoded array of bytes. This array is then passed into the `Write` method of the `Stream` to save the information to the file.

10.4. Additional Features

The system design of PMKS+D also needed to be able to incorporate new features that did not exist in PMKS or PMKS+. This section will describe our implementation of the two most important new features to be added to PMKS+D.

10.4.1. SolidWorks Export

The most exciting new feature to be added to PMKS+D is the ability to export a CAD model of the linkage into SolidWorks. As explained in Chapter 9.4.1: File, this feature improves the UX by greatly reducing the time necessary to create a model of the linkage. The program

arranges copies of provided SolidWorks template files to construct the linkage. In Figure 10.3. below, a linkage created in PMKS+D is compared to the CAD model generated in SolidWorks.

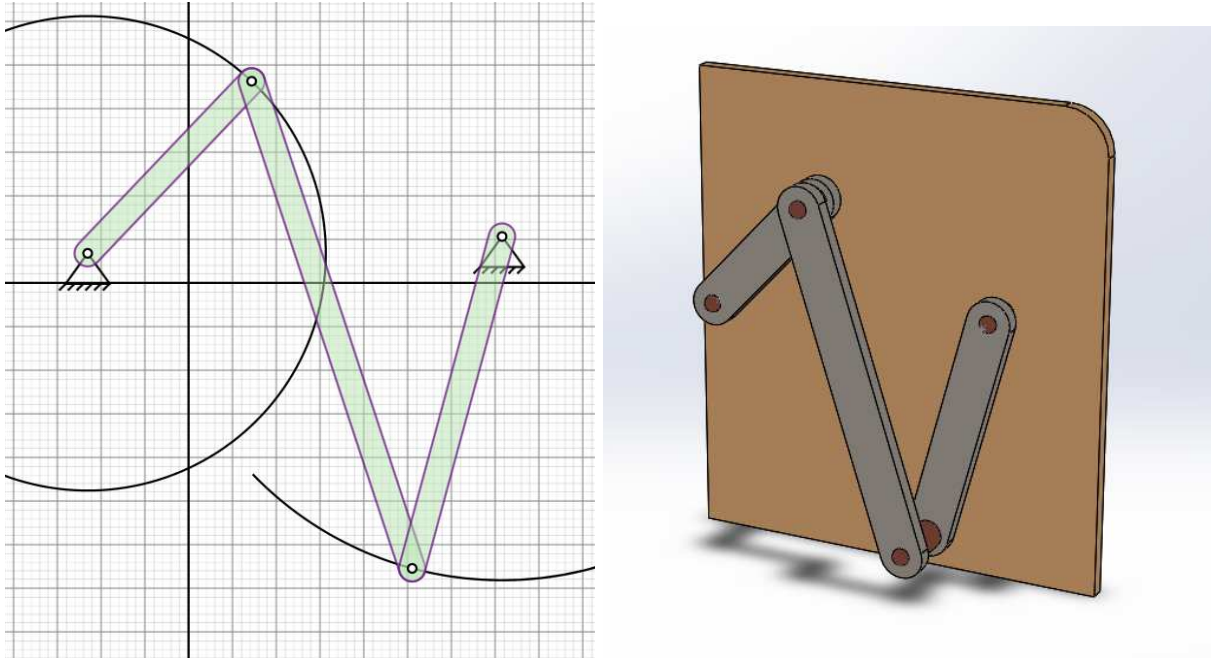


Figure 10.3.: 4-Bar linkage in PMKS+D and SolidWorks

While the SolidWorks export is quicker than manually creating the linkage, the process could take up to several minutes to complete. As a result, the team decided to implement the export process on a separate thread so that the user could continue to use PMKS+D while the export occurs.

When the export button is clicked, an event handler creates a new `Thread` to handle the calculations required for the export. This `Thread` opens a confirmation window with information about the requirements for the export process to succeed. If the user clicks `Yes` to

confirm that they would like the process to continue, an event handler calculates the vector loops of the current linkage. If this process succeeds, two folder browser windows appear in sequence.

The first asks the user to locate the directory where the SolidWorks template files are located. This step adds complexity to the export process, but the files were unable to be successfully included in the executable and time constraints prevented the team from finding another solution.

The second folder browser asks the user to select a directory where they would like the generated SolidWorks files to be saved. If the user selects a valid directory, the vector loops are passed into a `GenerateSolidWorks` method. This method implements an algorithm developed by Oluchukwu Okafor and adapted by Professor Radhakrishnan to convert the linkage into SolidWorks documents and combine them into a single assembly. Once complete, a window appears to notify the user that the export is finished and the `Thread` returns.

10.4.2. Graphing

The second most important new feature added to PMKS+D was the automatic graph generation. As explained in Chapter 9.4.2.: Analysis, the graphing feature improves the UX by automatically generating graphs of the analysis data for the linkage. More information about the visual design of these graphs can be found in that chapter.

When a user clicks a context menu to open a graph, the ID of the component clicked is compared to the IDs of the components in the Model. If the IDs match, the component is passed into a new window along with the analysis data from the simulator.

The minimum and maximum extents for the x and y axes are calculated, and the distance between tick marks is determined from the range. A `GeometryGroup` is created for each axis to draw a line between the extents and a line for each tick mark. A title and axis `Labels` are added to the graph based on the graph type selected and the type of component selected. Labels for the values for each tick mark are calculated and added as well. Then, a `Polyline` is created to connect each pair of x and y values in the dataset. Finally, the `GeometryGroups`, `Labels`, and `Polyline` are added to the `Canvas` to display the graph.

If the user changes the type of graph using the dropdown menu, an event handler repeats this process with the new dataset.

The user can also click on the graph to display the value of the clicked datapoint. A `MouseMove` event tracks which datapoint the user's mouse is over, if any, and a `MouseUp` event places an `Ellipse` over the datapoint's location and adds a `Label` displaying the x and y values of the datapoint.

Figure 10.4. below shows an example of the graph for the x velocity of a joint with a datapoint clicked.

Velocity X ▾ Export Selected Data Save Selected Data as Image Export All Data for this Joint

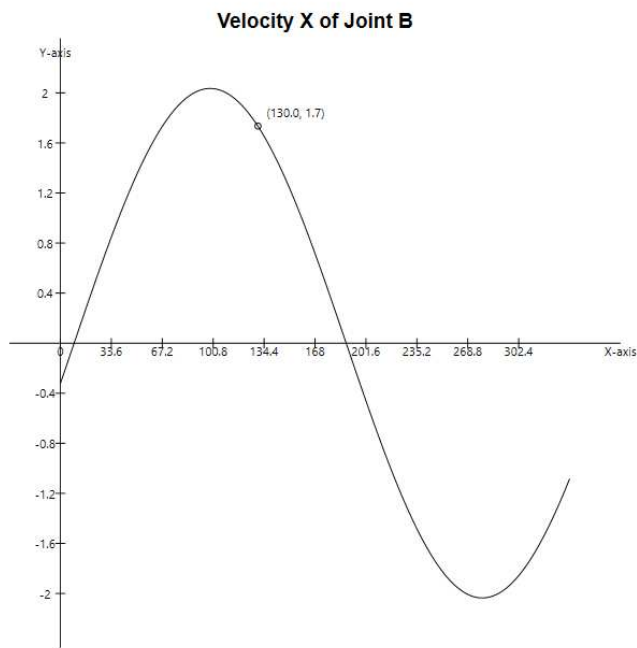


Figure 10.4.: The x velocity of a joint with a datapoint clicked.

11. Application Testing

In addition to the user testing, the program was carefully tested in a more controlled fashion by another Computer Science student, Robert Datile. They began by acquiring the program from the team and downloading it, followed by studying the help and tutorial documentation posted online by the team. Later versions of the program were acquired from the website that had been set up by the advisors to allow evaluators to access it (pmks.mech.website). Then, the tester collaborated with Professor Radhakrishnan, one of our advisors, to put together a clear list of what needed to be tested.

The testing process was performed in two 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 recreating several diagrams which we considered representative of common use cases for the program. These examples were created through discussion with Professor 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. Finally, we checked for discrepancies in the user interface between the desktop and web versions of PMKS+. This was to maintain ease of use and a unified visual style between the two. 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.

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. In order to ensure that performance was maintained even for a student who was forced to use the program remotely, tests were performed on executables in a WPI-based Windows remote desktop. This also simplified testing of the ability to export linkages to Solidworks, as that program is standard on WPI-provided remote desktops.

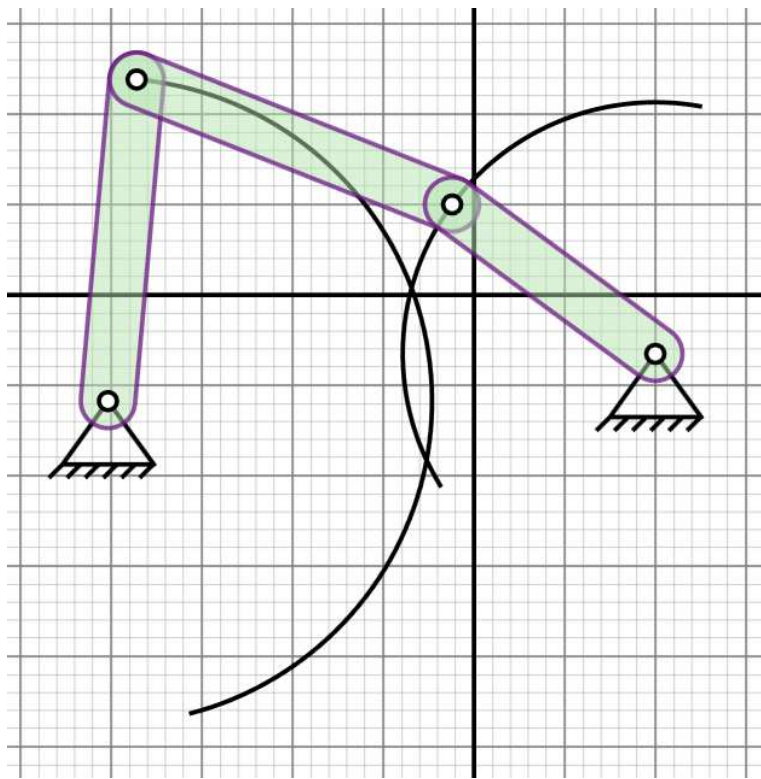


Figure 11.1.: 4-Bar revolute linkage in PMKS+D.

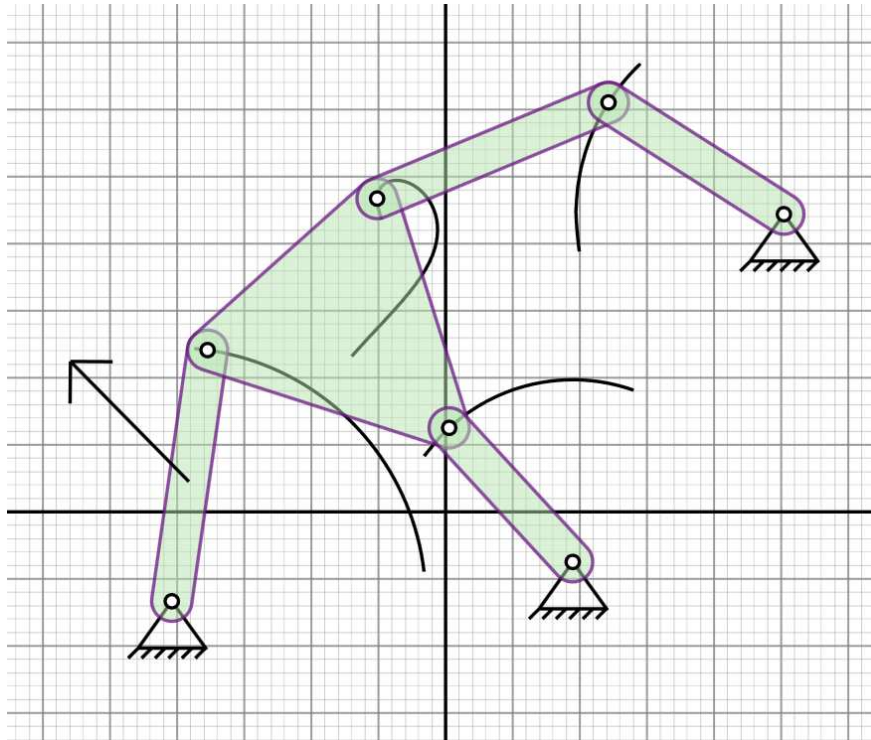


Figure 11.2.: 6-Bar linkage with a force in PMKS+D.

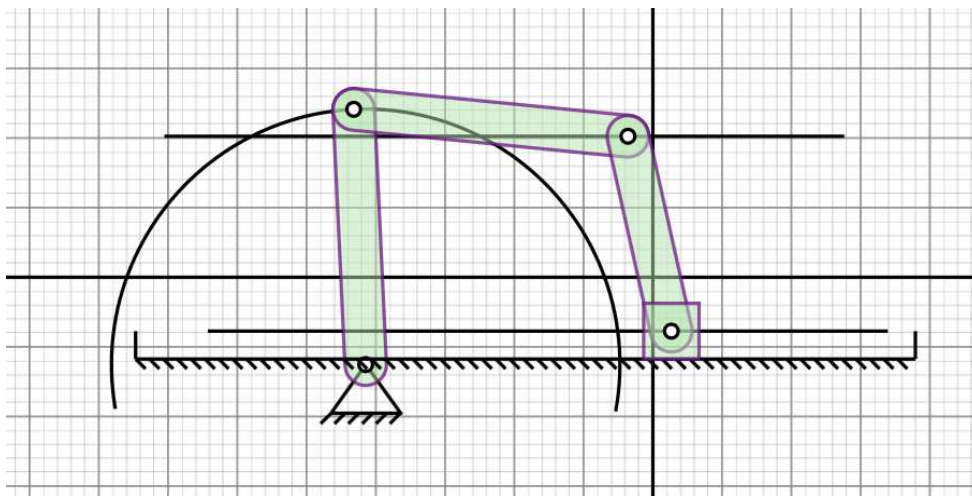


Figure 11.3.: 4-Bar slider crank linkage in PMKS+D.

11.1. Example Linkage Testing

This set of tests were performed by recreating and manipulating sample linkages. Figures 11.1., 11.2., and 11.3. above demonstrate the example linkages used to test the application.

<p>4-Bar Linkage (Corresponds to Figure 11.1.)</p>	<p><i>Creation:</i></p> <ul style="list-style-type: none"> ● A ground joint is placed at the origin on start-up ● Problem: Also, no perceptible signal for which is the input joint. ● Adding a link just creates a bar link with a joint at the other end of a default length and position relative to the starting joint ● Problem: Sometimes stops allowing the user to interact with the elements of the canvas until either it spontaneously starts working again or the user clears and loads a save. May be related to lag from the remote desktop. ● Need to create or start with an original joint created on the canvas, which is always a ground <p><i>Animation:</i> Works as expected</p> <p><i>Save:</i> Works as expected</p> <p><i>Reopen the saved file:</i> Works as expected</p> <p><i>Change the input/middle/output shapes:</i></p> <ul style="list-style-type: none"> ● Unlike in the Web version, there is no ability to change the shape of a link <p><i>Test movements:</i> Works as expected</p> <p><i>Check export to SolidWorks:</i></p> <ul style="list-style-type: none"> ● Problem: Appears to mostly work, but is somewhat time consuming, I don't think all of the parts were quite slotted together properly into the assembly, and the program closes automatically after performing it ● Update: seemed to work perfectly this time
<p>6-Bar Linkage with Force (Corresponds to Figure 11.2.)</p>	<p><i>Creation:</i></p> <ul style="list-style-type: none"> ● Problem: I was unable to create it, as the linkage repeatedly became completely unresponsive to right clicking immediately following the addition of the fourth link

<p>4-Bar Slider Crank Linkage (Corresponds to Figure 11.3.)</p>	<p><i>Creation:</i></p> <ul style="list-style-type: none"> ● Problem: I was unable to create one. <ul style="list-style-type: none"> ○ It looks like the DOF might not be calculating properly? Four bar slider crank displayed 3 degrees of freedom, so the sliding joint might not be reducing that properly ○ Update: It turns out the sliding joint needs to be toggled to ground before it calculates properly. Now otherwise works as expected. <p><i>Animation:</i> Works as expected</p> <p><i>Save:</i> Works as expected</p> <p><i>Reopen it:</i> Works as expected</p> <p><i>Test movements:</i></p> <ul style="list-style-type: none"> ● Mostly works as expected, but: <ul style="list-style-type: none"> ○ Problem: removing the initial ground joint causes the entire linkage to stop being interactable, and clearing does little to help. Forces the user to close the program outright. <p><i>Export to SolidWorks:</i></p> <ul style="list-style-type: none"> ● Works as expected, though it didn't really construct a track for the slider.
---	---

11.2. Individual UI Feature Testing

This set of tests focused on individual features in the user interface.

