Report on Milestone 4: Drill Program for The Cornell University Big Red Marching Band Show Committee

David Kelly - dmk257 - Percussionist, Class of ‘14
Christian Compton - cbc74 - Trumpet, Class of ‘14
Philip Corriveau - pdc52
Reid Cohen - rac336
Yize Li - yl2376
Omari Powell - ojp3

Table of Contents

General Overview

Purpose and Scope

The purpose of this project was to redesign and redevelop the current program that the Big Red Marching Band uses create its drill (a description of the formations the marching band creates during halftime and pregame at football games). The current program, known as Rank Panda, was designed in house by members of the band and unfortunately leaves much to be desired. Our project, named Rank Panda 2.0, involved redeveloping this program from scratch in order to fix bugs, making it easier to install on different operating systems, and adding other requested features that did not exist in the original product. Existing features which needed to be reworked included the ability to create and save drill, playback drill as it will be performed on the field, and export drill to .pdf format in order to be distributed to the band. The new features which we initially hoped to implement included collision detection amongst the bands ranks (lines of players on the field), the incorporation of sound files with playback, and exporting the drill to an MP4.

Objectives and Goals

The main objective of this project was to develop a new drill program for the Big Red Marching Band that is free of the bugs and problems that plague the current version of Rank Panda. This allows the marching band to write drill much more quickly, easily, and with less frustration. Secondary goals of this project included adding newly requested features on top of the existing features.

Primary success for this project will be judged on several points:

- Does Rank Panda 2.0 cover all of the features that the previous Rank Panda implemented?
- Does Rank Panda 2.0 implement all features in a manner that is easier to use than the previous Rank Panda?
- Is Rank Panda 2.0 bug free, allowing for an uninterrupted user experience?
- Does Rank Panda 2.0 run across platforms, specifically Windows and Mac OS X?

These areas are all equally important for judging the primary success of Rank Panda 2.0. Secondary success for this project, after completing all of the points above to the clients' desires, include implementing all secondary features in a manner that is functional, efficient, and easy for the user to use.

Requirements

Rank Panda 2.0 aimed to fulfill the following requirements:

Functional Requirements

- Create and save projects for shows that can be edited and changed at a later date
- Load projects that have already been created
- Graphically display the ranks and provide drill playback functionality
- Allow for music to be played back with the drill
- Export drill to video files
- Export drill to PDF
- Warn the user of ranks collision during the drill

Usability Requirements

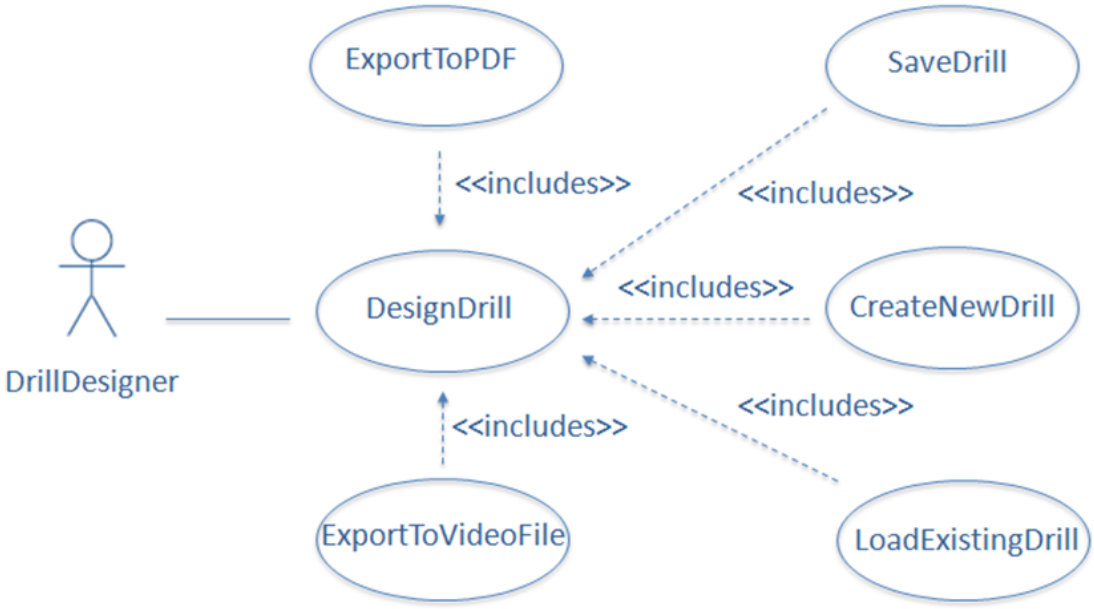
- Cross Platform - Rank Panda 2.0 must be able to work across multiple platforms, most importantly Windows and Mac OS X
- Easy installation - the previous Rank Panda had a rather difficult installation procedure and the clients request this one be much simpler
- Easy to learn UI - the Show Committee chairs in the Big Red Marching Band change every year, so new users must be able to quickly learn how to use the program
- Similar UI to the previous Rank Panda - The client has requested that the interface be similar to the existing rank program's interface

