Project Number: MQP-MOW-0902

BASEBALL VISUALIZATION

A Major Qualifying Project Report

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Jennifer Hunt

Date: May 3, 2010

Approved by:
Professor Matthew O. Ward, Advisor

Abstract

The goal of this major qualifying project is to create an adaptable and entertaining visual representation of baseball pitch data collected from MLB.com. Several different techniques are used to create an application that allows the selection of different pitches, displays the visualization of the selected pitch trajectories, and creates several different graphs of the pitch data. The insights gained from user testing are utilized to make suggestions for future enhancements to the project.

Table of Contents

Table of Figures

1. INTRODUCTION

MLB.com provides a plethora of Major League Baseball data, stored in XML files, recorded from baseball games played since 2007. The data is available for the public to download and analyze, and contains game conditions, player stats and specs, as well as the data necessary to reconstruct the baseball pitch trajectory. The specific recorded data for each game includes the home team, the away team, player names and heights, and the baseball's initial position, velocity, and acceleration.

Baseball fans, athletes, and coaches are interested in compiling and analyzing pitch statistics in order to discern trends and potentially increase player performance. Current methods of visualizing the MLB.com baseball data include the PitchFX Tool [1] provided by BrooksBaseball.net and MLB.com Gameday [8]. The PitchFX tool provides a graphical analysis of the pitch data, but does not provide spatial representations of pitches that would allow pitch comparison in 3D space. Gameday provides a 3D visualization of live pitch data, but does not permit the user to select specific pitches or pitches from multiple games. A baseball visualization application is needed that combines statistical pitch analysis and the 3D representation of the pitch and pitch path with the ability to select the pitch information from a broad selection of different pitch sets.

The purpose of this Major Qualifying Project (MQP) was to design, implement, and evaluate a system that creates 3D visual representation of user specified baseball pitch data. The desired features of the baseball visualization application include user specified search criteria, pitch and pitch path animation, pitch size and coloring options, as well as a strike zone with height modification that depends on the current batter. The project also needs an area to display large amounts of statistical data that can be analyzed similar to the data provided with the PitchFX Tool [1].

In order to achieve the project goal, a small portion of the MLB.com pitch data needs to be downloaded. A baseball visualization database needs to be created, and populated with the MLB.com pitch data. A set of queries need to be created to retrieve the lists of pitchers, batters, and dates from the database. A second set of queries needs to be created to retrieve specific pitches from the database based on the user specified search criteria. The calculations required to describe the pitch trajectories were defined. A baseball field was created that displays the calculated pitch trajectories to the screen. The user interface of the application was created to allow the user access to the system’s functionalities. The interface was divided into three primary sections: the pitch selection panel, the visualization panel, and the graph panel.

In order to evaluate the baseball visualization application, user testing was conducted. The user testing scenario involved three distinct phases. The first phase consisted of the administration of a short questionnaire in order to categorize the subject’s baseball background and explain trends and outliers in the results. The second phase involved familiarizing the test subject with the application, administering a sequence of tasks for the subject to complete, and observing the subject’s behavior when using the application. The last phase of the user testing scenario, involved the administration of a follow up questionnaire that requested comments on the system, and suggestions for area of improvement. These comments were used to suggest future modifications to the application.

The following sections provide details of the background research, design, implementation, results, and conclusions of the Baseball Visualization Major Qualifying Project. The background research, Section 2, contains a description of the current baseball visualization applications, visualization techniques, and visualization tools. Section 3 explains the desired features, options for implementation, advantages and disadvantages, and the final design decisions. The implementation of the baseball visualization application is discussed in Section 4. The results of the project and the user testing are discussed in Section 5, and Section 6 discusses suggestions for future work and the project conclusions.

2. BACKGROUND RESEARCH

Players, coaches, fans, sports journalists, and umpires are interested in compiling statistics about baseball pitches [4]. Motion capture is used to capture the flight of each baseball pitch [1]. This flight data, as well as other related information, is collected and stored on MLB.com for the public to use. The MLB.com pitch data can be visualized in order to detect trends in player performance. The following section presents background information on the project, including current baseball visualizations, visualization techniques, and visualization tools.

2.1 Current Baseball Visualizations

There are various visualizations related to Major League Baseball games, including non-obtrusive additions of visual elements in broadcasted games and graphical representations of pitch data. KZone, the PitchFX tool, and Gameday are all examples of baseball visualizations.

2.1.1 KZone

In 2001, Sportvision created the KZone System [4]. The ESPN KZone system monitors the trajectory of each pitch. Figure 1 shows an example of the KZone system during a game.