<p>General Examination of UI</p>	<p>Numerous slight differences from the web version, including:</p> <ul style="list-style-type: none"> ● Different coloration of buttons ● Different size and arrangement for the coordinate display ● Links have some transparency ● Tables have a have a different visual appearance (color and shape of tabs in particular) ● Joint table lacks action column ● Problem: When zooming in and out using the scroll wheel, the grid can seem to desynchronize with the X and Y axis ● Problem: There is no visual cue for which ground joint is the input
----------------------------------	---

Bottom Panel	<p><i>Clear Button:</i></p> <ul style="list-style-type: none"> ● Problem: Mostly works, but: <ul style="list-style-type: none"> ○ The DOF and joint table doesn't update until a new joint has been added. ○ After clearing the image of the first joint added snaps to the origin while it's interactivity remains in the position clicked. Fixes itself after adding a second joint <p><i>DoF Display:</i> Works as expected</p> <p><i>Coordinate Display:</i> Different layout than PMKS+ Web, but it works as expected</p> <p><i>Play/Pause:</i> Works as expected</p> <p><i>Stop:</i> Works as expected</p> <p><i>Reverse:</i> Works as expected</p> <p><i>Speed Change:</i></p> <ul style="list-style-type: none"> ● Problem: 1x speed is faster than 2x or 3x, 2x is the slowest <p><i>Info:</i></p> <ul style="list-style-type: none"> ● Functions like stop button, otherwise nothing. <p><i>Zoom Buttons:</i></p> <ul style="list-style-type: none"> ● Works as expected, as does scroll. Understandably laggy when working on a remote desktop. <p><i>Center:</i></p> <ul style="list-style-type: none"> ● Works, but: <ul style="list-style-type: none"> ○ Problem: does not always center on the origin, which was unexpected
Top Tab	<p><i>About:</i> Does nothing</p> <p><i>File tab Buttons:</i> All work as expected</p> <p><i>Analysis Tab Buttons:</i></p> <ul style="list-style-type: none"> ● Kinematics button works for export, static option on the force button also works, all other options and buttons in the Analysis tab do nothing.

	<p><i>Tutorial:</i> Works as expected</p> <p><i>Help:</i> Works as expected</p> <p><i>Report:</i> Works as expected</p> <p><i>Linkages:</i></p> <ul style="list-style-type: none"> ○ Non-functional, as are all other help tab buttons, apart from those previously mentioned. <p><i>Settings:</i></p> <ul style="list-style-type: none"> ● Settings offers properties and RPM input, which seem to work. ● Relative to Web version, help and settings tabs are switched
Adaptation to Different Resolutions	In sufficiently small resolutions, achieved by scrunching up the display window, the tables can obstruct the bottom bar buttons and the top tabs start to stack up weirdly, while the top buttons can just get lost outright.
Tables	<p><i>Check joint tables and edit/delete:</i></p> <ul style="list-style-type: none"> ● All columns work as expected, but there is no actions column <p><i>Update:</i></p> <ul style="list-style-type: none"> ● does not seem to update when the clear button is used <p><i>Check link tables – edit/delete:</i></p> <ul style="list-style-type: none"> ● Currently non-functional <p><i>Check force tables – edit/delete:</i></p> <ul style="list-style-type: none"> ● Same as joint table
Input Toggling	Problem: There does not appear to be any way to change the input joint.
Functionality in remote desktop	Largely functional, but does create noticeable lag
Are the right click options the same as for the Web version?	<p>Several discrepancies, including:</p> <ul style="list-style-type: none"> ● Only option when clicking on canvas is “Create Joint”. ● When clicking on a link, it offers “create joint”, “add force” or “show kinematic data” ● No option to delete a link without deleting a joint ● Ability to delete joints seems pretty inconsistent, varying from joint to joint. <ul style="list-style-type: none"> ○ Seems to only want you to be able to delete the most recently created joint.

	<ul style="list-style-type: none"> ● Has the hide or show position path option for all non grounded joints, which is useful if not entirely consistent with the web version. ● Implementation of joint control options seems either overly limited or incomplete: <ul style="list-style-type: none"> ○ Some joints get a full spread of options, some just get the previously mentioned option or show joint kinematics ○ If a ground joint is connected to another joint, the only option it offers is toggle grounded
--	--

11.3. Response to Application Testing

This section will involve the team responding to the results of the application testing. All issues encountered by the evaluator are an issue with the design and will be considered in the future, but this section will explore whether the issues discovered are deliberate design choices, bugs, or otherwise.

11.3.1. Example Linkage Testing

The lack of a graphic marking the input joint is unfortunately due to a lack of implementation time. The team noticed the need for this graphic before testing, and the evaluator noticing this makes it clear that this feature is needed in the future.

The evaluator mentions an issue with not being able to edit the linkage at certain points. The cause of this issue is unclear and the development team is unsure how to replicate this issue, but we will pay attention to the user evaluation to determine if this is a common issue.

In PMKS+ Web, which is an update to PMKS+ meant to align PMKS+ with PMKS+D, the user is able to change the shape of each link in a linkage to a number of distinct choices. The

lack of this feature is again due to development time, and the feature will be implemented in the future.

We are unsure of what caused the issue with the SolidWorks export, but the evaluator appeared to fix the issue themselves.

The evaluator was completely unable to replicate the 6-bar linkage with a force, as seen in Figure 11.2. We are not sure what caused this issue, as Figure 11.2. was created using the mouse-based control scheme as normal. We will pay attention to this problem during the user evaluations.

The evaluator initially had an issue creating the slider linkage, mentioning an issue with the DoF textbox being set to 3. This issue was due to the prismatic joint of the slider link not being toggled to a ground joint. This issue was caused by a deliberate design decision to require the user to toggle the joint to a ground joint. However, it makes sense to set a prismatic joint to a ground joint by default because a prismatic joint must always be grounded for the linkage to run.

The bug that occurs when deleting the ground joint initialized at the start of the program is a known bug and must be fixed in the future.

The SolidWorks export not creating a track for the sliding linkage is a known limitation of the feature and has not been implemented yet. This is warned in the confirmation window that appears when starting the SolidWorks export.

11.3.2. Individual UI Feature Testing

Slight differences in the UI between PMKS+D and PMKS+W are known and will be fixed in the future.

The issue with the scroll wheel has not been encountered by the development team, but we will pay attention to this problem during the user evaluations.

The problem with the clear button not updating the DoF textbox or joint table is not a bug, but was a design feature missed when implementing the clear button. This will be fixed in the future.

The issue with the first joint added after pressing the clear button snapping to (0, 0) appears to be a bug. It will be addressed in the future.

The issue with the Input Speed button is due to the design decision to make buttons represent the available actions rather than the current state. The button displays “x1” when the linkage is animating at “x3” speed to indicate that the button will change the speed to “x1”. We might decide to change this design decision depending on user feedback.

The Display IDs button not functioning appears to be a bug. This will be fixed in the future.

The issue with the Center button not centering the Canvas to (0, 0) appears to be a bug, but the development team has not encountered this issue yet.

The Analysis tab’s buttons not functioning is correct. Only the graphing for PVA data is functional. The other options will be implemented in the future.

The About and Linkages buttons not functioning is correct. These buttons will be implemented in the future.

The Help and Settings tabs being switched compared to PMKS+W was a design decision that will need to be discussed in the future. In most desktop applications, the Help tab is the rightmost tab on the upper menu, like in Adobe Acrobat and Microsoft Word.

The issues with the tables are known, and were not implemented due to a lack of time. These missing features will be implemented in the future.

The inability to change the input joint is a known issue and was left unimplemented due to time constraints. This will be implemented in the future.

The inconsistencies with the context menus are known and will be fixed in the future.

12. User Evaluation Design

To determine the success of PMKS+D's design, the team needed to have users test the system and collect their feedback. The team organized an informal focus group halfway through the project to collect feedback on initial functionality and design concepts. Secondly, the team organized a user evaluation survey administered remotely through Google Forms. This chapter outlines the design of both of these evaluations, as well as the processes used to analyze the collected data.

12.1. Focus Groups

Halfway through the project, the team conducted a focus group of Mechanical Engineering (ME) and Robotics (RBE) students to gather feedback on currently implemented features and the initial interface design. The 6 subjects were volunteers from Professor Radhakrishnan's mechanical engineering classes.

The team created a demonstration to show these qualities to our subjects. The following is a list of features the team demonstrated to the focus group:

- Controls for the Canvas
- Linkage creation process
- Animation controls
- Linkage graphics
- Interface layout

The team placed breaks between feature demonstrations to gather immediate feedback by asking the participants to share their thoughts. The feedback was typed up live into a Google Doc by our team. The team also showed a video of a linkage being exported to SolidWorks to gather feedback on that feature. To conclude the session, the team asked questions to prompt the subjects on additional preferences. The following is a list of four questions with which the team prompted the subjects:

- Did PMKS+ appear to shorten the number of linkage creation steps?
- Did PMKS+ appear to reduce error rates?
- What information would you like from this linkage program?
- What features would you want to see?

12.2. Design of User Evaluation Survey

The team was originally planning to perform an in-person, moderated user evaluation to collect preference and performance data, such as the average time taken to complete tasks. Unfortunately, circumstances required the team to perform a remote user evaluation. Therefore, some of the literature review research on calculating evaluation metrics and designing and moderating a user evaluation could not be applied to the extent that the team initially desired. For example, measuring speed of performance was determined to be too difficult to measure remotely.

Given what data the team could collect remotely, the team determined that the requirements for the user evaluation were:

- To discover usability issues within the UI;
- To collect subjective preference data on the interface and functionality;
- If possible, determine how the interface subjectively compares to similar applications;
- To determine the overall satisfaction with the application.

The team created a Google Form survey to facilitate an online user evaluation process. Through the use of a Slack channel, the survey, our application, and help materials were provided to students in the ME and RBE departments. These students were chosen because they represent our intended user base. The PMKS+D application was presented as a downloadable ZIP file containing the executable, necessary binary files, and a folder containing the template files needed for the SolidWorks export. The help materials provided additional information about how to properly use the different tools in the application.

The survey was divided into six sections: Welcome, Consent Form, Demographic Survey, PMKS+Desktop Tasks, Follow-Up Questions, and Concluding Questions. The welcome section introduced the users to PMKS+D, the purpose of the evaluation, and the consent form that informed the user of their rights. The next sections of this chapter will explain sections of the survey.

The full survey and consent form can be referred to in Appendix B: User Evaluation Script.

12.2.1. Demographics

The goal of the demographic survey was to gather information about the user's experience with other software related to PMKS+D. The users were asked to select their experience level with Working Model, PMKS Silverlight, PMKS+Web, and SolidWorks from one of three categories: not at all, moderately, and very experienced. The survey also asked students to select which courses the students had taken from a list of relevant courses. These questions were chosen to allow the team to determine if the satisfaction of a user was affected by their experience level with other similar applications.

12.2.2. PMKS+Desktop Tasks

The next step in the survey asked the subjects to perform tasks in the application. The first task was to recreate the linkage from the provided screenshot. Once the user had finished attempting to create the linkage, they were asked to upload a screenshot of their linkage creation to allow us to assess whether they had created the linkage correctly. An image of the linkage in the survey is shown below in Figure 12.1.

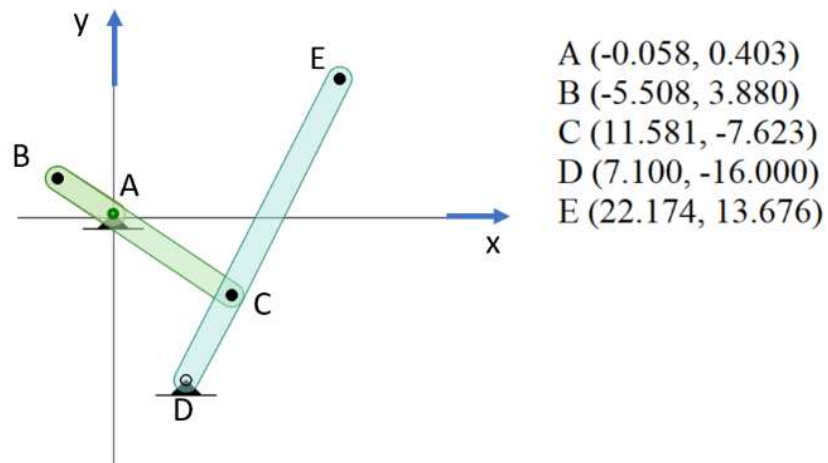


Figure 12.1.: Image of the linkage provided in the user survey.

This section of the survey then asked the user to animate the linkage clockwise, animate the linkage counterclockwise, change the animation speed, pause the animation, and reset the animation to the start. The next task asked the user to open a graph of the angular velocity of link BC and upload a picture to allow us to assess whether the user had found the graph. The expected graph can be seen in Figure 12.2. below.

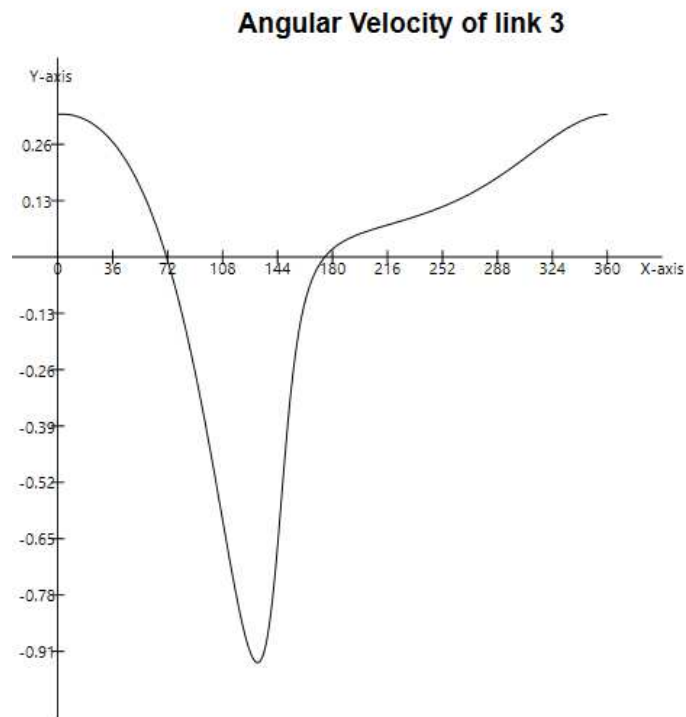


Figure 12.2.: Expected angular velocity graph for user evaluation submission.

Finally, the survey asked the subjects to test the export to SolidWorks feature. Subjects were asked to upload an image of the generated CAD model. An image of the expected CAD model can be found in Figure 12.3. below.



Figure 12.3.: Expected SolidWorks CAD model for user evaluation submission.

These tasks were chosen because the team felt that these tasks would introduce the user to the most important features and UI aspects of the application. Users would also encounter any potential usability issues with the interface, and the screenshot uploads would allow us to calculate the success and error rates for each task. The data collected from the image uploads would allow the team to evaluate how effective and error tolerant the interface is, as described in Chapter 5.2.1.: The Most Common Metrics.

12.2.4. Open Response and Follow-Up Questions

The second-to-last section of the survey contains open response and follow-up questions for the PMKS+D tasks performed in the previous section. This section contained one categorical qualitative question concerning the speed of the linkage creation and editing process. It asked

users to select if the process was quicker than, about the same as, or slower than they expected.

The remaining questions were long answer questions where the subjects were allowed to write their responses. The following list contains the remaining follow-up questions:

- What did you find difficult in PMKS+ Desktop?
- What did you feel was easy in PMKS+ Desktop?
- What did you like about PMKS+ Desktop?
- What did you dislike about PMKS+ Desktop?

These questions were chosen to obtain subjective preference data on the interface and functionality that the user encountered while performing the tasks. The responses for difficult and easy features of PMKS+D would allow the team to evaluate how efficient and easy to learn the interface is, as described in Chapter 5.2.1.: The Most Common Metrics. The responses for liked and disliked features would allow the team to evaluate how engaging the interface is, as described in Chapter 5.2.1.

12.2.5. Concluding Questions

The final section of the survey asked a few remaining questions that gather more information on the overall opinion of the application. This section contains one quantitative question asking the user to rate their satisfaction of PMKS+D on a scale of one to five, where one is very unsatisfied and five is very satisfied.

The remaining questions are open response questions, and are listed below.

- Please list the top 3 features that you used in PMKS+ Desktop.
 - If you want, please explain your choices.
- Please write any comments you have on how PMKS+ Desktop compares to Working Model, PMKS Silverlight, or PMKS+ Web.
- Please list any recommendations for features and improvements you would like to see in PMKS+ Desktop.

The Likert scale question was chosen to provide quantitative data about the participants' overall satisfaction with PMKS+D. The open response questions were chosen to allow us to determine the opinion of PMKS+D relative to other applications that the participants used and collect recommendations for future work that can be done on the software.

12.3. Design of Analysis Procedure

The team created a procedure to analyze the data collected by the survey. The demographic section provides information on user categorization. Secondly, the image uploads provide a metric of the success rate of creating the linkage, generating a graph, and generating a CAD model. Finally, the qualitative and quantitative data on user feedback provide data on the approval rating of the application as a whole as well as approval rating on individual features.

12.3.1. User Categories

The team created categories of users based on their experience with Working Model and PMKS Silverlight. These categories were divided into users who had some experience versus those who were not experienced. This allowed the team to determine if the experience of students had an effect on the responses received. The category of PMKS+ Web was ignored because very few users had used it, and the category of SolidWorks was ignored because it is not an application that is comparable to PMKS or Working Model.

12.3.2. Image Upload

The team also created an analysis plan for the image uploads. The first image upload asked users to provide an accurate image of the provided linkage. The second prompt asked students to upload an image of an angular velocity graph of a link. The final prompt was for an image of a generated SolidWorks model.