Non-functional Requirements

- Extensibility - the Webmaster in the Big Red Bands often works on software projects in the band. The code must be easily extensible so that changes can be made at a later date if desired

Requirements Models

Use Case Diagram



Use Case Specification

Name of Use Case: DesignDrill

Actor: DrillDesigner

Flow of events:
DrillDesigner opens Rank Panda 2.0

1. Window prompts user with options to create a new file, load an existing file and cancel.
2. DrillDesigner selects New.
3. CreateNewDrill displays a wizard that allows DrillDesigner to input drill information including tempo and counts per measure throughout the song.
4. DrillDesigner continues to make modifications to drill until they would like to save their work.
5. DrillDesigner goes to File menu and selects Save.
6. SaveDrill displays a file chooser to select the location
7. After selecting the save location, DrillDesigner selects Save and SaveDrill saves the XML file.
8. DrillDesigner closes program.

Alternate Paths:

The user may choose to load existing drill rather than creating new drill, changing steps 2 and 3.

2. DrillDesigner selects Load.
3. LoadExisting Drill reads from an XML file which describes the drill and loads the drill into program for editting.

The user may choose to export to .pdf before closing the program, changing step 8.

8. DrillDesigner goes to File and selects Export to PDF
9. ExportToPDF creates a .pdf file summarizing the drill.
10. DrillDesigner closes program.

The user may choose to export to a video file before closing the program, changing step 8.

8. DrillDesigner goes to File and selects Export to Video
9. ExportToVideo creates a video file summarizing the drill.
10. DrillDesigner closes program.

Entry Conditions:

1. RankPanda 2.0 must be installed on DrillDesigner's computer.
2. Storage requirements: must have available space to save an XML file.

Edit Existing Drill Scenario

Purpose: Scenario that describes the use of Rank Panda 2.0 by a Show Committee member who needs to edit existing drill.

Individual: Corey, a member of the Show Committee who is the Drill Designer.

Equipment: Any computer with Rank Panda 2.0 installed.

Scenario:

1. Corey opens Rank Panda 2.0.
2. He selects "Load" from the list of options.
3. Window opens that displays information from existing routine, including moves with ranks that have already been created.
4. Corey makes edits to the Drill by using the command bar at the bottom of the main view
5. After all changes have been made, Corey navigates to the File menu and selects "Save" or "Save As" to overwrite the previous XML save file or save to a new XML file that contains these changes.
6. Corey has completed his desired changes for this session and closes out of program.

Create New Drill Scenario

Purpose: Scenario that describes the use of Rank Panda 2.0 by a Show Committee member

who needs to create new drill.

Individual: Corey, a member of the Show Committee who is the Drill Designer.

Equipment: Any computer with Rank Panda 2.0 installed.

Scenario:

1. Corey opens Rank Panda 2.0.
2. He selects “New” from the list of options.
3. A wizard opens which prompts Corey to enter information about the song including tempo changes and counts per measure changes.
4. When done, Corey selects okay and the main Rank Panda 2.0 screen opens allowing Corey to create the drill.
5. Corey makes edits to the Drill by using the command bar at the bottom of the main view
6. After all changes have been made, Corey navigates to the File menu and selects “Save” or “Save As” to save a new XML file that contains these changes.
7. Corey is prompted for a location to save the file by a file chooser.
8. Corey has completed his desired changes for this session and closes out of program.

Export Drill to PDF Scenario

Purpose: Scenario that describes the use of Rank Panda 2.0 by a Show Committee member who needs to generate a PDF of the drill showing

Individual: Corey, a member of the Show Committee who is the Drill Designer.

Equipment: Any computer with Rank Panda 2.0 installed.

Scenario:

1. Corey opens Rank Panda 2.0.
2. He opens the project which he wants to be exported to PDF.
3. Corey selects the “Export to PDF” button from the File Menu.
4. The system prompts Corey to save the project if any changes have been made that are not save.
5. Corey is prompted for a location to save the file by a file chooser.
6. A PDF file is generated, and opened outside of the program for Corey to look over before printing and distributing to the band members. The PDF displays the rank locations at the beginning of each move, and the respective steps which the ranks must take to get there in time.
7. If satisfied, Corey exits by closing out of the program, or proceeds to edit the existing project.

Export Drill to Video Scenario

Purpose: Scenario that describes the use of Rank Panda 2.0 by a Show Committee member who needs to generate a PDF of the drill showing

Individual: Corey, a member of the Show Committee who is the Drill Designer.

Equipment: Any computer with Rank Panda 2.0 installed.