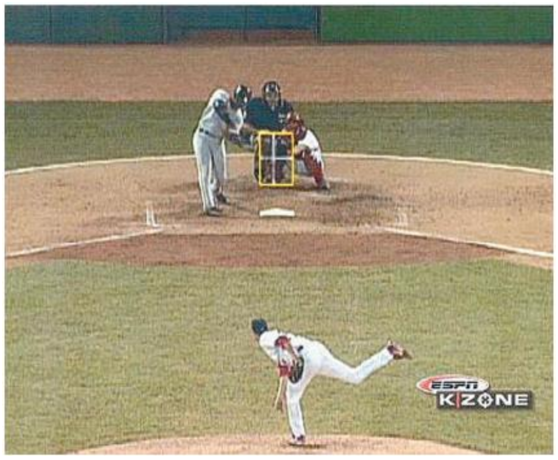


Figure 1: K Zone during a televised game [4]

The system uses computer generated graphics to enhance the viewer's experience by outlining the strike zone boundaries. The system implements three subsystems: the camera pan-tilt-zoom encoding subsystem, the measurement subsystem, and the graphic overlay subsystem. The camera pan-tilt-zoom encoding subsystem calibrates the cameras that are used for broadcasting the game [4]. The measurement subsystem is used to determine whether each baseball is a ball or a strike, by measuring the batter's stance and detecting the trajectory of each baseball [4]. The graphic overlay subsystem uses the camera calibration data, produced by the camera pan-tilt-zoom encoding subsystem, and the baseball measurements, collected by the measurement subsystem, in order to produce the televised graphics [4].

2.1.2 PitchFX Tool

The PitchFX Tool [1], displayed on BrooksBaseball.net, provides a statistical analysis of any pitcher during any game. The user enters a date, game, and pitcher, and the tool displays a variety of information regarding that particular combination. Figure 2 shows a strike zone plot that was generated by the PitchFX Tool.