The first of the image uploads asks the user to upload an image of the linkage they created in PMKS+D. The linkage should look like the linkage in Figure 12.4. below:

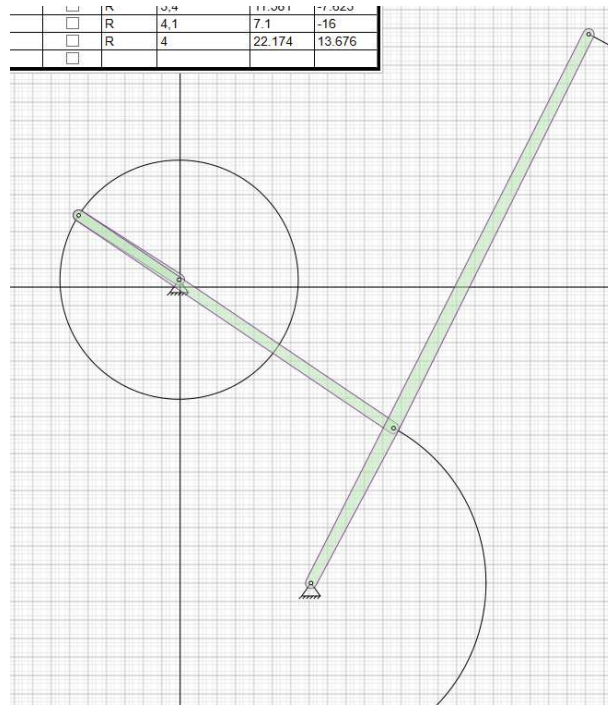


Figure 12.4.: Correct recreation of example user evaluation linkage.

The team considered an image as correct if there wasn't any noticeable deviation in joint location, incorrect/empty path lines, or absence of any given component. The team also considered submissions as correct if the linkage appeared to be correct but was in the middle of animating. The correctness was confirmed by the entries in the joints table and locations of the movement paths..

The second image upload asked students to find a graph of the angular velocity of link BC, which appears in Figure 12.5. below.

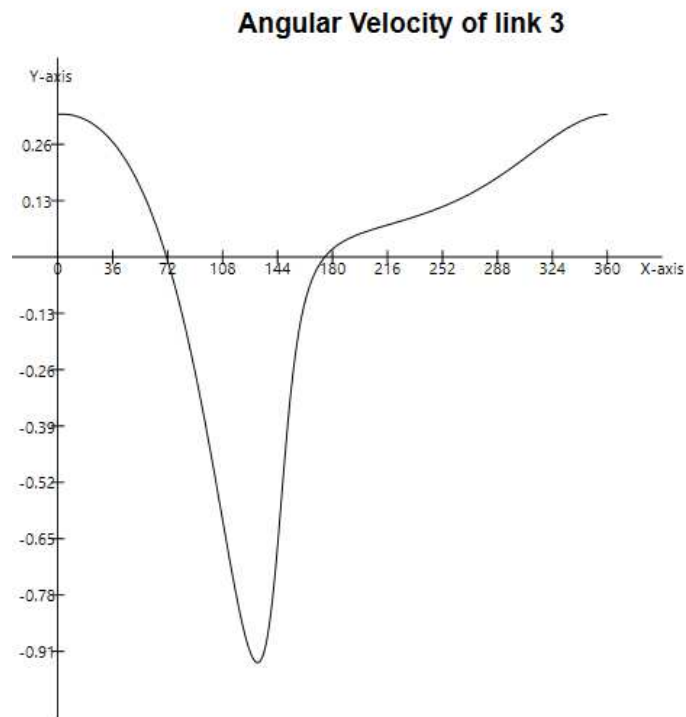


Figure 12.5.: The correct angular velocity graph for the user evaluation.

Because the team wanted to test if users could find the graph for the angular velocity of a link, the image was considered correct if it was of a graph of the angular velocity of a link.

The final image upload was of the generated CAD model using the “Export to SolidWorks” button. A correct implementation would look like Figure 12.6. below.

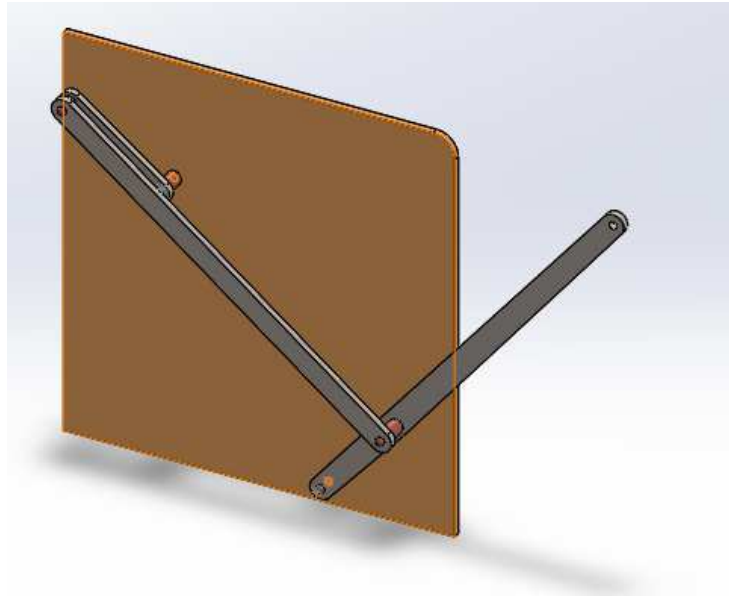


Figure 12.6.: Correct CAD model from SolidWorks export.

Because the team wanted to see if users could complete the export process, the image was considered correct if the image was of any CAD model generated through the SolidWorks export button.

Using the analysis techniques described, the team was able to determine the success rate of these three tasks. Correctness was also analyzed with user responses to understand more about the results.

12.3.3. Likert Scale

Two questions had users rate different aspects of their experience with PMKS+D on a Likert scale. Each response was counted and the mode was determined. Then, the responses were charted in a column chart as percentages of the evaluation group to determine the distribution of responses.

12.3.4. Open Response

The remaining data consists of answers to the qualitative questions of the survey. Responses to these questions were read, categorized, and counted to determine common responses. For example, if a user replied that they liked the SolidWorks export, the animation controls, and the linkage creation process, then a single tally was added to the SolidWorks export, animation controls, and linkage creation response counts. Then, the most common responses for each question were determined and displayed in percentage column charts.

13. Data Analysis

Once the results from each user evaluation were collected, the raw data could be analyzed. From this analysis, the team could determine the user's satisfaction with different features of PMKS+D. The team also made decisions to change the design of PMKS+D or suggested future changes to address issues the users encountered.

13.1. Focus Group

The focus group provided the team with many potential feature additions. The participants gave several useful points of feedback on the UI mockups.

First of all, the position of the user controls, especially the lower menu, was thought to be too one-sided. The team later adjusted the lower menu to span the entire window.

Secondly, the group suggested adding more hover text to explain the controls on the screen without needing to view the help. As a result, the team implemented hover text on the basic controls, and more extensive hover text is being considered for future work.

Thirdly, the group expressed the desire to specify the input RPM for the analysis. This suggestion was incorporated within the Settings tab of the upper menu.

The focus group also suggested several features that they would like to see as mechanical engineering students. One feature participants wished to see was a length column within the link table so they could view and edit the length between each joint of each link. Similar to link length participants also suggested a length between joints in a poly link. Participants also

suggested undo and redo functionality. These and other suggestions are strongly considered by the team for future work, as it would align with the “User freedom and control” heuristic.

13.2. User Evaluation

Once the deadline for the user evaluation was reached, the responses for the survey were closed and the team collected the responses. This section describes the data given by the 35 participants of the survey, analyzed according to the data analysis plan described in Chapter 12.3.: Design of Analysis Procedure.

13.2.1. User Categories

- Working Model: Experienced **11/35** (31%) Students
- PMKS/PMKS+: Experienced **11/35** (31%) Students
- Experienced in One or More: **16/35** (46%) Students

13.2.2. Image Uploads

The following list contains the calculated error rates for uploading a correct image for each task, as determined by the process described in Chapter 12.

- Linkage Creation: **9/35** (26%) Incorrect
- Angular Velocity Graph: **6/35** (17%) Incorrect
- SolidWorks Export CAD Model: **8/35** (23%) Incorrect

13.2.3. Open Response

The open response questions were analyzed according to the process described in Chapter 12. All responses to these questions can be found in Appendix C: Participant Responses to Open Response Questions. The lists below contain the three most common responses for each question.

Difficult to Use Features:

The first open response question asked about what the users found difficult in PMKS+Desktop. The responses to this question can be found in Appendix C.1.: What Was Difficult in PMKS+Desktop. The three most common responses were:

- Crashing (37%)
- Table Editing (23%)
- Creating Poly-Links/Finding Analysis Graph (17%)

Figure 13.1. below is a chart displaying the percentage of students that gave a certain response for what they felt was difficult in PMKS+D.

What did you find difficult in PMKS+Desktop?

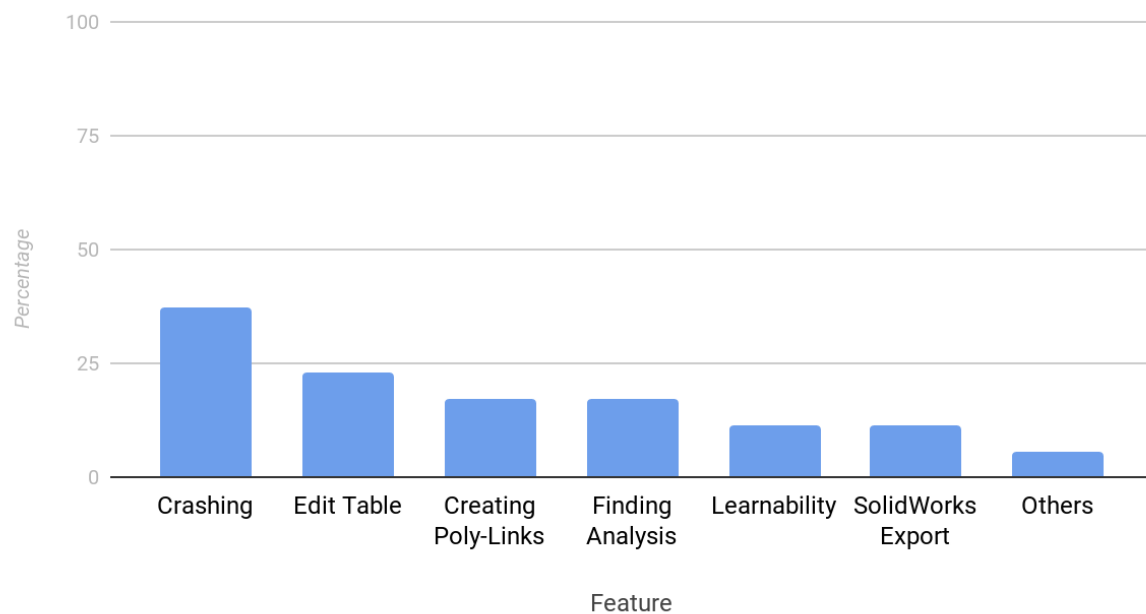


Figure 13.1.: Percentage of users for each difficult feature in PMKS+D.

Easy to Use Features:

The next open response question asked about what the users felt was easy in PMKS+Desktop. The responses to this question can be found in Appendix C.2.: What Was Easy in PMKS+Desktop. The three most common responses were:

- Creating Linkage (**57%**)
- Graphs/Analysis (**31%**)
- SolidWorks Export (**29%**)

Figure 13.2. below is a chart displaying the percentage of students that gave a certain response for what they felt was easy in PMKS+D.

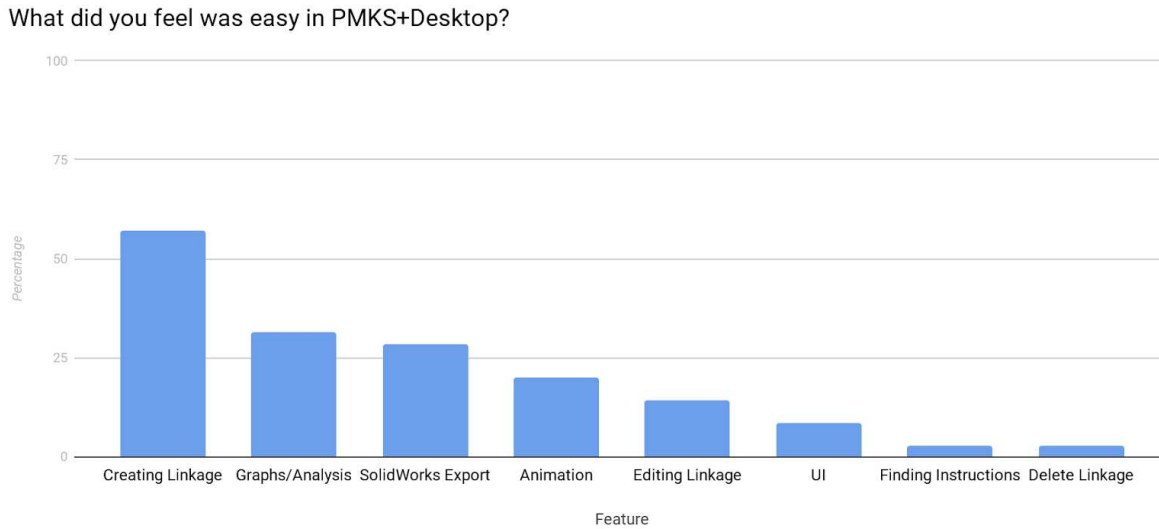


Figure 13.2.: Percentage of users for each easy feature in PMKS+D.

Liked Features:

The next open response question asked about what the users liked about PMKS+Desktop. The responses to this question can be found in Appendix C.3.: What Was Liked in PMKS+Desktop. The three most common responses were:

- SolidWorks (**46%**)
- Easy to Use (**43%**)
- UI/Graphs/Animation (**12%**)

Figure 13.3. below is a chart displaying the percentage of students that gave a certain response for what they liked about PMKS+D.

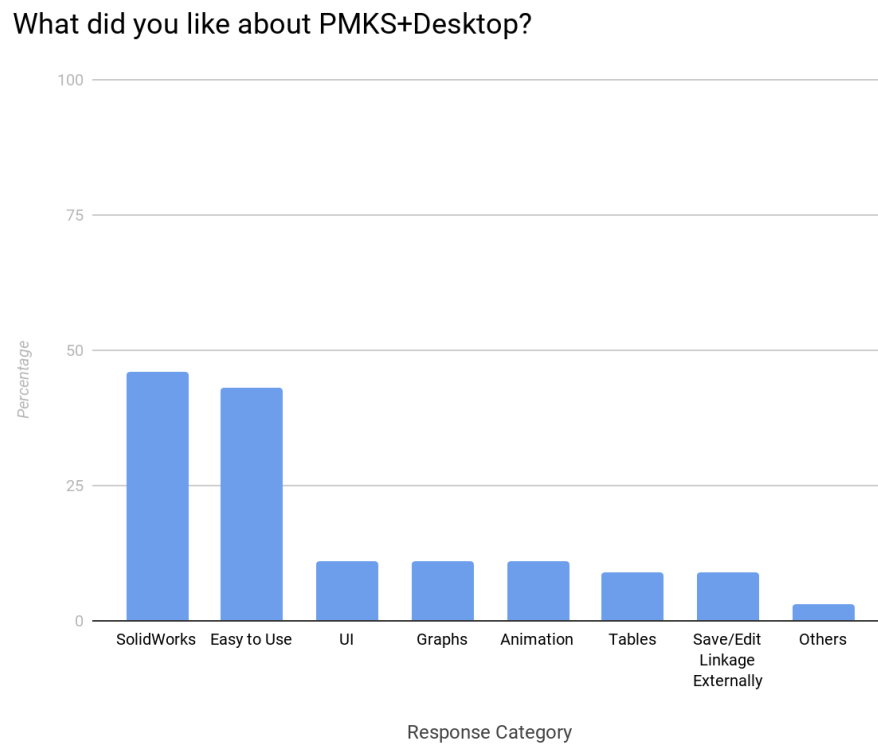


Figure 13.3.: Percentage of users for each feature liked in PMKS+D.

Disliked Features:

The next open response question asked about what the users disliked about PMKS+Desktop. The responses to this question can be found in Appendix C.4.: What Was Disliked in PMKS+Desktop.

The three most common responses were:

- Crashing (**30%**)
- Creating Linkage/Lack of Undo/Clunky UI/Edit Table Not Working/Not Enough Tutorial (**12%**)
- Not Enough Help/Slow Scroll Zoom (**10%**)

Figure 13.4. below is a chart displaying the percentage of students that gave a certain response for what they disliked about PMKS+D.

What did you dislike about PMKS+Desktop?

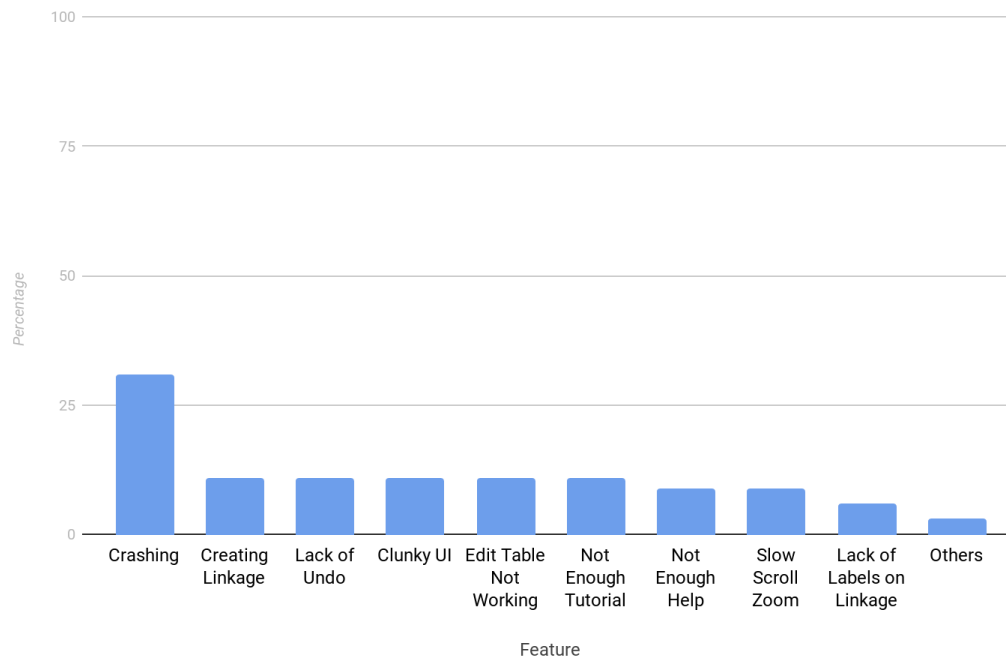


Figure 13.4.: Percentage of users for each feature disliked in PMKS+D.