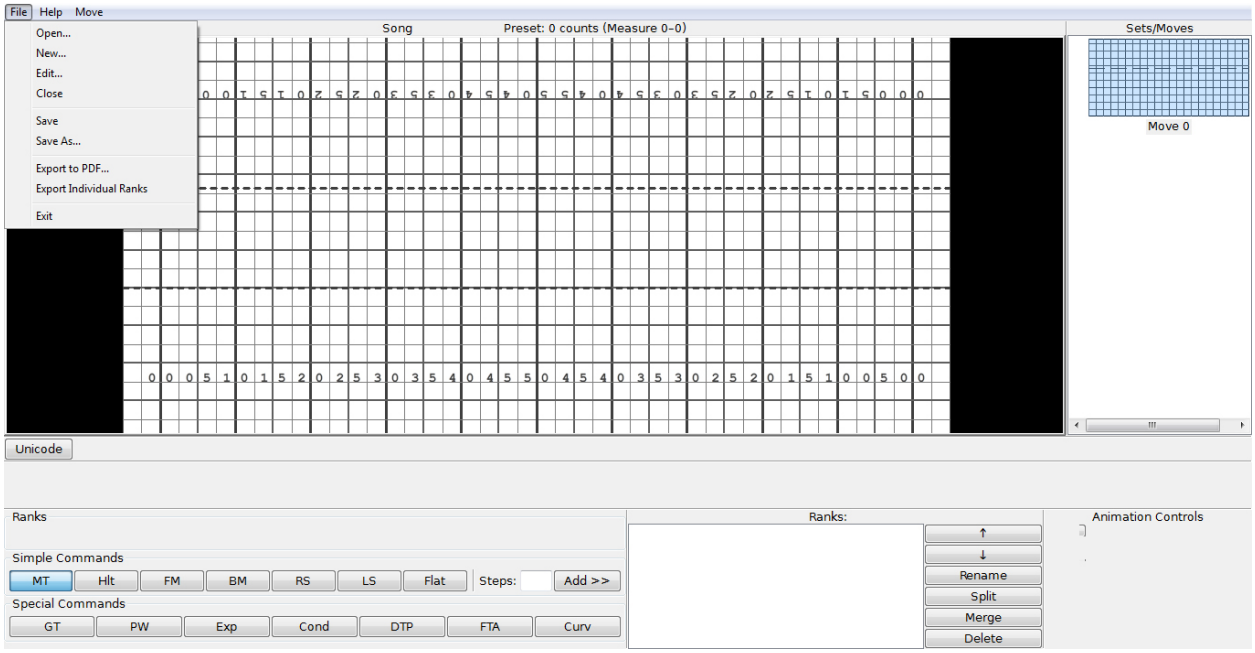
Scenario:

1. Corey opens Rank Panda 2.0.
2. He opens the project which he wants to be exported to Video.
3. Corey selects the “Export to PDF” button from the File Menu.
4. The system prompts Corey to save the project if any changes have been made that are not currently saved.
5. Corey is prompted for a location to save the file by a file chooser.
6. A video file summarizing the drill is generated, and opened outside of our program for Corey to look over.
7. If satisfied, Corey exits by closing out of the program, or proceeds to edit the existing project.

Preliminary Design

User Interface

Shown below is the existing interface for the program Rank Panda 1.0, currently used by the Big Red Marching Band to write its drill for shows. Rank Panda 2.0’s interface is based on this existing interface per requests made by our client. The other key goals in developing the user interface were increased emphasis on simplicity and familiarity. It is important not to have the screen appear overwhelmingly complex and also critical to make use of familiar functions that users know from the previous system. The client has encouraged us to modify the old user interface if we determine ways in which it can be presented more simply.

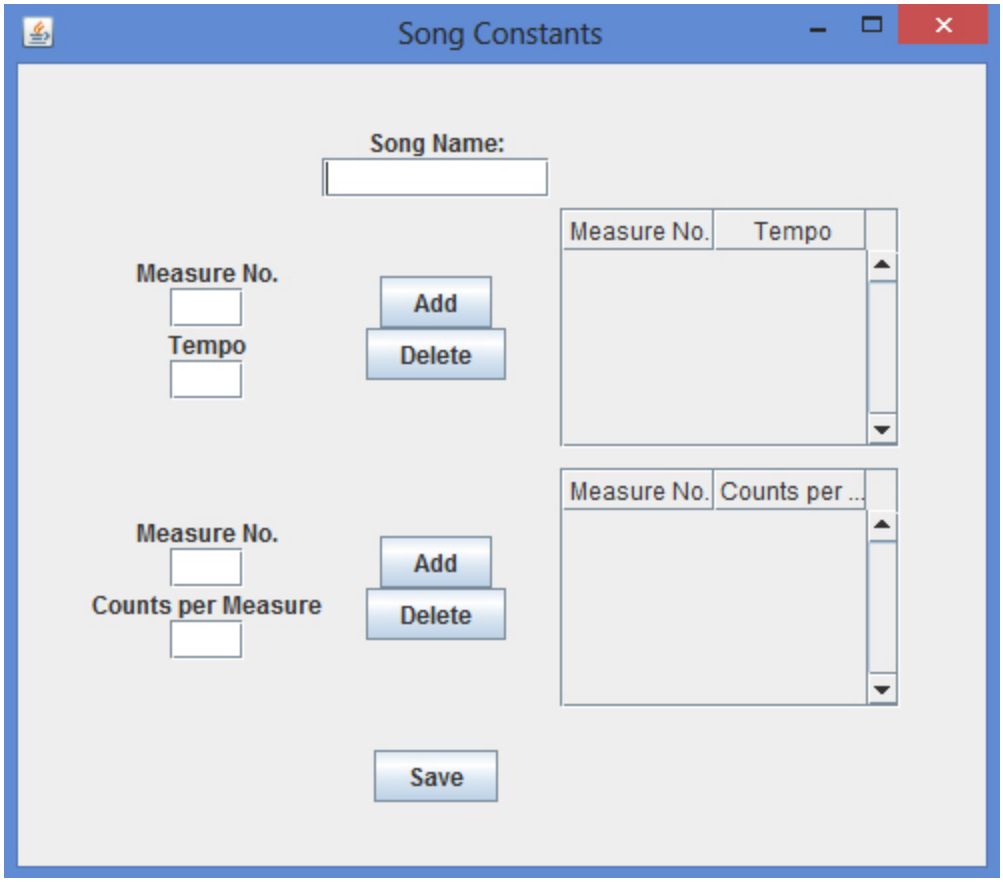


Introduction Screen

Shown below is the window which appears once the program is initially launched. It gives the user three options: to create a new project, load an existing project, or cancel by exiting the program. If the “New” button is selected, the Song Constants wizard (as shown in the next section) takes information for the new project appears, and prompts the user for data pertaining to the song and drill. The goal for this part of the UI was to keep it simple so the user does not have to go searching through menus in order to load a previous project or start with a new one.

Song Constants Screen

Shown below is the screen for the Song Constants wizard. This will allow the user to enter information about the song, like the measure number in which when the tempo or counts per measure change. This information is stored and can be used to provide a realistic playback experience for each drill with matching speeds to the song. This window is launched immediately after the user selects “new” in the Introduction Screen or when the user selects the “Edit Song Constants” option in the Edit drop down menu. If song constants have already been added, the screen will present the user with the previously stored information, ready to be added to, modified, or deleted.