Top Three Features:

The next open response question asked about what the users disliked about PMKS+Desktop. The responses to this question can be found in Appendix C.5.: Top 3 Features in PMKS+Desktop. The three most common responses were:

- SolidWorks (**63%**)
- Linkage Creation (**51%**)
- Analysis (**34%**)

Figure 13.5. below is a chart displaying the percentage of students that gave a certain response for what they felt was one of the top 3 features in PMKS+D.

Features Mentioned in Top 3 Features

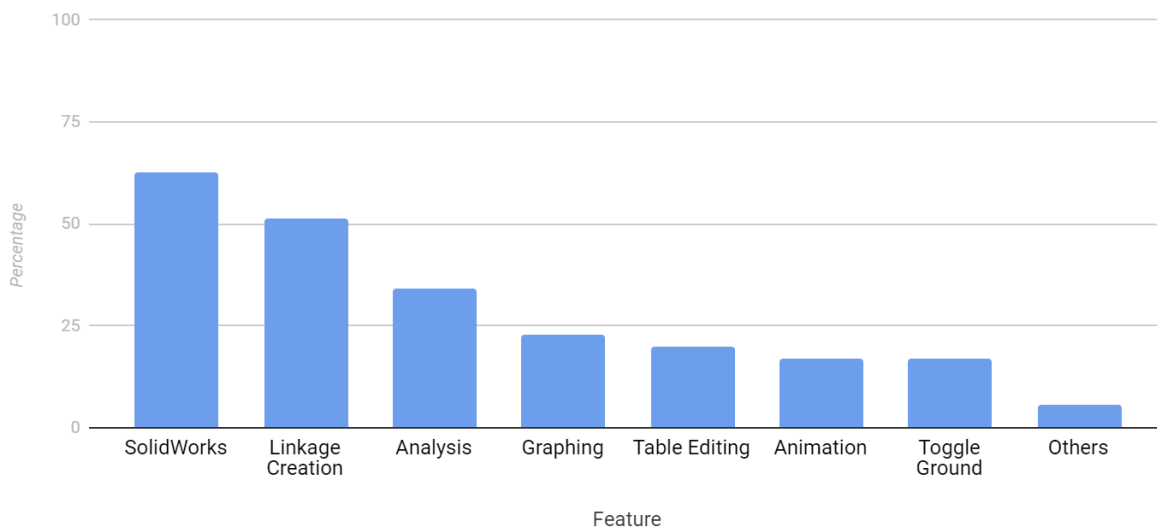


Figure 13.5.: Percentage of users that placed each feature in the top 3 features of PMKS+D.

13.2.4. Rating and Follow-Up Questions

The remaining analysis pertains to the rating and follow-up questions, as described in Chapter 12.

Speed of Linkage Creation and Editing Process:

The first rating question asked users to rate how quick the linkage creation and editing process felt to them. The mode was found to be “Quicker than expected” at 51% of users, with “About the same” at 23%, and “Slower than expected” at 26%.

Figure 13.6. below is a graph displaying the percentage of students that gave each response to this question.

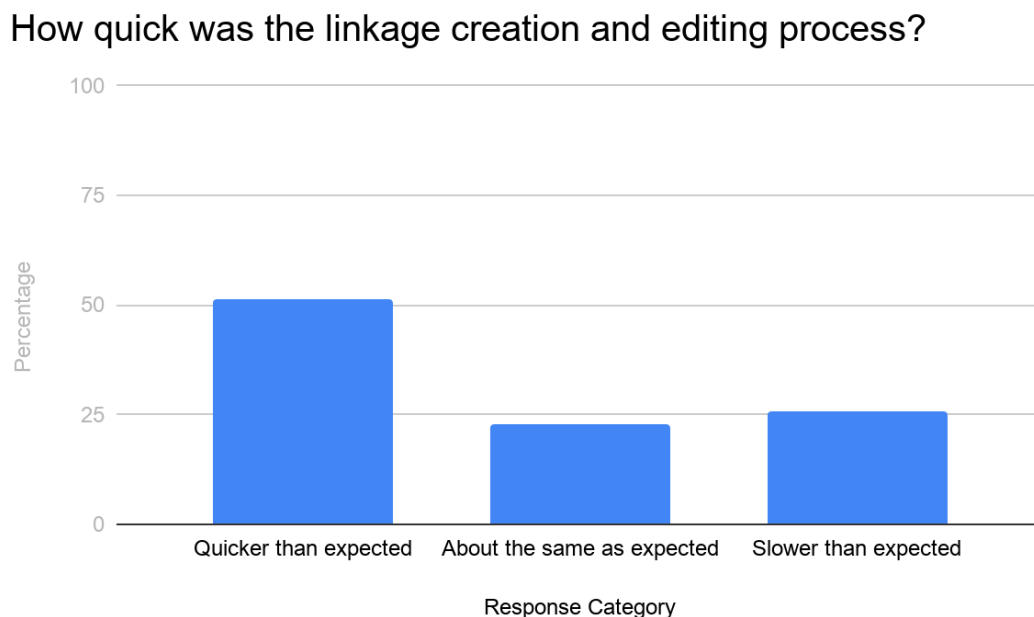


Figure 13.6.: Percentage of users that rated each category of the linkage creation and editing speed.

Overall Satisfaction with PMKS+Desktop:

The next rating question of the survey asked users to rate their satisfaction with PMKS+D on a 5-point Likert Scale. The mode was found to be a rating of 4 at 43%. The percentages for each rating are:

- 5: 9%
- 4: 43% (Mode)
- 3: 31%
- 2: 14%
- 1: 3%

Figure 13.7. below is a graph that displays the percentage of students that gave each Likert Scale rating for their overall satisfaction with PMKS+D.

Overall Satisfaction with PMKS+Desktop

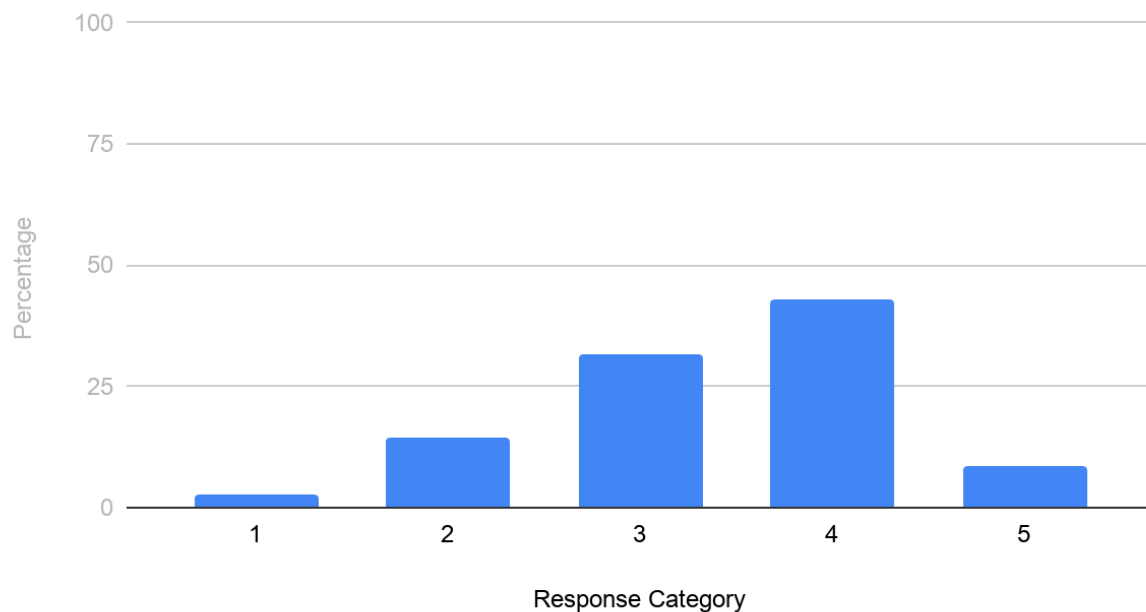


Figure 13.7.: Percentage of users that gave each Likert Scale ranking for overall satisfaction with PMKS+D.

Comparison to Working Model, PMKS, or PMKS+:

The first follow-up question asked students to compare their experience with PMKS+D to Working Model, PMKS, or PMKS+, if applicable. The responses of the students that had some experience with Working Model, PMKS, or PMKS+ were counted. Responses were categorized as Positive, Neutral, or Negative. Those with no experience in any application were counted as not applicable (N/A). All responses for this question can be found in Appendix C.6.: Comparison of PMKS+Desktop to Other Software.

The percentages for each response were:

- Positive: 31% of total, 69% of applicable
- Neutral: 11% of total, 25% of applicable
- Negative: 3% of total, 6% of applicable
- N/A: 54% of total

Figure 13.8. below is a graph that displays the percentage of students that gave each type of response comparing PMKS+D to other applications.

Comparisons of PMKS+D to Other Software (WM, PMKS, PMKS+)

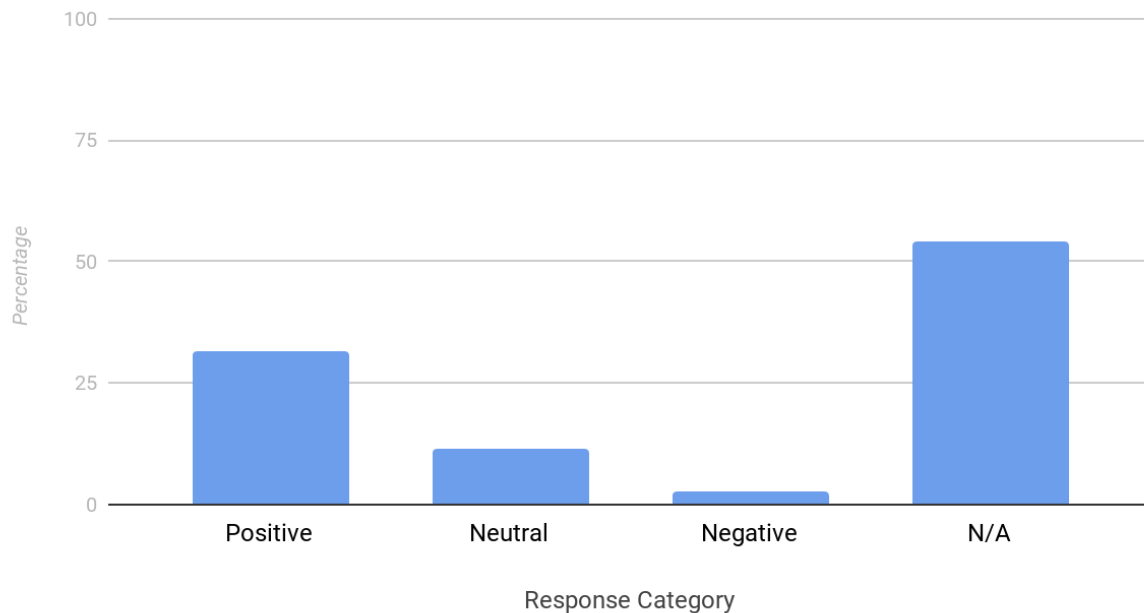


Figure 13.8.: Percentage of users that gave each response comparing PMKS+D to Working Model, PMKS, and PMKS+.

Recommendations to Improve PMKS+D:

The final question of the survey asked users to suggest new features and improvements for PMKS+Desktop. All the responses for recommendations to improve PMKS+D can be found in Appendix C.7.: Recommendations for Improving PMKS+Desktop. The three most common responses were:

- Fix existing bugs and features (**49%**)
- Improve help and tutorials/Implement planned features (**14%**)
- Add editing link angle/length to table (**11%**)

Figure 13.9. below is a graph that displays the percentage of students that gave each recommendation on how to improve PMKS+D.

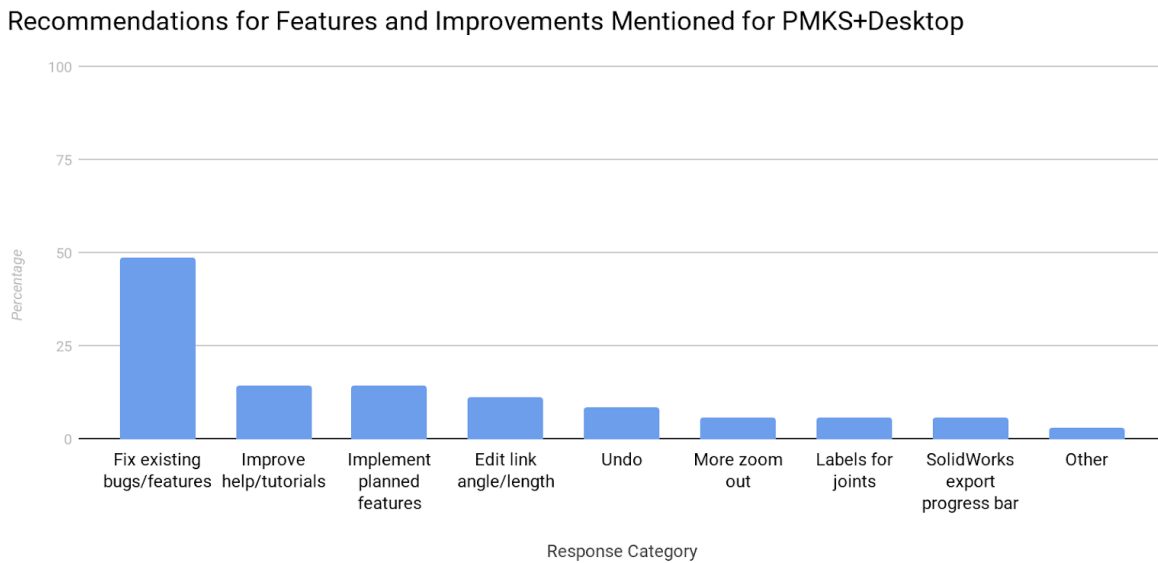


Figure 13.9.: Percentage of users that gave each recommendation for PMKS+D.

13.3. Evaluation of Analysis

With the raw data processed, the team could begin analyzing the processed data to construct conclusions about the evaluation.

13.3.1. User Categories

The responses to the demographic questions demonstrated that 46% of the participants of this survey were experienced with at least one alternate linkage simulator. Additionally, 31% of the users had some experience with PMKS or PMKS+.

This result is good because it means that the results of our survey come from an audience that has a mix of experienced and new users, rather than being homogenous. We have a large enough dataset of both to be able to see if the data differs for experienced and inexperienced users.

13.3.2. Image Uploads

The image uploads are a good indication of which features are both intuitive and simple. Roughly 74% of the students were able to accomplish the task of creating the linkage properly, which indicates that the task is of average difficulty, as suggested by Rubin & Chisnell (2008). Furthermore, most of the remaining incorrect images showed that participants made attempts that would have been successful if not for a few minor changes. For example, in Figure 13.10. below, a linkage is nearly correct, but the joints are not located in the correct position.

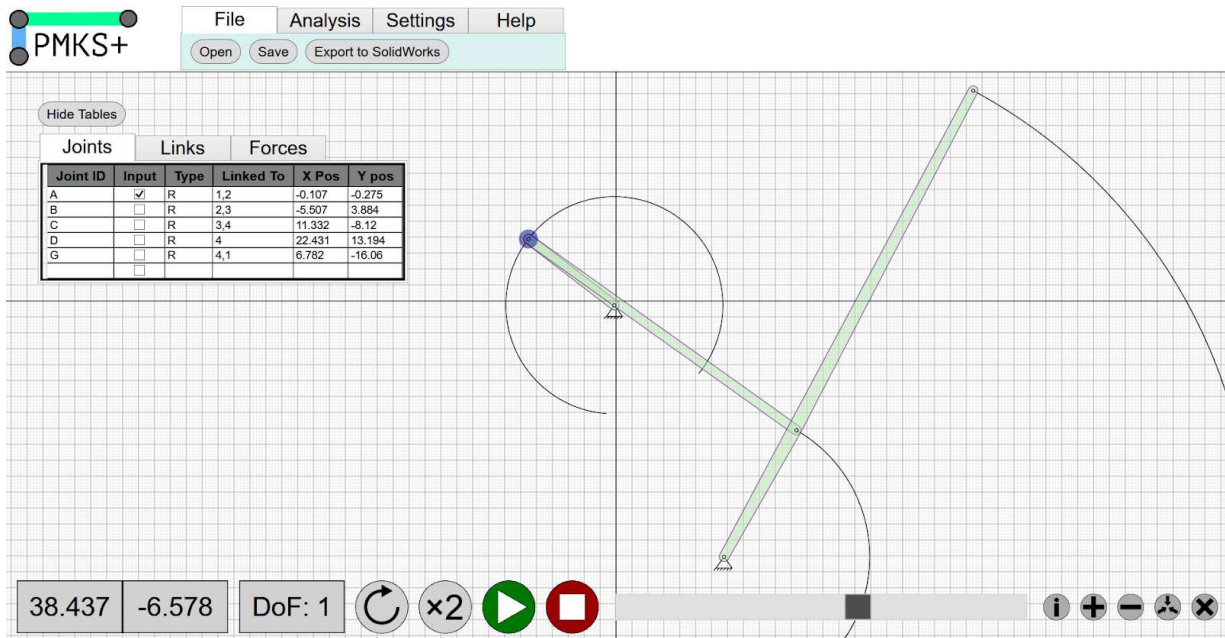


Figure 13.10.: A nearly-correct linkage uploaded.