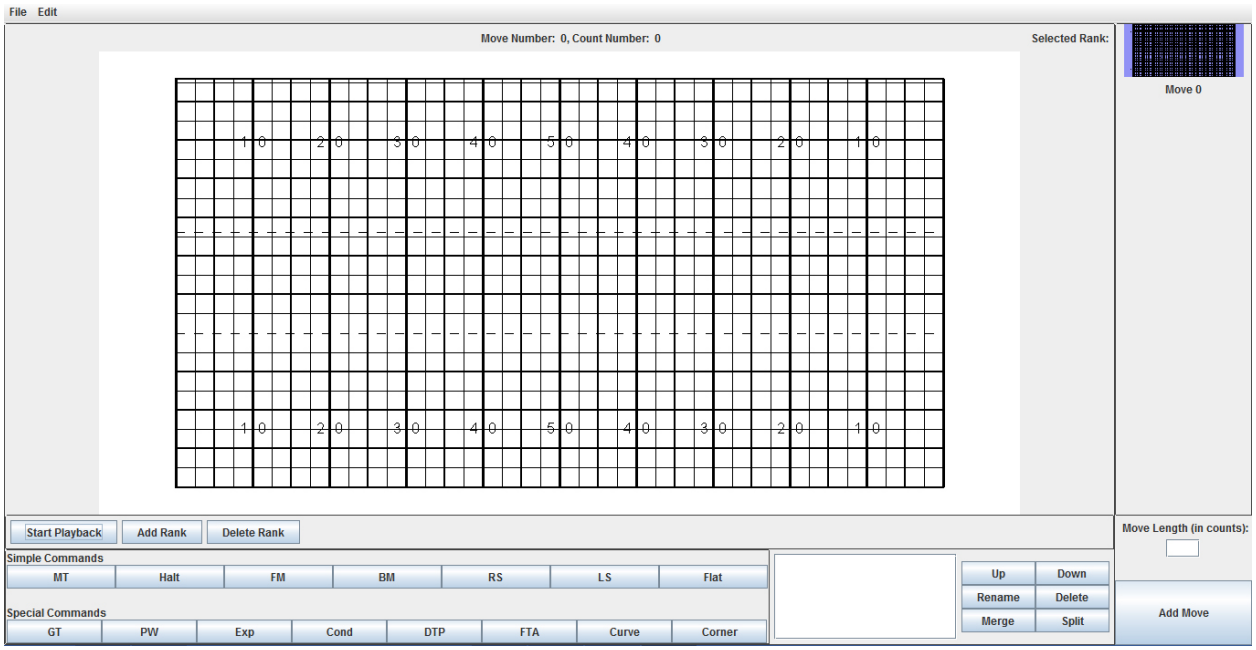


Project Screen

Shown below is the RankPanda 2.0 user interface main view, which can be divided into three main sections. The football field, the largest component, is the graphic area where the ranks will be drawn and displayed to the user with a background of a football field. The move bar, located on the right, contains a list of moves in the project that the user has created and can be used to to switch between editing different moves. The move bar also has an button to add a new move at the bottom of the screen. This button will be used often, and from a usability standpoint it provides a more positive user experience to have it clearly visible and accessible.

The third section of the main view is the Command Bar. Located below the football field, it contains a list of commands that the user can assign to ranks. These are all common abbreviations of marching band commands that will be familiar to the user. On the right of the command bar is a panel which displays the commands for a selected rank. This buttons to the right of this can be used to edit the ranks commands as needed.

In addition to these three main components in the view, the menu above allows the user to perform other critical project functions such as saving, loading, and exporting to PDF.



System Design

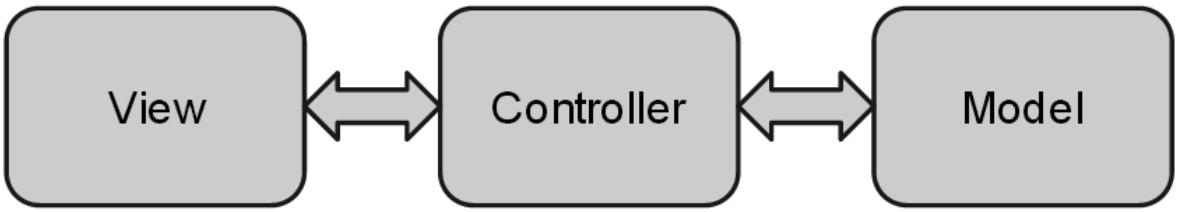
A standalone program satisfies all of the needs and requirements of the clients as there is no need for a web based product, a database, or anything more complicated. This also follows along with the previous iteration of Rank Panda which was also a standalone program. Java was chosen as the main programming language for the project for multiple reasons:

- It fixes the issue difficult installation procedure as well as the cross platform compatibility that were lacking in the previous version of Rank Panda. All that is necessary for installation is the Java Virtual Machine (JVM) which generally does not have a difficult installation procedure on Windows or Mac OS X, our target platforms. It can be somewhat difficult installation on some versions of Linux, but this is not directly a target platform. Beyond the JVM, all the user will need is an executable .jar file to run.
- There are many useful libraries for parsing and generating XML, generating .pdf files, playing music files, and creating video files. This gives us a lot of options to compare between to get the best libraries for our purposes.
- Execution speed is not a huge concern, so the benefit we would get from languages like C/C++ would not be the most useful. For example, if a song plays at 240 beats per minute (rather fast) we would only render 4 frames per second maximum during drill

playback. Java can easily achieve these speeds.

- The very structured nature of Java as an object oriented language makes creating and following a clean system architecture easier.

For our system architecture, we decided to go with a Model-View-Controller (MVC) architecture with some influences from the Model-View-Presenter (MVP) architecture, as depicted below.



The main difference between our architecture and standard MVC is that there is not a connection between the view and the model. This is similar to the MVP design, and provides an extra layer of abstraction between the views, controllers, and models as the controller acts as a middleman between the views and the models. The extra abstraction allows for multiple developers to easily work on separate parts of the code without affecting each other and helps ensure the integrity of the information in the models as the view must use the controllers' methods to change the information in the models. The reason this architecture does not fall into the MVP is because there is not one presenter for each individual view, as the controller(s) can handle interacting with multiple views.

For saving and loading projects, XML was chosen as the formatting of the file. This is a rather standard format, and there are a large variety of XML parsing and generating libraries available to use in Java. We decided to use the org.w3c.dom libraries to implement both of these functions, as they come built in with the Java API and are generally very easy to use. We created our own XML Schema for saving the files, which is detailed in the next section.

Java Swing was chosen for the UI as it is generally one of the simplest UI libraries in Java, handles almost everything we need it too, and provides a consistent feel across different platforms. The one area that Swing does not handle well is the drawing of the football field and the ranks. We are relying on AWT to handle this functionality because of its historical advantage in performance, as well as its affordances in terms of use of graphic widgets, while Swing handles everything else.