Angular velocity image upload accuracy rose to 83%. It appears that it might be easier to locate the graphing functionality than figure out the mouse-based creation method. While some users struggled to find the graph completely, as shown by uploading an empty screenshot, one user managed to find the joint graphing functionality (they uploaded a screenshot of a joint graph), and another was confused by the unimplemented Analysis tab (they uploaded the “unimplemented” error message that appears when clicking buttons in that tab). When the application is fully implemented and more help text is included, it might be possible to improve the success rate for this feature.

The final image upload of the CAD model produced an accuracy of 77%. This indicates that the SolidWorks generation is also of average difficulty. This is appropriate, considering the moderate 77% TV of this task calculated in in Chapter 9.2.: Applying UI Evaluation Metrics.

Additionally, multiple users uploaded images of error messages referring to crashes and bugs. If these issues are fixed, it is possible that the error rate for the SolidWorks export can decrease.

An issue that some users had with the SolidWorks export was figuring out the purpose of the two File Explorer windows. While these windows had a sentence describing what the user had to do, our team spoke with three users on Slack that did not understand that the first File Explorer window required them to locate the provided SolidWorks template files. It is possible that more users would have been able to complete the task successfully if this step were to be removed. This is supported by the 40% EE of the SolidWorks export task calculated in Chapter 9.2.: Applying UI Evaluation Metrics. If excess steps could be trimmed, the process could be easier for students to complete.

Overall, it appears that the three features evaluated were of average to below average difficulty. This is a good sign, and it appears that the success rate for these tasks can be improved through bug fixes and better help materials.

13.3.3. Open Response

What subjects liked most about the application was the SolidWorks export feature. Only a quarter of participants felt that the process was easy to complete, but nearly half of participants said they liked the feature, and 63% mentioned it in their top 3. This shows that there is room for improving the SolidWorks export's ease of use, but it is a very popular and important feature.

It is a good sign that the second most liked aspect (43%) about the application was how "Easy to Use" the application was. This further supports that the UI of PMKS+ was well-designed and was a good basis to use for PMKS+D.

It is also worth noting that the linkage creation method was positively received. The majority of participants (57%) felt that creating a linkage was the easiest aspect of the program, and the majority of participants (51%) listed the linkage creation within their top 3 features. While these numbers could be better, it is important that most of the participants felt strongly enough about the process to mention it multiple times. This result aligns with the 36% EE and 85% TV for the linkage creation method calculated in Chapter 9.5.: Applying UI Evaluation Metrics. The TV is high, supporting that users found the method easy to figure out. However, EE is low, supporting that there is room to improve the linkage creation process.

One aspect of the linkage creation method was mentioned to be particularly difficult. The creation of a poly link was mentioned 17% of the time by participants, which is a significant amount. This issue also appeared in the testing described in Chapter 11.: Application Testing, so this area should be investigated more thoroughly for potential bugs and usability issues.

The graphing and analysis was another major feature that the participants experienced. The response to this feature was somewhat mixed. 12% of participants listed it as a feature they liked, but 0% mentioned it as a feature they disliked. 31% felt that the analysis and graphing was easy, but 17% mentioned that finding the analysis was difficult. Finally, analysis and graphing were mentioned as a top 3 feature by nearly a quarter of users. It appears that the graphing functionality is not particularly well-liked, but it is a useful feature that participants rate highly. The perception of the feature could also be improved through better help materials and implementation of the **Analysis** tab.

This response aligns somewhat to the EE and TV of the angular velocity graph task using the context menus, as calculated in Chapter 9.5.: Applying UI Evaluation Metrics. Due to the

Analysis tab not being fully implemented, participants were forced to use the context menu method to reach the angular velocity graph. This method has a somewhat low 67% TV compared to the 77% TV of the **Analysis** tab method. This could explain the users that found the graph hard to locate and the responses that were disappointed that discovered the **Analysis** tab was not functional. The context menu method also has a perfect 100% EE, which could explain the significant portion of students that liked the feature and felt it was easy to use.

The most common aspect that subjects disapproved of the application was the crashing issues (37%). Crashing was an unfortunate product of the time constraints on the team. However, should these bugs be fixed, a very large portion of the negative response would be resolved.

The second largest complaint (23%) about PMKS+D was the difficulty of using the tables to edit the linkage. Due to bugs, using the table to edit the linkage could cause crashes or otherwise force the user to restart the program. Again, future bug fixes could improve the perception of the application by a significant amount.

11% of participants disliked the insufficient tutorial and help materials, supporting the speculation above that many users were having issues due to a lack of help materials.

13.3.4. Rating and Follow-Up Questions

Again, users as a whole appear to like the linkage creation method. The majority (51%) of participants felt that the process was quicker than expected. However, 26% felt that the process was slower than expected. This is interesting because the creation method was not listed as a difficult aspect of the system. 12% stated that they disliked creating linkages, but this does not account for the whole discrepancy. This result is, however, somewhat corroborated by the

low 36% EE of the linkage creation method calculated in Chapter 9.5.1.: Essential Efficiency.

Perhaps the response to this question would improve if the EE were improved by trimming down the number of context menus required, as suggested in that chapter.

The Likert Scale for overall satisfaction of PMKS+D had a mode of 4 out of 5. This is a good indication of an overall positive user experience. 14% of users gave PMKS+D a negative overall rating, while nearly a third (31%) of users gave a neutral rating. There is room for improvement, but the previously discussed adjustments to prevent crashing and completion of remaining features would likely cause the rating to go up as well.

The overall comparison of PMKS+D to other software such as Working Model, PMKS, or PMKS+ was positive. A majority (69%) of experienced users felt that PMKS+D was an improvement to these applications and only 6% had a negative opinion, which is a very promising result. As one experienced user says, “[t]he dislikes are just the glitches and issues right now... [i]t seems super promising however”. Another experienced user said that PMKS+D “[d]efinitely has the potential to be much better than Working Model”.

The recommendations made by students will be explored in more depth in Chapter 14.2.: Future Work. The most common responses included fixing the encountered bugs so the included features work correctly (49%), improving the help materials and tutorial (14%), and implementing features that are already planned (14%), like implementing the Analysis tab. Again, this shows that our application could be greatly improved simply by fixing the bugs, improving the help and tutorial materials, and fully implementing functionality.

14. Conclusions

After the completion of all intended components of the project, the evaluation of the success of the project could be conducted. The following sections of this chapter discuss the team's evaluation of the project, potential future work, and our experience with this project.

14.1. Evaluation of the Project

In order to evaluate the success of the project, all established goals must be considered. This section will discuss both the project's goals and the user evaluation's goals to determine if the team succeeded in this project.

14.1.1. Achievement of Goals

Below is the list of goals from Chapter 3. that the team initially chose:

- **Convert** PMKS from a web-based Silverlight application to a C#-based, Windows Presentation Foundation desktop application.
- **Implement** the functionality of PMKS Silverlight using C#.
- **Replicate** the UI of PMKS+ and make necessary adjustments for a desktop environment.
- **Conduct** user evaluations and improve the UI and UX from collected user feedback.

The team **successfully converted** the PMKS Silverlight application to a C#-based, WPF desktop application. PMKS+Desktop is a C#-based, WPF desktop application and has implemented the most important features of PMKS.

The team **has not fully implemented the functionality** of PMKS Silverlight using C#. There are some features, like adjustment of prismatic joint angles, that the PMKS Silverlight application could accomplish that PMKS+D currently cannot. Furthermore, table editing was only partially implemented, and other functions, such as the scroll wheel zooming, have bugs that need to be addressed. However, while not all of the functionality is converted yet, **the most important functionality of PMKS Silverlight has been implemented**. The simulator, Canvas, and linkage animation are all fully functional, and PMKS+D's system and interface designs were constructed to allow the missing features to be easily implemented in the near future. Additionally, new functionality, like the SolidWorks export and graphing, were prioritized above low priority features of PMKS.

The team successfully **replicated the UI of PMKS+**. The UI of PMKS+D is clearly an evolution of the original PMKS+ UI. By working closely with the advisors and the 2019-2020 PMKS+Web MQP team, our team was able to coordinate **necessary additions and adjustments to the UI** to synchronize PMKS+D with PMKS+W.

The fourth goal the team set was the improvement of UI and UX from collected user feedback. The team has **conducted a focus group**, which yielded a few results that the team used to **somewhat improve the UI and UX** of the application. However, the team **has not had the time to implement the feedback from the recently-conducted user evaluation** to improve the UI and UX of PMKS+D.

Overall, the team feels that they have successfully met the majority of the established goals for the project. Secondly, the partially unmet goals were caused by reprioritization agreed upon by both the team and advisers, and complications outside of our control. Therefore, the team feels that the project can be considered as satisfactorily complete.

14.1.2. User Evaluation Achievements

The requirements for the user evaluation from Chapter 12. were:

- To **discover** usability issues within the UI;
- To **collect** subjective preference data on the interface and functionality;
- If possible, **determine** how the interface subjectively compares to similar applications;
- To **determine** the overall satisfaction of the application.

The user evaluation has **successfully discovered usability issues** within the UI. As shown in Chapter 13., multiple issues with the interface were made apparent through user feedback, such as the crashing issues and an insufficient amount of help information.

The user evaluation has **successfully collected preference data on the interface and functionality**. Features such as the SolidWorks export and mouse-based linkage creation method were praised, and features such as the graphing and tables were disliked for their inconsistencies and apparent bugs.

The user evaluation was able to **collect some information** on how experienced users **compare PMKS+D to similar applications**. The majority of the feedback from experienced users was that PMKS+D fares better than other linkage simulation applications, and multiple

users expressed that PMKS+D has potential. If the bugs are fixed and the functionality is fully implemented, PMKS+D could be competitive with professional, paid applications, such as Working Model.

Finally, the user evaluation has **successfully determined the overall satisfaction** of PMKS+D. The Likert scale has demonstrated that most users rate their satisfaction positively, and many neutral and negative respondents have complained about easily-fixable issues like crashes, bugs, and help information.

The team believes that their user evaluation has successfully accomplished all of their established requirements. Therefore, while the team needed to adapt the initial evaluation plan due to outside circumstances, the user evaluation can be considered a success.

14.2. Future Work

The team worked to implement several new features for the PMKS+D application. Due to time constraints, the team was not able to complete all of the features from PMKS or all desired additional features. Furthermore, there are bugs that will need to be fixed. Also, through the user evaluations, the team collected additional potential features that would be useful to implement.

14.2.1. Unfinished Features

There were a couple of features that presented difficulties for the team. While these features are implemented to some extent, they will need to be improved and completed in the future. Below is a list of features that were implemented to some extent but remain incomplete:

General:

- Fix bugs and crashing issues

UI:

- Changes to the UI to synchronize it with PMKS+Web
- Implementing all graphs and buttons in the Analysis tab
- Implementing all buttons in the Help tab

Context Menu:

- Set Input: requires the reordering of Model information to allow an arbitrary ground link to be set as the input
- Remove Input: was temporarily removed from the implementation to avoid issues with Set Input

Canvas/Link Manipulation:

- Drag Force: a force's initial position is not able to be dragged and placed at any point within a polylink's shape
- Adding Link: occasionally adding a joint and a link on a complete linkage causes all components of the linkage to become uneditable

Tables:

- **Joints:** editing the ID, Input, Type, and Linked To columns cause the program to crash
- **Links:** due to time constraints, only the Link ID column was implemented; the other columns need to be implemented as well
- **Forces:** editing the ID and On Link columns cause the program to crash

14.2.2. Future Features

There are several features from PMKS and PMKS+ that need to be added in the future. Additionally, there are many features and improvements that were suggested by users that would improve the application, such as the suggestions by users found in Appendix C.7.:

Recommendations for Improving PMKS+Desktop. Below is a list of features in no particular order that could positively benefit PMKS+D:

- **Change the upper menu from a tab menu to a menu with section headers:** Improves the essential efficiency and task visibility of multiple tasks
- **Improved Tutorials and help information:** improves user comprehension of the application's functionality
- **A dictionary of terminology:** improves user comprehension of the application's terminology
- **View and edit a link's length in the table:** provides more precise editing of a linkage and seems more intuitive to users
- **Snap dragged objects to gridlines:** allows user to more-easily place joints in the desired place

- Confirmation when clearing the linkage: prevents the user from accidentally deleting their linkage
- Additional hover text: improves user comprehension of the UI
- Accessibility options: e.g., colorblind color modes
- Undo/Redo: allows the user to fix mistakes automatically
- Keyboard shortcuts: provides a faster way to accomplish tasks for experienced users
- Introduction window: allows the user to select a blank Canvas or example linkages
- Exploded freebody diagram view of linkage: provides additional information about the forces acting on the linkage
- Auto-update/update button: doesn't require the user/Information Technology department to manually update the software
- Multi-threading: potentially speeds-up the application
- Compatibility with MATLAB: export analysis data directly into MATLAB
- Homework problems and solutions: allow students to complete assignments remotely

14.3. Project Experience

This year-long project required a great deal of previously-acquired skills and knowledge while also requiring us to develop new skill sets and knowledge bases. In the following subsections, the team discusses what initial skills and knowledge were required for this project as well as the skills and knowledge gained from this experience.

14.3.1. Applied Skills

Due to the application's history and unique functionality there were several known skills that applied particularly well to this project. The team began the project with little C# and XAML experience. Due to several completed Computer Science courses at WPI, the team already had extensive coding experience with C++, Java, and HTML. This allowed the team to quickly learn the new languages required for the project.

Over the course of the year, the project required many meetings with advisers, professors, students, and other teams, where large amounts of useful information were discussed.

Documentation skills were very important for maintaining a fast progression through the different stages of the project while being able to reflect on what was previously discussed.

Additionally, notes needed to be organized so we could retrieve important information quickly.

The project also required the team to learn several new things independently. **Self-taught learning** and research skills were very important skills that allowed the team to quickly adapt to unforeseen complications and acquire new information.

14.3.2. Applied Knowledge

The team also needed to apply a wide knowledge base in order to complete the project. This knowledge base included basic user interface design, compartmentalization, and algorithm efficiency.

Without a basic understanding of **user interface design**, the team would have struggled with learning more about UI design during our literature review and exploration of PMKS and

PMKS+. Fortunately, the team had previous UI design knowledge gained from WPI courses. This allowed the team to focus on researching more advanced topics, such as metrics, heuristics, and user evaluations.

PMKS+D centered around the implementation of the original PMKS Silverlight simulator. This required a knowledge of **compartmentalization**. By dividing the code into smaller components that communicate with each other through a limited set of data structures and variables, the team was able to build an understanding of how the simulator functioned so we could use it correctly.

Another important knowledge base for the completion of this project was **algorithm efficiency**. The animation, along with the other mathematical formulas required for this project, created computationally expensive processes. By understanding the importance of algorithm efficiency, the team was able to optimize the computational cost of the program to some extent and eliminate some excessive computation.

14.3.3. Gained Skills

The team gained several skills over the course of the project. The first was the extensive experience gained from using **C#** as the core language used in the project. Over 10,000 lines of functional code were created in both the exploration and the final product. The code required for this project gave the team experience with property declarations, event handlers, design patterns, I/O, and integration of external programs.

The team also gained extensive **XAML** and UI design experience. The team used `TabControl`s, `DockPanel`s, `StackPanel`s, `table`s, `Canvas`s, `Grid`s, `Shape`s,

Buttons, TextBoxes, and multiple other elements. The team combined these elements to design a new interface, following established design principles.

The team also gained extensive **graphic design** and **animation** skills. The team implemented several different graphical elements to display the different components of the linkage while keeping visibility and aesthetics in mind. Furthermore, the application needed to implement interactive animation that coincides with user actions.

A major portion of this project was **feature implementation**. The team quickly learned the skill set to implement a feature from concept to completion. Over the course of the project, new features were continuously added. As the team learned this skill set, our feature implementation became more rapid.

The final skill that was improved upon was **scheduling**. Over the course of the project, the team learned a great deal about implementing productive scheduling. The team organized several regular meetings with professors to discuss the application and ensure timely progress. The team also scheduled a focus group and organized a remote user evaluation. The project also required the team to conduct multiple meetings with the PMKS+ Web team and learn how to collaborate to create a better interface as a whole.

14.3.4. Gained Knowledge

The team learned much over the year-long implementation of the application. The first topic that our team learned about was **linkages**. Our team needed to gain a large amount of knowledge about linkages as well as the software that simulate them. This allowed us to become familiar with linkages and to understand what an application for linkages needed.

A second area where we gained new knowledge was **professional coding standards, such as documentation and design patterns**. These standards were important for organizing the different components of the application. Learning and applying these standards also helped the team achieve a readable code to allow further development of the application to occur more smoothly.

Another field where we gained knowledge was **user interface design**. Throughout the implementation, there were several new UI design techniques that we learned to implement a large-scale and fully functional design. Furthermore, the team encountered additional challenges with adapting the existing layout while implementing new unique features and maintaining a consistent UI with our PMKS+W counterpart. Before the implementation of new features, the team learned to create several design sketches and mockups to ensure that a well thought-out design was implemented.

The team also gained a large amount of knowledge about conducting a user evaluation and analyzing the results. The literature review taught us about multiple forms of evaluation, and the process of conducting one ourselves taught us more about online survey design. Finally, our data analysis plan gave us insight into the different ways you can analyze data and how to draw conclusions from it.