The last two libraries left to be specified are the libraries used to export the drill to .pdf files and video files. For exporting drill to .pdf, we are using the Apache PDFBox. We were considering the iText library, but it has a very strict license if a copy of the library is not purchased and PDFBox was able to handle everything we needed. For exporting the drill to a video file, we were looking into the Monte Media Library and the Java Media Framework. Monte Media Library is quite large - containing many more features than we will ever use - and is also still in experimental stages of development according to the developer. Java Media Framework

seems to handle everything we need and will likely be used for music file playback as well, but has not gotten the best reviews. Experimentation was slated to begin with Java Media Framework for creating video files after it has been used to include music with the drill playback, but unfortunately we never reached that stage of development. If Java Media Framework did not appear to be the right choice for creating video files, than we were intending tol look into the Monte Media Library.

XML Schema for Saving Projects

The XML Schema we designed is meant to almost directly correspond with the models of the project. This way, it is very simple to convert the drill created during use of the program into an XML save file as well as read it back into the models. Below is an example of a save file following our XML schema which contains all of the possible tags for a save file. The highest level of the hierarchy is the drill tag. The children of the drill tag contain information about tempo changes, counts per measure changes, as well as the moves - partitions of the song that describe rank movements. The tempo and counts per measure changes contain two fields: the measure number the change happens at as well as the value it is changing too. The move tags contain more information as they are what define the movements of the ranks on the field. Nested within the move tag are the number of the move and the information for each rank. Each rank contains its name, start position, and commands it must follow. Within the start position are the three points of each rank and the x and y positions of each point. The three points are necessary to describe each rank's position so that the tags can easily define a rank's shape if it is straight line, curve, or corner. Within the command tags are tags for command-type, the number of counts for each command, and the command's index in order to identify the order of the commands.

Example:

```
<drill>
	<tempo-change>
		<measure-number>1</measure-number>
		<tempo>120</tempo>
	</tempo-change>
	<tempo-change>
		<measure-number>35</measure-number>
		<tempo>160</tempo>
	</tempo-change>
	<counts-per-measure-change>
		<measure-number>1</measure-number>
		<counts-per-measure>4</counts-per-measure>
	</counts-per-measure-change>
	<counts-per-measure-change>
		<measure-number>20</measure-number>
		<counts-per-measure>2</counts-per-measure>
	</counts-per-measure-change>
	<move>
		<number>0</number>
```

```
        <move-counts>0</move-counts>
        <rank>
                <rank-name>A</rank-name>
                <start-position>
                        <front-point>
                                <x>45</x>
                                <y>10</y>
                        </front-point>
                        <point-one>
                                <x>50</x>
                                <y>10</y>
                        </point-one>
                        <end-point>
                                <x>55</x>
                                <y>10</y>
                        </end-point>
                        <line-type>0</line-type>
                </start-position>
                <end-position>
                        <front-point>
                                <x>45</x>
                                <y>10</y>
                        </front-point>
                        <point-one>
                                <x>50</x>
                                <y>10</y>
                        </point-one>
                        <end-point>
                                <x>55</x>
                                <y>10</y>
                        </end-point>
                        <line-type>0</line-type>
                </end-position>
        </rank>
        <comments>This is the zero move</comments>
</move>
<move>
        <number>1</number>
        <move-counts>100</move-counts>
        <rank>
                <rank-name>A</rank-name>
                <start-position>
                        <front-point>
                                <x>45</x>
                                <y>10</y>
                        </front-point>
                        <point-one>
                                <x>50</x>
                                <y>10</y>
                        </point-one>
```

```
                        <end-point>
                            <x>55</x>
                            <y>10</y>
                        </end-point>
                        <line-type>0</line-type>
                    </start-position>
                    <command>
                        <index>0</index>
                        <command-type>0</command-type>
                        <counts>12</counts>
                        <command-name></command-name>
                    </command>
                    <command>
                        <index>1</index>
                        <command-type>2</command-type>
                        <counts>8</counts>
                        <command-name></command-name>
                    </command>
                    <command>
                        <index>2</index>
                        <command-type>1</command-type>
                        <counts>4</counts>
                        <command-name>High step</command-name>
                    </command>
                    <end-position>
                        <front-point>
                            <x>45</x>
                            <y>15</y>
                        </front-point>
                        <point-one>
                            <x>50</x>
                            <y>15</y>
                        </point-one>
                        <end-point>
                            <x>55</x>
                            <y>15</y>
                        </end-point>
                        <line-type>0</line-type>
                    </end-position>
                </rank>
                <comments>This is the first move</comments>
        </move>
</drill>
```

Final Progress

Summary of Semester

Final Sprint

Due to further delays in the previous sprint, this last sprint was largely spent finishing the primary goals, as it seemed highly infeasible that we would finish the secondary goals in any reliable manner. The original dates we had planned for this sprint are listed below:

Preliminary Shipment - November 25th

- Storing the tempo changes and measure lengths for later use (finish on UI side)
- A complete set of commands for ranks
- The ability to export drill to .pdf files
- Finished playback
- Incorporate changes from second presentation
- Refine product to a shipping state
- Changes based on client feedback
- Updated UI fully the above

Third Presentation - December 3

- Rank collision detection
- Export drill to video
- Beginning development of sound playback
- Updated previous features based on feedback from Preliminary Shipment

Product Shipment - December 16

- Finished sound playback
- Refinement of product
    - UI refinements
    - Code refinements for scalability and extensibility
- Package product and source code for easy client access
- Deliver product to client

While our goals for this sprint did include plans to work on the secondary goals, these were mainly included on the off chance that we encountered no problems while implementing the rest of the primary goals. This allowed us more room to work on the primary goals if we determined that time would be better spent polishing those rather than working starting on the secondary goals.

As with previous sprints, again we encountered delays. Quickly, we made the call with the clients that it would not be feasible to attempt the secondary goals and spent the entirety of this sprint working on and polishing the primary goals to exactly fit the client's desires. During user testing with the client, we took note of many changes that the clients wanted for the UI, and spent large portions of the sprint working on those on top of the primary goals.

Of the primary goals, we spent time finishing the storing of the song constants (tempo changes and counts per measure changes), implementing more commands, finishing exporting drill to .pdf files, implementing playback and refining the UI. We made large progress in each of these areas and accomplished a lot with these tasks. Of the primary goals, the two which still need work are the commands and the playback. With the commands, FTA, DTP, and Corner still need to be implemented. As much of the framework is already in place, this would not be too difficult, it unfortunately just required time we did not have. Currently, playback will play the drill for the user to watch, but only does so at one tempo and starting at move 0. Adding multiple tempos and starting in different areas is not very difficult and for the most part only requires a measure to counts conversion function. This, too, simply took time we did not have.

During this sprint, the users also provided us many little features and changes to the UI they would like during user testing. Many of these were little improvements to the old Rank Panda. The changes included things such as “ghost ranks” to indicate where a command would end, selecting multiple ranks, keyboard shortcuts and many others. A lot of time was spent during this sprint implementing these features and getting the UI exactly to where the client wanted it. While we did make a large amount of progress in this area, we unfortunately did not have the time to implement all of the features they desired, as described below in the description of work left to complete.

Overall, this sprint was one of the most productive sprints, if not the most productive, for the team. We implemented many unfinished features that had fallen behind in previous delays as well as spent much time implementing changes the clients had requested during user testing. Unfortunately, due to the delays in previous sprints we were not able to entirely finish everything covered under the primary goals.

Semester Overview

After the feedback that we got on the feasibility report for this project, we realized that while the primary goals were achievable in the scope of the semester, the secondary goals which would provide many enhanced features would take significantly more effort. Thus, we created two schedules: the first was the optimistic plan, assuming we were able to get all of the primary goals done in time, leaving enough time for the secondary goals to be implemented fully; the second was the fall back plan which detailed how to finish only the primary goals in order for the product to be minimally successful. This fall back plan was the schedule we ultimately ended up using due to a variety of delays as well as extra features requested by the clients for the UI during user testing.

In retrospect, while we were not able to complete these enhanced secondary features, the product as it currently stands is an improvement over the legacy system in many ways. The fact alone that this product can be easily installed and reliably run on a variety of systems, including Windows, Mac, and Linux systems which can run Java, greatly expands its use. We have implemented several features, such as playback and move comments, which either did not exist or did not work in the previous Rank Panda; improved efficiency of drawing ranks on the field (particularly curved ranks) as well as the overall performance of the program; and a UI that is similar to the previous Rank Panda but includes many changes requested by the clients from

it. As bugs were a large problem in Rank Panda, we have done our best to track down and stamp out bugs in Rank Panda 2.0. We created a spreadsheet used by both the developers and clients to track these bugs and resolve them while they happened. While it is inevitable that other bugs will pop up over time, we have set up plans for further support of the program (as detailed in the Pans for Further Support section) and have provided the clients with our contact information should they ever need us for anything.

While we have had a fair amount of successes with the program, we have also had an equal share of struggles. One of the biggest issues that we ran into was the time constraint. Unfortunately we lost a considerable amount of time during the first and second sprint. Our last sprint, as previously mentioned was by far the most productive, and in retrospect it would have been better if we could have emulated that same productivity in the first two legs. Because of these delays, we were unable to tackle the secondary goals and left several small unfinished sections in the primary goals. In order to make further support and implementation of these areas straight forward, we have done our best to lay out the framework for them and provide clean, extensible code which can be easily modified later. Particularly, we would have liked to start development earlier on the commands and spend more time on them, as that was an area that was important to the clients and product. By completing the framework for the commands, implementing them will not take much time and been relatively straight forward.