14.4. Overall Evaluation

In conclusion, our team's project, *PMKS+Desktop: Modernizing and Expanding a Legacy Silverlight Application*, is a success. Through teamwork and with our advisers, our team has succeeded in accomplishing the goals established for the MQP and the user evaluations. We have demonstrated our ability to apply our existing skills and knowledge to a task, and have learned new skills and information throughout this experience. Through our work, PMKS+Desktop is a well-designed, functional application that will succeed PMKS as a learning tool for students and will be improved upon in the future.

15. References

- Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). *User-Centered Design*. Bainbridge, W. (Ed.). *Encyclopedia of Human-Computer Interaction*. Thousand Oaks: Sage Publications, 37(4), 445-456.
- Andrews, J.; Knox, B.; Leech, B.; & Riviera, G. (2018). *Automated design of planar linkages: Slider-Crank analysis* [PDF file]. Retrieved from <https://digitalcommons.wpi.edu/mqp-all/7219/>
- Angular. (2019). *Browser support*. Retrieved from <https://angular.io/guide/browser-support>
- Appikarla, P.; Cecil, G.; Taylor, M.; & Tsiakmakis, D. (2019). *PMKS+: Recreating a legacy application* [PDF file]. Retrieved from https://automateddesign.slack.com/files/U6Y1VB8A2/FE3GE4C0K/se_for_capstones_ver1.01.pdf
- Campbell, Matthew. (2014). *PMKS: Planar mechanism kinematic simulator*. Retrieved from <https://designengrslab.github.io/PMKS/>
- Constantine, L. L.; & Lockwood, L. A. D. (1999). *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. New York, NY: ACM Press.
- Constantine, L.; & Lockwood, L. (2002). *Usage-centered engineering for web applications*. *IEEE Software*. 19. 42-50. 10.1109/52.991331.
- Dassault Systèmes. (2019). *SOLIDWORKS and SW PDM system requirements*. Retrieved from <https://www.solidworks.com/sw/support/SystemRequirements.html>

- Dayyan, M. (2008, September 17). *Analog clock in WPF*. Retrieved from <https://www.codeproject.com/Articles/29438/Analog-Clock-in-WPF>
- De George, A.; & Victor, Y. (2019). *XAML overview (WPF & .NET Core)*. Microsoft Corporation. Retrieved from <https://docs.microsoft.com/en-us/dotnet/desktop-wpf/fundamentals/xaml>
- GitHub. (2020). *GitHub.com*. Retrieved from <https://github.com/>
- Guru99. (2019). *C# tutorial for beginners: Learn in 7 days*. Retrieved from <https://www.guru99.com/c-sharp-tutorial.html>
- JnHs; Satran, M.; Jacobs, M.; Hillebrand, M.; & Koren, A. (2018). *The app certification process*. Microsoft Corporation. Retrieved from <https://docs.microsoft.com/en-us/windows/uwp/publish/the-app-certification-process>
- Microsoft Corporation. (2011). *Get Silverlight 5*. Retrieved from <https://www.microsoft.com/silverlight/>
- Microsoft Corporation. (2017). *The Model-View-ViewModel pattern*. Retrieved from <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>
- Microsoft Corporation. (2019a). *Canvas Class*. Retrieved from <https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.canvas?view=netframework-4.8>
- Microsoft Corporation. (2019b). *EllipseGeometry Class*. Retrieved from <https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.ellipsegeometry?view=netframework-4.8>

Microsoft Corporation. (2019c). *Geometry Class*. Retrieved from

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.geometry?view=netframework-4.8>

Microsoft Corporation. (2019d). *Line Class*. Retrieved from

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.shapes.line?view=netframework-4.8>

Microsoft Corporation. (2019f). *PathFigure Class*. Retrieved from

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.pathfigure?view=netframework-4.8>

Microsoft Corporation. (2019e). *QuadraticBezierSegment Class*. Retrieved from

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.quadraticbeziersegment?view=netframework-4.8>

Microsoft Corporation. (2019g). *RotateTransform Class*. Retrieved from

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.media.rotatetransform?view=netframework-4.8>

Microsoft Corporation. (2019h). *Timer Class*. Retrieved from

<https://docs.microsoft.com/en-us/dotnet/api/system.timers.timer?view=netframework-4.8>

Microsoft Corporation. (2019i). *Window.WindowState Property*. Retrieved from

https://docs.microsoft.com/en-us/dotnet/api/system.windows.window.windowstate?view=netframework-4.8#System_Windows_Window_WindowState

NetMarketShare. (2019a). *Operating system market share*. Retrieved from

<https://netmarketshare.com/operating-system-market-share.aspx>

- NetMarketShare. (2019b). *Operating system share by version*. Retrieved from <https://bit.ly/3cEeq2T>
- NextTurn; Schonning, N.; yishengjin1413; Erland; Wenzel, M.; Blome, M.; Latham, L.; Wagner, B.; Pratt, T.; Lee, D.; Hoffman, M.; Jones, M.; & Potapenko, N. (2017). *Deploying a WPF Application (WPF)*. Microsoft Corporation. Retrieved from <https://docs.microsoft.com/en-us/dotnet/framework/wpf/app-development/deploying-a-wpf-application-wpf>
- Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 152-158. Retrieved from <https://dl.acm.org/doi/pdf/10.1145/191666.191729>
- Node.js Foundation. (2019a). *About Node.js*. Retrieved from <https://nodejs.org/en/about/>
- Node.js Foundation. (2019b). *Downloads*. Retrieved from <https://nodejs.org/en/download/>
- Rothenhofer, G. (2009). *Linkages* [PowerPoint slides]. Retrieved from <https://bit.ly/2WD5Wni>
- Rubin, J.; & Chisnell, D. (2008). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, Second Edition. Indianapolis, IN: Wiley Publishing, Inc.
- Schofield, M. (2019). *Choose your app platform*. Microsoft Corporation. Retrieved from <https://docs.microsoft.com/en-us/windows/apps/desktop/choose-your-platform>
- Slack. (2020). *Slack.com*. Retrieved from <https://slack.com/>
- Stone, D.; Jarrett, C.; Woodroffe, M.; & Minocha, S. (2005). *User Interface Design and Evaluation*. San Francisco, CA: Morgan Kaufmann Publishers.

Whitney, T.; Jacobs, M.; Satran, M.; Weston, S.; Radich, Q.; Schofield, M.; JnHs; C., K.; & Das, D. (2018). *What's a Universal Windows Platform (UWP) app?* Microsoft Corporation.

Retrieved from

<https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide>

WPI Information Technology Services. (2019). *Student computer recommendations*. Retrieved from <https://its.wpi.edu/article/Student-Computer-Recommendations>

Zhang, Y.; Finger, S.; & Behrens, S. (2010). Chapter 5: Planar linkages. In *Introduction to Mechanisms* [E-Book]. Retrieved from

<https://www.cs.cmu.edu/~rapidproto/mechanisms/chpt5.html>.

Appendices

Appendix A: Notes on Linkage Simulation Applications

\App \ Feature\	MechAnalyzer	Linkage Mechanism Designer and Simulator	SAM	Working Model	MotionGen
Platform:	Windows XP and up	Windows XP and up	Windows 7 and up	Windows XP and up	Android and iOS
Cost:	Free	Free	Free demo, free 4-week full trial, purchase	Purchase	Free
How to Access:	Download link	Download link	Download link	Can be used for free in Higgins 230. Otherwise, download link	Apple App Store or Google Play Store
Custom Linkages?	No. Only a few preset linkages can be selected from the dropdown in the lower left corner	Yes	Yes	Yes	Yes, sort of (see below)
# of Steps to Create 4-Bar Linkage:	N/A	12	16	17	5

<p>Steps to Create 4-Bar Linkage:</p>	<p>None, the program starts with a 4-bar linkage displayed.</p>	<p>(1.) Select the default anchor (2.) Right-click anywhere on plane to open menu (3.) Left-click on "Add Linked Connector" (4. & 5.) Repeat steps 2 and 3. (6.) Repeat step 2. (7.) Left-click on "Anchor" (8.) Select both the anchor and the previous joint (9.) Click "Link" (10.) Right-click the original anchor joint (11.) Left-click "Rotating Input Achor" (12.) Left-click "OK"</p>	<p>(1.) Left-click "Create beam element" (2.) Left-click anywhere on the plane to create the first joint. (3.) Left-click anywhere on the plane to create the second joint. (4.) Left-click the previous joint (5.) Repeat step 3. (6. & 7.) Repeat steps 4 and 5. (8.) Left-click "Fix node in X- and/or Y- direction" (9.) Left-click the first node in the linkage (10.) Left-click diagonally from the node to make it an X- and Y-anchor. (11.) Left-click the last node in the linkage. (12.) Repeat step 10.</p>	<p>(1.) Left-click "Link" (2. & 3.) Left-click on the canvas to place the start of the link and left-click somewhere else to place the end of the link. (4.-9.) Repeat steps 1 through 3 twice more to place two more links overlapping each other in the correct positions. (10.) Left-click "Pin joint". (11.) Left-click where two links overlap to connect them. (12. & 13.) Repeat steps 10 and 11 to add a joint connecting the last link. (14. & 15.) Repeat steps 10 and 11 to connect one link to the ground link. (16.) Left-click "Motor".</p>	<p>(1.) Tap the icon with two connected circles on the top right. (2.) Tap somewhere on the grid to place the input joint. (3.) Drag your finger and lift to create the second joint. (4.) Tap somewhere on the grid to create the second anchor joint. (5.) Repeat step 3. A triangular link with a POI will be created between the two free links.</p>
--	---	--	---	---	--

			<p>(13.) Left-click "Angle Input Motion"</p> <p>(14.) Repeat step 9.</p> <p>(15.) Left-click "Add"</p> <p>(16.) Left-click "OK"</p>	<p>(17.) Left-click on the non-grounded link to add a motor and connect the link to the ground link.</p>	
Change Shape of Link	Can only change length in the "Parameters" menu in the bottom center	Left-click a joint to move it anywhere, and the link's length/angle will automatically adjust.	Select the "Move Node" tool, then left-click a node, move the mouse to the desired location, then left-click again. The link's length/angle will automatically adjust.	Not once the link is created. You can, however, move the locations of the joints on the link.	Can only drag the joints to change the position/length of the links
Joints in Middle of Link?	Yes, but not creatable by the user	No (you can make a triangle, then put the third joint between the other two, though)	No (you can make a triangle, then put the third joint between the other two, though)	Yes. Drag a point/joint onto the link in the correct location.	No
Dedicated Polygonal Links?	Yes, but not creatable by the user	Yes. Create as many joints as you want, then select them all and click "Combine"	No. You can simulate larger polygons using triangles, however.	Yes. Use any of the shape tools to create a link of any shape you wish.	Yes, but not creatable by the user

Arbitrary Joint Placement in Poly Links?	No	No. Joints may be placed in the center of a poly link at arbitrary points (create a Joint, drag it over the poly link, select all the joints, then click "combine"), but the poly links still have joints at all corners.	N/A, no poly links	Yes. Drag a point/joint onto the link in the correct location.	N/A, unable to place individual joints.
Sliding Links?	Yes, but not creatable by the user	Yes	Yes	Yes	Yes.
Limit Angle of Rotation?	No	Yes. Right-click the input joint, then change the "Oscillation Limit Angle" and "Oscillation Start Angle" attributes	Potentially. While editing the input motion attributes, you can change the Motion "Value" from 360 to something else. However, I could not get this to work. Additionally, this would not let you set the angle bounds, just the angle difference from the starting angle.	No	No
Infinite Linkage Rotation?	No. Number of rotations must be specified, but it affects the simulation and is honestly confusing.	Yes	No. The linkage rotates one direction, then reverses direction at the end.	Yes	Yes.

Create Forces?	No. Gravity acting on linkage is calculated, however.	No	Yes. Do Loads > Force, then click a Joint. It will bring up a menu where you can specify magnitude, angle, etc. as a function of time or another property.	Yes. Select the Force icon from the left-side menu, left-click on a part of the linkage to select where the force applies (the head), then left-click anywhere else to specify the direction and magnitude (the tail). Gravity is automatically simulated and unattached objects will fall when the simulation is run.	No.
Display Movement Paths?	Only for certain joints.	Yes. Must manually enable it for each joint by right-clicking the joint, then click "Draw Motion Path"	Yes. Must manually enable it for each joint by right-clicking the joint, then Node Properties > Display > Path.	Yes. Must manually enable it for each joint by right-clicking the joint, then Appearance > Tack	Yes.
Export Data?	No	Yes. File > Save As... or File > Export > Export to DXF.../Export Motion Paths...	Yes. File > Save As.../Export DXF... or Results > Export...	Yes. File > Save As...	Yes. Tap the box with an upwards arrow at the top menu. Then select PDF or XML.
Import Linkage?	No	Yes. File > Open...	Yes. File > Open... or File > Import DXF...	Yes. File > Open...	No.

Take Photos/Videos?	No	Yes. To print a picture, you can do Printing > Print or File > Print. To save a picture or video, do File > Export > Export Video.../Export Image...	Yes. For pictures, do File > Print... or File > Display > Mechanism to Clipboard or File > Results > Graph to Clipboard. For videos, do File > Display > Create Movie...	Unknown	No.
Display Grid?	No	Yes. On by default. Home > View > Details > Show Grid to toggle on/off. Details > Edit Grid to edit.	Yes. On by default. File > Preferences... > Display > Grid > None to turn it off.	Yes. Off by default. To display gridlines, do View > Workspace... > Navigation > Grid Lines	Yes. On by default. To turn off the grid, tap the gears at the top right, tap Settings, then turn off "Display Grid"
Display Axes on Grid?	No	No	No	Yes. Off by default. To display axes, do View > Workspace... > Navigation > X,Y Axes	Yes. No option to turn off.
Samples?	The entire program.	Yes. File > Samples	Yes. File > Wizard > 4 Bar Mechanism will allow you to create a 4-bar linkage with default dimensions or dimensions you specify.	No.	Yes. Tap the hamburger bar on the top left, then tap "Examples", then select an example.

Graph View?	Yes. Can add displays of position, velocity, acceleration, and force for each joint and link	No	Yes. Right-click the graph on the left, then select "Select Curves". Then click any linkage element on the right to select with attributes you want to graph.	Yes. Right-click an object, then Measure > Select the measure you want to graph.	No.
Camera Controls	Zoom in, zoom out, orbit, pan, select view buttons. When orbiting or panning, use left-click on the view screen.	Right-click to pan, scroll wheel to zoom in/zoom out. Zoom in/zoom out and center screen buttons.	Scroll wheel to zoom in/zoom out. Zoom out, undo zoom, pan, center contents, and zoom on selection buttons.	Scroll wheel to zoom out. Zoom in, zoom out, zoom to extents, pan, and zoom window buttons.	Pinch/un-pinch to zoom in/zoom out. Button to put screen back to original zoom level/position. Pan with two fingers.
About Page?	Yes	Yes	Yes	Yes	Yes.
Help Page?	Yes	Yes	Yes	Yes	Yes.

<p>Positive Thoughts and Features:</p>	<ul style="list-style-type: none"> ● The controls for the camera have appropriate icons and helpful hover-over text. ● There are multiple methods of movement analysis available. 	<ul style="list-style-type: none"> ● UI is smoother and more friendly than MechAnalyzer; reminiscent of Word/Excel ● Labels are more specific than MechAnalyzer ● Parts List that displays all links at relative sizes next to each other with labels 	<ul style="list-style-type: none"> ● Both a graph and the mechanism are displayed simultaneously ● The click-and-drag creation of links works well. ● Status messages, hover text, and a description are displayed along the bottom of the screen. 	<ul style="list-style-type: none"> ● The UI is colorful and engaging. ● All buttons have hover text and a description on the bottom of the window ● Linkages with multiple degrees of freedom can be ran 	<ul style="list-style-type: none"> ● You can take pictures/use pictures from the Photo Album and set that as the grid's background ● You can turn on/off the view of the timeline and drag to a specific moment in the linkage's rotation. ● Two movement paths are displayed, and a certain option allows you to switch between them. Not quite sure what the point is, but it's interesting ● The UI is clean, modern, and engaging
---	---	--	---	---	---

<p>Negative Thoughts and Features:</p>	<ul style="list-style-type: none"> ● Overall, the window seems plain and square and antiquated. The links are brightly colored and have no outline, thereby making the structure difficult to see for me. ● The slider link (seen in yellow behind the green triangle) looks like a normal link with no joints, making it difficult to understand for a new user like me. ● Labels of the controls are vague; for example, the menu section containing the lengths of the links is simply called "Parameters" ● Each length is labelled "L1", "L2", etc. but the links themselves have no label, requiring trial-and-error to change the right one; that is, until you discover that 	<ul style="list-style-type: none"> ● At the start of the program, a lone ground joint appears: This isn't a very interesting display, and it isn't marked as the input either, requiring you to set it as the input even after creating a linkage ● Linkage initially shows up small ● No coordinate plane makes the space feel empty and unfriendly, and feels like you could lose where the center is ● Option for coordinate plane is hidden within a submenu ● The linkage cannot be edited while it is moving; it must be stopped, edited, and then restarted. ● Not clear how to create additional links/joints to new user 	<ul style="list-style-type: none"> ● Very buggy in general. I managed to get an access violation just trying to make a linkage... ● Colors are rather flat, and the UI is square. Not very engaging. ● Slightly confusing to determine how to create a link. ● Very confusing to determine how to make joints anchor joints and how to choose an input. ● The sides of the window feel underused. 	<ul style="list-style-type: none"> ● The style of the UI is outdated. ● The menu sections on the left are unlabeled. ● It isn't clear how to connect joints to create a link. ● The ability to show axes and grid lines are hidden in multiple submenus ● To create multiple joints/rectangles/etc, you must repeatedly select the tool after creating one. ● Sometimes an apparently working linkage doesn't move when the animation is run, and no status/error message is displayed. 	<ul style="list-style-type: none"> ● The UI in general feels large and cumbersome. When multiple options are open at the same time, they overlap and make choosing the option you want difficult.
---	--	---	--	---	--

	<p>the links are only labelled in the tiny diagram at the bottom. You can click it to zoom in though.</p> <ul style="list-style-type: none"> ● It is not clear that you need to manually run an analysis before you can see the linkage rotate ● The analysis options are abbreviated and have no hover text to explain ● What I assume are the minimum and maximum rotation angles of the input joint are simply labelled “minimum value” and “maximum value” and cannot be changed. ● The “time” and “steps” values don’t seem to be the time/steps per rotation but rather for the entire analysis, causing strange behavior when just trying to rotate the linkage multiple times. 	<ul style="list-style-type: none"> ● Sides/bottom of the screen feel underused ● Can only see the parameters for one joint at a time ● Linkages that can't complete a revolution simply stop and show an error message. ● You can view movement paths, but they are drawn as the link rotates so it is still unclear if the linkage will/won’t fail to rotate. 			
--	--	--	--	--	--

Appendix B: User Evaluation Script

PMKS+ Desktop Evaluation

Welcome to the PMKS+ Desktop User Evaluation.

* Required

1. Email address *

Skip to section 1 (Consent Form) Skip to section 2 (Consent Form)

Consent Form

Thank you for considering participating in this study. Research is being conducted to evaluate the effectiveness of the linkage analysis software, PMKS+ Desktop. This research is being conducted by the WPI MQP CS group of Peter Prygocki and Fabian Gaziano. The project is advised by Professor David Brown (CS) and Professor Pradeep Radhakrishnan (ME).

Over the next few minutes, you will be asked to complete tasks using PMKS+ Desktop. Following the use of the software, you will be asked some questions about your experience and thoughts, and your answers noted.

There are no known risks associated with participating in this study.

Please remember that your participation in this research is voluntary, confidential and anonymous. Only the researchers and project advisers named above will have access to the data collected. You may withdraw your consent to participate at any time without any penalty. This is a completely voluntary research project so you may stop at any time.

If you wish to obtain further information about this study, you may obtain a more detailed explanation of its goals after your participation has finished.

Demographic Survey

Please answer these questions before completing any tasks.

2. How experienced are you with creating linkages in Working Model? *

Mark only one oval.

- Not At All Experienced
- Moderately Experienced
- Very Experienced

3. How experienced are you with creating linkages in PMKS Silverlight? *

Mark only one oval.

- Not At All Experienced
 Moderately Experienced
 Very Experienced

4. How experienced are you with creating linkages in PMKS+ Web? *

Mark only one oval.

- Not At All Experienced
 Moderately Experienced
 Very Experienced

5. How experienced are you with creating linkages in SolidWorks? *

Mark only one oval.

- Not At All Experienced
 Moderately Experienced
 Very Experienced

6. Please select which relevant courses you have taken. *

Check all that apply.

- Introduction to Dynamic Systems
 Kinematics of Mechanisms
 Dynamics of Mechanisms
 Advanced Engineering Design
 Modeling and Analysis of Mechatronic Systems

Other: _____

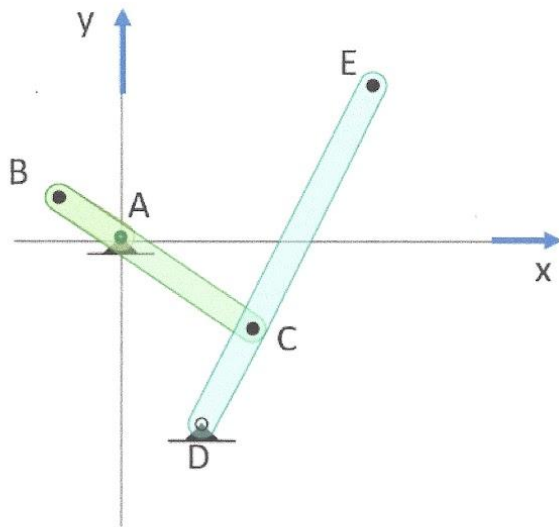
PMKS+
Desktop
Tasks

In this section, you will be carrying out the following tasks in PMKS+ Desktop. We recommend that you use a mouse if available.

Create a Four-Bar Linkage

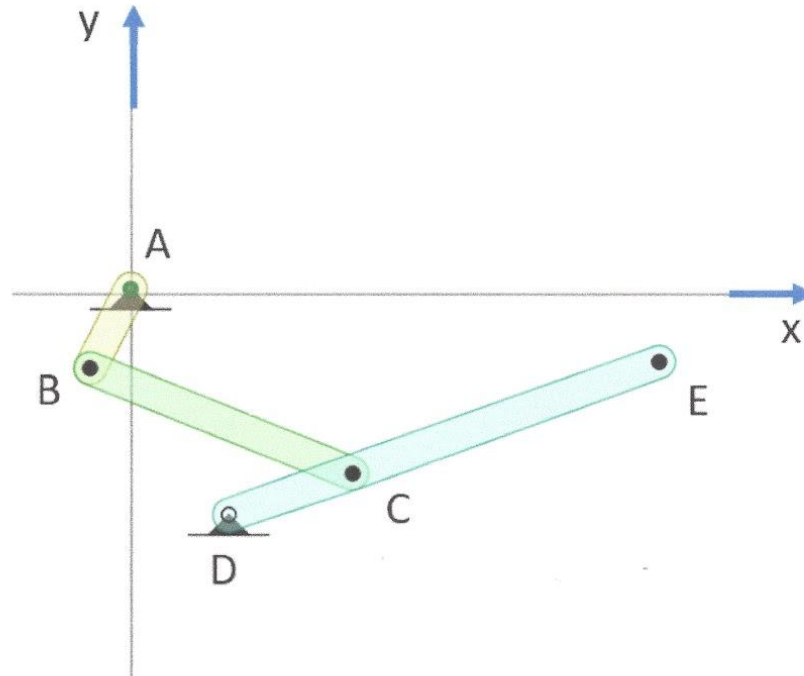
Create the four-bar linkage shown below with coordinates. The input is at A and rotates at 30 RPM in the counterclockwise direction. The links are AB, BC and DCE (ground link is the fourth link). All the joints are revolute joints. The length unit is cm.

Four-Bar Linkage



A (-0.058, 0.403)
B (-5.508, 3.880)
C (11.581, -7.623)
D (7.100, -16.000)
E (22.174, 13.676)

Another position of the same linkage is shown below for clarity:



7. Please Upload a Screenshot of the four-bar linkage created in PMKS+ Desktop *

Files submitted:

Animate the Linkage

1. Play the animation clockwise.
2. Reverse the direction of the input joint's animation.
3. Set the animation speed to fast (x3).
4. Pause the animation.
5. Reset the animation to the beginning with the Stop Button.

Graph and Export the Angular Velocity of Link BC

First, open the graph for the angular velocity over time for the link EF. Then, save the plot as an image.

8. Please Upload the Angular Velocity of Link BC *

Files submitted:

Export to SolidWorks

Please generate a CAD assembly of the linkage.

9. Please Upload a Screenshot of the CAD Assembly *

Files submitted:

Skip to question 10

Follow-Up Questions After
PMKS+ Desktop

Please answer these questions immediately after completing the PMKS+ Desktop tasks.

10. How quick was the linkage creation and editing process? *

Mark only one oval.

- Quicker than expected
- About the same as expected
- Slower than expected

11. What did you find difficult in PMKS+ Desktop? *

12. What did you feel was easy in PMKS+ Desktop ? *

13. What did you like about PMKS+ Desktop? *

14. What did you dislike about PMKS+ Desktop? *

Concluding
Questions

After performing the described tasks, please answer the following questions.

15. Please list the top 3 features that you used in PMKS+ Desktop. *

16. If you want, please explain your choices.

17. Please rate your overall satisfaction with PMKS+ Desktop. *

Mark only one oval.

1 2 3 4 5

Very Unsatisfied Very Satisfied

18. Please write any comments you have on how PMKS+ Desktop compares to Working Model, PMKS Silverlight or PMKS+ Web. *

19. Please list any recommendations for features and improvements you would like to see in PMKS+ Desktop. *

Thanks for filling out the survey. Your responses will be very useful to improve the software

This content is neither created nor endorsed by Google.



Appendix C: Participant Responses to Open Response Questions

Appendix C.1.: What Was Difficult in PMKS+Desktop

Participant #:	What did you find difficult in PMKS+ Desktop?
1	Very easy
2	Solidworks export crashed the program. The bar when animation run's turns white. Scroll wheel wont zoom in after loading lineage. Adding link creates a random link to a specific position which need to be dragged into place afterwards. Editing locations of joints was also not very easy.
3	UI was difficult to learn at first. Took a few minutes of clicking around to figure out what I was able to do.
4	Using Notepad to edit the joint locations instead of entering the location in the application itself
5	Figuring out how to generate the angular velocity graph was harder than expected.
6	I found it difficult to use the Table to edit the links. The creation of links was also not very easy. It is easier to create these things in working model.
7	If there was a function to edit the position of linkages within the software vs having to open it up in notepad, would make it a lot easier to understand since edits would be able to be viewed on the fly. Also the fact that you cant have the tutorial open on the side and do things in the software at the same time really bugged me.
8	Lack of snaps to linear from triangular with and ternary links, scroll speed was too slow, deleting links, editing links to an exact size. I can't edit anything while a simulation is paused, but there's no obvious indication that it is paused. The whole row disappears when editing link location.

9	The most frustrating part was that the application had a tendency to crash. When the program stayed open, it was very simple and understandable, but the crashes were annoying. The most difficult part was finding some of the options and settings, but they were fairly intuitive. They were not that difficult to find.
10	Instructions on how to make links specific lengths or pins at certain points were not given. It was unclear how to find how to export the velocity data. It was also unclear how to make each link, especially how to make the link with 3 points. I also had issues with the application crashing/closing down occasionally when I performed some actions, such as modifying the positions of link endpoints in the table or clicking on the Export to Solidworks button on the toolbar.
11	Finding things without instruction was difficult. I looked for linkage analysis in the analysis tab, not knowing I had to right click the link itself to get the graph. Also exporting to solidworks was an issue because I was not aware of how the saving process worked. Making the folder selection more clear or happen by default would be easier.
12	I couldn't figure out how to see the graphs of the data.
13	It took a while to figure out how to make a link with multiple joints.
14	Although many of the features were easy to use, there were subtleties that I was unable to figure out without help. Also, my machine crashed more than once while I tried to create the assembly.
15	It was a little difficult to figure out how to make the middle link and where to find the graph.
16	Selecting the joints was not as fast as I thought and I realized I had to hover my mouse there for a second. Entering the points in the notepad was more difficult than it needs to be, I deleted more than I was supposed to by accident and had to re-open the notepad. Exported data is too difficult to read on notepad without the lines. Not sure if I can add lines or not but I had trouble reading which data went to the corresponding links and joints

17	I was not sure how to work most of the program. I could make the linkage, but it took a while. I could not figure out how to add any forces and thus that's why I did not attach a graph or CAD assembly screenshot. The animation would not work and it was hard to find out how to make the whole program work.
18	There were many times where the software would get jammed and stop responding. I needed to restart it many times throughout the process.
19	I had some troubles creating the given link and whenever I tried to get the angular velocity of the link the program crashed on me. Therefore to get the Angular Velocity, I created my own link given in the power point.
20	There is no undo button, so when I made a few mistakes it required me to start from the beginning. Also I found it difficult to move the intersection between two links, so I had to build the linkage backwards for the link to connect to the center of the final grounded link.
21	The tutorial is too simple not really help a lot
22	Export had some issues as is shown in my screenshot of assembly but the triangular linkage present in the powerpoint converted over amazingly.
23	The program generally crashed after each step so I had to save my linkage after each step in case of a crash.
24	Moderate or easy to use with the introduction ppt.
25	I was only able to right-click on already existing joints to create a linkage. Right-clicking on the grid gave me no options. Deleting the already existing joint at (0,0) would close the program. Exporting the file to Solidworks was a difficult process which I was not able to complete. I could not delete any linkages or joints. No undo function. Show Joint Kinematics caused a crash. Position should be easy to place on integers.
26	Crashes and creating a ternary link
27	I was a bit confused about deleting linkages (forced me to restart) and trying to add the joint in the middle of the link. After I figured out how to do that, it was pretty smooth sailing from there on out.

28	I couldn't get the files to export properly into Solidworks, I kept trying but I could not end up getting it to work.
29	Very intuitive i did not have much difficulty at all using the software.
30	IT crashed 4 tiem and some of the features were less than clear on their use
31	The linkage building process is not super robust, it does not allow for creation of external bars. I could not create a bar and then connect it later, you have to start with the connection. It is not very natural especially if you are used to something like solidworks, but I got the hang of it.
32	There are many bugs which will lead to the application to close without saving: If you click onto another box after clicking on the first for editing in table Pressing tab in table (Expected this functionality to switch between values similar to Excel) Attempting to relabel a link in table Can not look at tutorial and edit the system at the same time have to reopen every time for tutorial steps
33	There were frequent crashes and I had a difficult time exporting the final version to solid works fully assembled
34	It was slightly annoying to have to open the file in notepad to give precise coordinates. It was also not immediately obvious that I needed to right-click to show the linkage angular velocity.
35	Editing the linkage through the table doesn't work. For example, I can't change joint locations or toggle which link is the input link. Also, the program seems to freeze just about every time I try to create a 6-bar linkage.

Appendix C.2.: What Was Easy in PMKS+Desktop

Participant #:	What did you feel was easy in PMKS+Desktop?
1	Everything
2	Play, pause, reset and delete linkage were quite easy to use. Same with export bars.
3	Creating linkages and graphical information.
4	Finding angular velocity plot
5	Running the simulation was relatively easy.
6	The animations were easy to use. It was easy to change the direction of motion. It was also easy to pull up motion diagrams from the links.
7	Playing the model was straightforward as well as adding linkages once I understood how the function worked.
8	quickly building linkages. Exporting Solidworks files. Adding links is smooth and makes sense.
9	The interface was very intuitive and the entire process from creating the linkages to exporting to solidworks was extremely quick and simple. The program functionality was really solid. The best part was the graphing and exporting capabilities. By clicking single buttons it immediately generated very useful charts and seamlessly generated a CAD file.
10	Once I figured out how to perform certain functions, such as making a link or exporting velocity data, it was relatively simple to repeat the action. The main issues were not in performing the actions, they were in learning how to do so.
11	Editing link size and position was very easy. Entering coordinates into the PMKS+ is much quicker than trying to do the same in solidworks.
12	Creating the linkage in PMKS+ was very easy to do.
13	It was really easy to quickly sketch the linkage then add the dimensions after.
14	Creating the linkage itself was not difficult
15	Putting in the datapoints and linkages and mounts.
16	Making links and joints was super easy and exporting to solidworks was effortless. Graphs were easy to read and interpret.

17	I could figure out how to make the first input joint and add links and toggle to ground.
18	Retrieving the analysis and exporting to Solidworks was very easy.
19	Creating links and adding joints. I have used linkage soft wares before like Norton Linkages, but this is the first one with a nice GUI.
20	The plugging in coordinates and auto generation of motion and links was particularly interesting and easy. It is also super convenient transferring it over to AutoCAD
21	its fast to make the linkage system with a bit experience
22	Actual creation of links, and solidworks export. Its amazing that it creates it so fast!
23	The program navigation was quite easy.
24	Adding multiple linkages on a same bar is much easier than that doing that on Solidworks.
25	It was easy to create linkages and joints. Animation was easy to do. Finding angular velocity was easy.
26	Creating normal linkages and viewing the amount of degrees that those links can move
27	It was very easy to go from a base linkage drawing to a full CAD assembly. This would have taken probably a lot longer if I was simply going straight to CAD.
28	I felt creating the linkage was relatively easy once you followed the instructions. The instructions were straight forward and clear on that part.
29	I found actually creating the linkage in the software to be very easy coming from a cad background. I also found the conversion to CAD to be incredibly easy and much quicker than it would have taken to do just in solidworks.
30	quickly editing the links and setting up motion studies
31	Obtaining the velocity and acceleration graphs is very easy, and animating the motion is easy aswell.
32	Very easy to add components and the analysis was literally at the click of a button! Exporting to Solidworks had issues for me but worked very well for the tutorial

33	The program is very simplistic. Being able to access most of the options with right click made it very easy to create the linkage system.
34	This was a faster way of creating a SolidWorks Linkage.
35	The desktop version seems to work more reliably overall than the previous versions.

Appendix C.3.: What Was Liked in PMKS+Desktop

Participant #:	What did you like about PMKS+ Desktop?
1	Everything
2	The angler and velocity acceleration chart.
3	Simple and easy to use once I learned how to use it.
4	Plots are easy to find
5	The easy access to point coordinates in the tables
6	It made a solid works cad assembly from the link i made inside of the program. i thought that was pretty cool.
7	The software layout is clean and its easy to get to all the functions since there either on the main screen or under only a single menu. It also worked very smoothly.
8	The DOF display, the integrated materials option, solid works export
9	The best part of the program was the SolidWorks export function. Being able to create the linkage in just a few minutes and then immediately generate a usable and functional CAD model would be incredibly useful in a lot of situations.
10	I liked how it was easy to create links and export data and Solidworks files once I figured out how to do so. The Solidworks export I think could be very beneficial if I needed a linkage for a CAD design. It could be cool to see some sort of automatic update to the associated Solidworks file if the linkage is modified in PMKS+ (similar to the automatic updating used with Solidworks CAM).
11	i really enjoy the animation preview to make sure the linkage does the intended motion before exporting to solidworks.
12	I really liked the ability to open the linkage in note pad and edit the placement of joints to get links the exact size that you want.
13	I like the detail in the CAD model it creates, I was expecting a sketch block diagram.
14	The design is very simple
15	It was cool watching the simulation.