Overall, while we have not completed the clients' primary requirements by the end of this semester, we have completed a large portion of them and left only several small tasks for the product to be considered finished, having already laid out the framework for future developers. We have done our best to provide clean extensible code for them to change later as well as providing concise documentation. On the UI side, we have incorporated the clients' initial feedback during the requirements phase as well as a large amount of the new features and changes requested during user testing.

Completed Requirements

Of the requirements mentioned at the beginning of this report, below are the requirements we have completed:

Functional Requirements

- Create and save projects for shows that can be edited and changed at a later date
- Load projects that have already been created
- Graphically display the ranks and provide drill playback functionality
- Export drill to PDF

Usability Requirements

- Cross Platform - Rank Panda 2.0 must be able to work across multiple platforms, most importantly Windows and Mac OS X
- Easy installation - the previous Rank Panda had a rather difficult installation procedure and the clients request this one be much simpler

- Easy to learn UI - the Show Committee chairs in the Big Red Marching Band change every year, so new users must be able to quickly learn how to use the program
- Similar UI to the previous Rank Panda - The client has requested that the interface be similar to the existing rank program's interface

Non-functional Requirements

- Extensibility - the Webmaster in the Big Red Bands often works on software projects in the band. The code must be easily extensible so that changes can be made at a later date if desired

Incomplete Requirements

Of the requirements mentioned at the beginning of this report, below are the requirements we have not completed.

Functional Requirements

- Allow for music to be played back with the drill
- Export drill to video files
- Warn the user of ranks collision during the drill

Description of Work Left to Compete

As shown above, there are a few requirements that were not completed for this project. For the most part, we have completed nearly all of the primary goals, but there are a few features which still must be implemented. These include the FTA, DTP, and Corner commands, a measure number to counts conversion, and playback incorporating tempo. For the remaining commands, most of the framework is in place for them already. All three commands will need code to draw them and describe their movements, but FTA and DTP also require a method for storing the waypoints that the rank would move between during playback. This would not be too difficult, but would simply require more time. A measure number to count conversion (and vice versa) would allow users to enter measure numbers for the ends of moves rather than counts and is the only thing hindering playback incorporating tempo. This conversion requires a list of counts per measure changes throughout the song which is already inputted into the program when the user creates drill for the first time, but the function using this data was simply not implemented yet. This, again, would not be very difficult but would simply require more time.

During the User Testing that we conducted with our client, they presented some additional features that they were looking to have implemented, features which were not initially presented in the feasibility report. Some of these included a "ghost rank" functionality, where the program will display a marker for where the rank will have to end up; keyboard shortcuts, especially when adding ranks; and the ability to select multiple ranks and edit them simultaneously; and a "no-op" command that would take the place of very complicated moves. In watching the users interact with the current system, some other shortcuts that would be very beneficial included being able to click the delete key to delete the selected property, and pressing

the “enter” key to submit the selected property. While we were unable to complete all of the additional features the clients requested for the UI, we developed the best possible you we could, satisfying as many of their additional requirements as possible.

Unfortunately, we were unable to complete any of the secondary goals that we had hoped to do. From the start, we knew these would be very difficult to accomplish on top of creating a brand new Rank Panda 2.0 from the ground up. These were designed to be extra features which the clients found useful to be developed if the primary goals of the project went exceedingly well. This was not the case, and we were unfortunately not able to begin on these goals.

Plans for Further Support

In the past, there has not been any official support for software created for or by the band as dictated by the marching band constitution. The closest the band had to this was the webmaster who managed the content of the marching band website. Because of this, much of the support for technical projects in the band has fallen to the members of the band who created them or alumni after those members graduated. This has often meant that many projects have gone unsupported over the years and users would have to either accept problems they encountered or attempt to fix them themselves.

There has been a trend by the current webmaster (David Kelly, dmk257, a member of this group) to push for more support for any technical projects the band has needed, such as creating a new backend for the website and recovering lost data. While there still is nothing set in the constitution to officially put the webmaster in charge of supporting the other technical projects of the Big Red Marching Band, the current webmaster has agreed to continue supporting this project past its delivery to the client and the next webmaster has been made aware of the project and agreed to support it if possible given their school work.

Our team approached this project bearing in mind that there could be many ways which the band could further develop this system beyond the scope of our project. As result, we purposefully made the code easily extensible, readable, and provided sufficient documentation so that future developers who wish to work on this project may do so with ease.

If the clients or users have any questions about the product beyond what is described in the user manual and this document, the contact information about each member of this group has been added to both of these documents so that the clients or users can reach out to them as needed. Two of the members of this group are also members of the marching band - David Kelly, dmk257, a percussionist in the Class of 2014, and Christian Compton, cbc74, a trumpet in the Class of 2014 - who the clients or the users can reach out to if they would like a contact who was a member of the band. This has also been indicated at the top of each document.