16	I like the idea behind exporting it into a manufacturable linkage like i've seen in class. Easy to work with. I really liked the graphs for each link and joint. I like how easy to use this software is. The vector loops and position paths are really nice how they auto-update.
17	I think it has the potential to be a useful and easy to use program, but there needs to be more steps in tutorials and more tutorials to explain everything past making a simple linkage.
18	I like how it was able to fully export and assemble in solidworks
19	I liked how you could get data for different links and export it as a Solid Works file. I like how the software created the Solid Works Assembly for me, as a CS major and a person who has created applications and used Solid Works, I know this part would have been really hard and I respect the team's efforts on this.
20	The ability to transfer the whole design to autoCAD and also the ability to edit the locations freehand or plug in absolute numbers for where the joints would be.
21	it fast and after learn how to use it it will be easy and to use
22	It was very intuitive and a great tool!
23	Easy navigation, good user interface
24	It can save as excel files and editing tables on excel.
25	This program has potential for analyzing a 2d assembly. It is simple to use and would be a good program for 1st or 2nd year students to understand the forces behind linkages.
26	User Interface is a lot nicer and more friendly compared to PMKS Silverlight
27	The ease of transitioning what you have made on it to solidworks.
28	Nothing in particular, I haven't used anything like that before so I don't really have anything to compare it to.
29	Definitely my favorite part was the conversion to CAD,this saves so much time when one needs to generate masses or moments of inertia from a cad model for a linkage.
30	The conversion to solidworks was really nice. Made up for the fact that it crashed 4 time
31	I liked the straightforward export to solidworks feature. Tt took a couple tries, but when it works, it is a nice feature.

32	Autozoom function is very helpful. The UI is very easy to navigate. Appreciate the breakdown of different analyses, could possibly switch to drop down list?
33	I like that it animates the movement of the linkage system. Having the ability to see the forces on each joint at every point in its movement cycle is also very useful.
34	This did not require a lot of experience in SolidWorks to create something.
35	The UI is very clean and the learning curve is low. I'm very impressed by the export to solidworks feature.

Appendix C.4.: What Was Disliked in PMKS+Desktop

Participant #:	What did you dislike about PMKS+ Desktop?
1	Nothing, loved it
2	Creating linkage feels a little cluttered process.
3	Program would crash if I clicked on too many menus or just from clicking around.
4	interface is not intuitive. Tutorial needs more thorough explanation and accurate naming
5	The format of the file constantly wanted me to open it with excel.
6	It was very clunky. Some of the controls were unintuitive as one needed to right click to perform an action. The top bar could be used when clicking on a link to highlight what actions a user can take.
7	Zoom wheel function was slow but using the buttons on the screen are quicker but would like to see the scroll wheel zoom be quicker. As well the maximum outward zoom is not enough and could barley fit the whole assembly in the page. Also it overall looks a bit bland but functional.
8	I want a way to edit individual links numerically. Adding ternary links with a center hinge was not intuitive.
9	The only thing that was annoying was the crashes. Outside of that, the functionality was fairly good. There were a few things that seemed to be slow or maybe glitchy but thats just small problems.
10	Lack of instructions and usable Help section was frustrating.
11	I do not like the menu system and save system. If the program was less particular about where menu options are located and where/what file type things are saved as.
12	I couldn't figure out how to look at the graphs that should have been created. Whenever I clicked on a button in the analysis tab it said "this feature is yet to be implemented".
13	The CW, CCW arrow is inverted from the way the simulation actually moves. Sometimes when you double click on the table the program crashes.

14	Some of the features were very visible and intuitive to use. Some of them seemed to be hidden and I did not know how to use them without help.
15	It kept crashing in the middle of a task.
16	I wish there was an undo option, multiple times I moved a link and had to go back into the notepad to set my desired coordinates. I didn't like that the data was not exported to an excel file. I also didn't like that I was unable to use the software with the tutorial open. I also think the tutorial should outline how to export to solidworks like you did in your powerpoint. I also didn't like that if you create a joint in space and move your cursor off of it, it disappears.
17	The tutorials and help buttons were not very useful. I was very confused and the program would crash everytime I clicked a new button. I also could barely figure out how to undo anything or redo etc.
18	It crashed a lot, the linkage building was not intuitive.
19	The Zoom out and Zoom in the application. Its not responsive to the scroll bar and it would have been nice to have a Ctrl + Z option. Moreover the program kept crashing on me when I tried to get the Angular Velocity of the given link and the Solid Works model of the link.
20	The dislikes are just the glitches and issues right now. There were a few crashes when grounding joints, and certain dropdown menus still don't lead anywhere (as is to be expected). It seems super promising however.
21	the buggy , the program keep crashing it crashed more than 20 times when before I finally finished every thing
22	I wish the linkages were labeled in the software grid (a,b,c,d) to make changing the coordinates easier in a more complex linkage.
23	If you forget to save it is likely you will lose all your progress due to crashing.
24	I was having issues with assembly. I couldn't have an assembly with all parts in it for two times. I think it might be because that the program was stopped during outputting process by unstable connections. Since I am using remote desktop, it may increase the chance of that.

25	There are still many bugs/crashes/ease of life improvements which need to be addressed. My main concern was not being able to create linkages from right clicking anywhere on the grid. Once all the software is polished, it will be a good introduction to linkages.
26	Mainly just the crashes
27	I had to restart a few times because I couldn't figure out how to undo or delete a link.
28	Nothing in particular.
29	Using notepad to change the linkages was a little odd i would have liked a feature in the PMKS software to do so.
30	Its a really janky user interface. Sometime is would click on joins and not be able to edit them or the option tap wouldn't come up, also I think it would be nice to be able to insert links and then attach them. So you dont have to build off of them. over all you really need to improve the UI then this software will be great.
31	I disliked the method of inputting dimensions. It is inconvenient to have to use notepad. Also I wish there was a better way of knowing which joint is which.
32	Wish there was a way to switch modes for editing because if I want to create and move them around manually, there is currently no way to do that without pausing and playing at the right position. Can zoom out but not in with scroll wheel. Linkages and About tabs do not work. Wish there was a way to set position of your cursor when making the link that way you do not have to edit it in the table everytime.
33	The tutorial section should have more details. While the program is simple in hindsight it was hard to do simple things like export to solidworks.
34	This is not directly available for Mac.
35	Zooming out with the scroll wheel works as expected, however zooming out does not.

Appendix C.5.: Top 3 Features in PMKS+Desktop

Participant #:	Please list the top 3 features that you used in PMKS+ Desktop.	If you want, please explain your choices.
1	Showing graphs of linkage behaviors, toggle the curve that a joint would travel, export onto SolidWorks	
2	Left click and create linkage, drag joint to different location, and edit joint location manually via the joints chart.	These are the key actions needed to create any linkage in the application.
3	Help tab, Add linkage, movement interface	Help tab was useful when learning how to use the program
4	Plots, Adding joint to a link, Link animation	
5	Linkage creation, Toggle ground, x2 and x3 speeds	I like that these features were simple to figure out and that you could adjust the playback speed of the simulation without changing the speed of rotation which would alter your analysis results.
6	Creating links, Analyzing links, Looking at the links motion	
7	The export to solidworks function is really nice. The graph showing the forces on links is really nice, and the ability to change the speed of the the links rotation is really nice as well.	The export to solidworks function makes it really easy to visualize the links in a 3D space, The graphs showing the forces on links based on size makes it easy to understand where the highest forces of a certain design would occur and at which point which would make troubleshooting really easy, and the change of speed function makes it easy to visualize how the links would work at a greater or lesser speed like they would in real life.
8	Solid works export, material and stress analysis,	

9	The SolidWorks export, the graphing function, the table function	All of these functions are extremely useful and simple to use. The tables allow the linkages to be completed in just a few minutes and then the analysis and export of the linkages is instant with the touch of a button.
10	The table of joint info was useful, exporting the velocity data of a link could be useful for reporting purposes, and exporting the linkage to Solidworks could help with design work.	
11	Adding links to make a linkage system, Creating a velocity graph of a link, exporting to solidworks.	
12	Animations, editing lengths in notepad, creating linkage	
13	The tables, Clicking and dragging points, and export to SolidWorks	
14	Create link, toggle ground, add force	
15	Datapoints, simulating, and linkages	
16	Export to Solidworks, Linkage Creation (Joints and Links), Data Exporting and Graph Display	Exporting to solidworks is a wonderful feature, the linkage creation by right clicking is easy to learn and use. The graphs and data are perfect for analysis on linkages and save a ton of time.
17	create joint, create link, toggle ground	
18	Solidworks export, kinematic analysis, motion simulation	
19	SolidWorks Export, Creating Links and Toggling Ground	
20	1) Create Link. 2) the editing table 3) motion simulation	These were just what came up the most in the process of creating this linkage
21	Its fast, It easy to use , and the export to solid-work is really nice	
22	Analysis of linkage members, exporting to solidworks, and velocity tables	
23	Kinematics/force analysis, ability to export to Solidworks, built in tutorial	

24	Show tables'; 'Export to Solidworks'; 'Tutorial'	
25	Creating Joints/Linkages, Analyzing angular velocity, locking joint to ground	The features surrounding creating joints and linkages worked well and was easy to use. Analyzing angular velocity of linkages was easy to find and read.
26	Export to solidworks, the kinematic analysis, and the UI of the program	My export to Solidworks crashes everytime but I think it is a useful feature when it works
27	I used the table edit function for the position, export to CAD feature, and the kinematic analysis feature on the links	I used the table to edit the position of each joint, the CAD feature so I can use the assembly on solidworks, and the kinematic analysis feature to quickly calculate the angular velocity.
28	Export to Solidworks, create link, create joint	Export to solidworks because I tried very many times to get it to work, but I couldn't get it to work
29	conversion to CAD, ability to make joints, angular velocity graph generation of linkages	I have already explained the CAD choice but i found the joint creation to be easy and enjoyable, and the angular velocity graph generations is incredibly helpful when checking ones answer or even generating one.
30	link creation join creation and the edit table	
31	Export to Solidworks, Show Link Kinematics, Play Animation	

32	Link Kinematics, Help Button, Autofocus Button	Link Kinematics I hope will developed further because seeing stress testing against maybe different types materials would be extremely helpful for anyone prototyping the system, but as of right now it is very easy to observe the acceleration graph using the animation buttons. In the future could see possibility of loading Matlab script or some kind of executable for calculated motions. Help button has been designed very well but maybe use different thickness border to make it easier to see/click?
33	Create Joint, Toggle Ground, Analyse angular forces	
34	1. Conversion to notepad for easier editing. 2. Keeping track of data in motion. 3. Exporting to SolidWorks	The third feature makes it much easier to create certain structures in SolidWorks.
35	Solidworks export, link kinematics export, link creation	

Appendix C.6.: Comparison of PMKS+Desktop to Other Software

Participant #:	Please write any comments you have on how PMKS+ Desktop compares to Working Model, PMKS Silverlight or PMKS+ Web.
1	PMKS+ Desktop > PMKS+ Web > PMKS Silverlight > Working Model
2	It is very slimier to PMKS+ Web.
3	Working model has more features.
4	N/A
5	NA
6	It is more friendly then working model as it took me less time to get used to it.
7	I have not used any of the other programs so I have no comments on how they compare.
8	PMKS+ is cleaner and simpler, but has fewer features.
9	I'm not familiar with any of these programs.
10	I have not used any of these other applications, but PMKS+ Desktop could be very useful once it is complete. I would definitely use it once it is complete.
11	I do not have experience with any of these. It was much quicker than building the linkage in solidworks, however.
12	N/A
13	The export to SolidWorks is a great feature
14	more universal
15	I haven't used anything else.
16	Needs bug fixing to be able to compete, but definitely has a fighting chance against Working Model. I've never used silverlight and I've only experimented with the Web version, but this is way more convenient and new-user friendly.
17	I dont have any experience with those other programs.
18	It is very similar but a bit more simple.
19	PMKS Desktop compares well to PMKS+ Web. It has a more responsive GUI and it much more easy to use than PMKS+ Web. I had a lot troubles using PMKS+ Web and therefore like the fact that there is a much better version available, however it would preferable if the features and the nice UI of PMKS+ Desktop was available on the Web as people generally prefer to not download applications.

20	Unfamiliar with the exact setup of these. It was far easier than setting it up by hand in solidworks however.
21	It is easier to use and faster
22	Much better than web, i haven't used the others.
23	I have no experience with the other three softwares.
24	I haven't use the other softwares before.
25	None.
26	It functions as a more user friendly experience compared to the alternative linkage design software
27	It was pretty easy to use after I got the hang of the program. There was not a very steep learning curve when compared to a lot of other similar programs.
28	I don't know seeing as though I haven't used Working Model or PMKS Silverlight or PMKS+ Web.
29	I have no comment
30	I cant comment too much on those as I do not have the XP in them
31	I have no experience with other softwares.
32	Definitely has the potential to be much better than Working Model and is already significantly less buggier than the Web version. Also had some trouble
33	Never used these
34	I have little experience with the others.
25	So far, this program is WAY easier to use than Working Model and PMKS Silverlight. I haven't used PMKS+ Web yet. That said, there are some significant issues which I highlighted in other comments. My satisfaction rating will be much higher once those are addressed.

Appendix C.7.: Recommendations for Improving PMKS+Desktop

Participant #:	Please list any recommendations for features and improvements you would like to see in PMKS+ Desktop.
1	Be able to implement gears onto mechanism
2	Please allow for deleting link/joint option. Allow for creating link from selected joint to desired position directly instead of dragging it afterwards. There is a bug when editing the joints table where clicking a field clear's the row while editing. Links table is empty.
3	File export to Solidworks did not work for me.
4	Solidworks compatibility
5	An easy way to create a new blank file.
6	Please fix the table. If it worked I think it could be very powerful. Please also add some sort of menu that appears when clicking a link instead of right clicking.
7	A progress bar indicating the export from PMKS to solidworks would be very helpful since I thought my computer was just glitching out the whole time. As well as an ability built into the software to manually input data points rather than having to edit them in a notepad editor. Also perhaps implementing a feature where if you hover your cursor over an icon, it tells you the name and function of that icon.
8	A way to describe link size numerically, and a snap for when holes on the same ternary link line up.
9	Just general bug and crash fixes. I'm not familiar with other programs like this so I don't have a lot of background to compare it to. I think what you have is extremely useful and making it more consistent would be fantastic.
10	I would improve the instructions and help sections as they seemed incomplete and were not useful in helping me figure out how to perform the described tasks. I would also like to see some sort of automatic update to the associated Solidworks model when a linkage is modified in PMKS+.
11	It would be nice to be able to import from solidworks into PMKS. This way you can place parts/motors/systems you already have made/downloaded and build a linkage around those pieces.
12	Improved functions out of the analysis tab for coming up with graphs, maybe add to the help feature to include more help with getting the graph
13	Add functionality to define links with lengths and angles as well as coordinates

14	Make the instructions and features more clear. Make the tutorial more detailed, similar to the slideshow.
15	Get it to stop crashing and look newer.
16	In addition to the dislikes from the previous page, the speed buttons seem to be misleading. I found that the 1x was the fastest, 2x was the slowest, and 3x was in between. I'd expect it to get faster as the number goes up which threw me off at first. I also know that this software is still in production but i'll list a few things I noticed just in case you are unaware (although you probably know of them already). I noticed that as you hover over the speed and direction buttons at the bottom of the screen, they say clockwise and 1x no matter which setting you press. The About button under the help section didn't work for me. The export data buttons under the analysis tab didn't do anything for me either.
17	Better tutorials and better help button. less confusing
18	N/A
19	Ctrl + Z. Crash Handling. Better Zoom
20	An undo button would be class! Also the capability to delete joints and links after you have created them. A less intuitive feature could be the ability to shift the connection points of links after you have generated them.
21	fix the bug make it stable , without it crashing after I open it longer than 10 minutes
22	Labels on the grid for the points of the linkages.
23	Add more thorough information to the tutorial so users have a better idea of how to use all the features, not just making a basic linkage.
24	It will be great if the Exporting can have a progress bar, showing how many percentage is down. Then I can know whether my connection is stable to export all parts and assembly.
25	Mainly, you need to focus on fixing the bugs I encountered while using the program. While the program seemed easy to use, the bugs made it difficult to design new linkages. For example, I had to begin the linkage with joint D, because I could not create a link DCE from the joint C. As a result, I had to close the program and restart the entire linkage. Being able to delete and edit the components of the linkage is a necessary function which I was not able to do.
26	Make the program more robust and it will be very beneficial to mechanical engineering students
27	Make there be an undo function or a delete link function that can easily be accessed.
28	Easier to export to solidworks

29	A way to change linkage length in the PMKS software would be a very welcome addition.
30	Just work on the UI, make it less buggy and more concistence. I dont want to spend 5 min just trying to click on something.
31	Better way of seeing which joint is which. In program dimension modification.
32	Already provided suggestions in previous sections! Thank you for letting me test it and I wish you the best of luck moving forward!
33	Please add a section explaining how to workaround some of the common problems found when exporting to solidworks
34	Find a way to edit the coordinates within the program instead of NotePad, and put the graphs of tracking the kinematics of links or joints in the analysis menu.
35	Improve the functionality of the table. Fix zooming in with a mouse scroll wheel. Add an option in the link menu to make that link the input link.