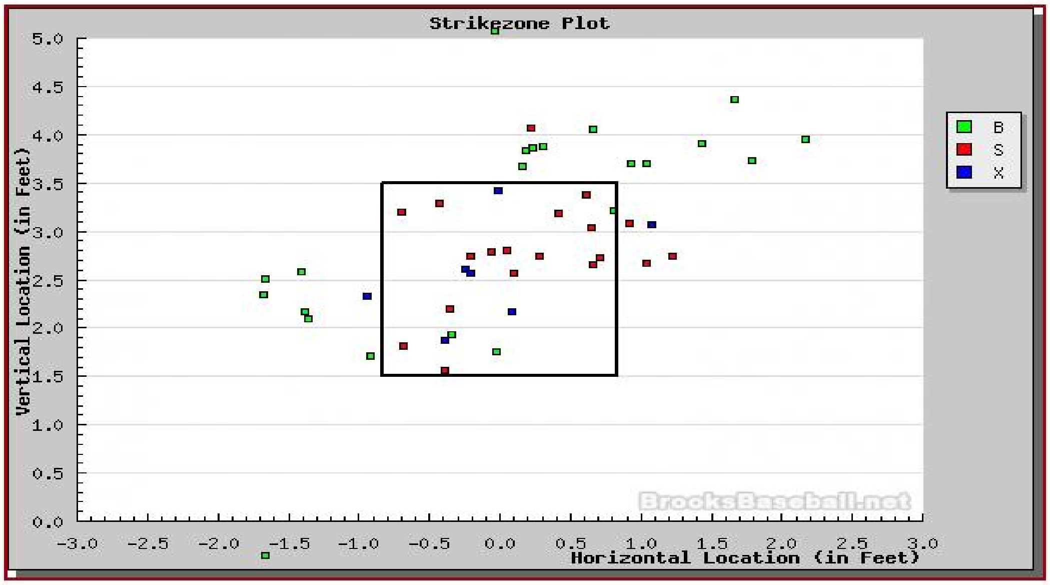


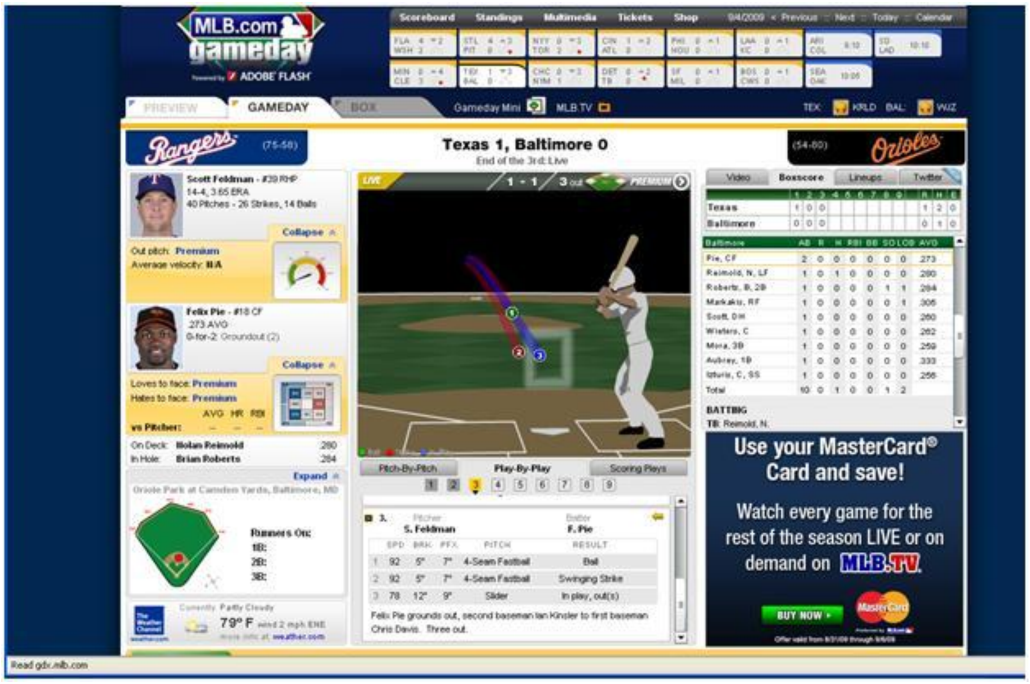
Figure 2: PitchFX tool strike zone plot [1]

In Figure 2, color indicates the umpire's call; for example, green represents balls (B), red represents strikes (S), and blue represents outs (X).

The statistical data provided by the tool includes pitch totals, pitch speed, release point, break, spin direction, spin magnitude, and horizontal movement [1]. The PitchFX tool is useful for producing a two-dimensional, graphical representation of the MLB.com pitch data that is used to analyze trends and increase player performance. The BrooksBaseball.net PitchFX tool does not allow pitch selection from specific types of pitches or different games. The tool does not have the capability to produce a three-dimensional spatial representation of the data.

2.1.3 MLB.com Gameday

Gameday [8] is a live visualization of baseball pitch trajectories and game information provided on MLB.com. Gameday has three options for display. There is a full 3D mode, a light 3D mode, and a miniature Gameday. Figure 3 provides an example of Gameday's full 3D mode. In full 3D mode the trajectory of the pitched baseball is drawn to the screen.



Texas 1, Baltimore 0
End of the 3rd: Live

Scott Feldman - #39 RHP
14-4, 3.65 ERA
40 Pitches - 26 Strikes, 14 Balls

Out pitch: Premium
Average velocity: N/A

Felix Pie - #18 CF
.273 AVG
0-for-2: Groundout (2)

Loves to face: Premium
Hates to face: Premium

	AVG	HR	RBI
vs Pitcher:	–	–	–

On Deck:	Nolan Reimold	.280
In Hole:	Brian Roberts	.284

Oriole Park at Camden Yards, Baltimore, MD

Runners On:
1B:
2B:
3B:

Currently: Partly Cloudy
79° F wind 2 mph ENE
more info at weather.com

	1	2	3	4	5	6	7	8	9	R	H	E
Texas	1	0	0							1	2	0
Baltimore	0	0	0							0	1	0

Baltimore	AB	R	H	RBI	BB	SO	LOB	AVG
Pie, CF	2	0	0	0	0	0	0	.273
Reimold, N, LF	1	0	1	0	0	0	0	.280
Roberts, B, 2B	1	0	0	0	0	1	1	.284
Markakis, RF	1	0	0	0	0	0	1	.306
Scott, DH	1	0	0	0	0	0	0	.260
Wieters, C	1	0	0	0	0	0	0	.262
Mora, 3B	1	0	0	0	0	0	0	.259
Aubrey, 1B	1	0	0	0	0	0	0	.333
Izturis, C, SS	1	0	0	0	0	0	0	.256
Total	10	0	1	0	0	1	2	

BATTING
TB: Reimold, N.

3. Pitcher S. Feldman Batter F. Pie

	SPD	BRK	PFX	PITCH	RESULT
1	92	5"	7"	4-Seam Fastball	Ball
2	92	5"	7"	4-Seam Fastball	Swinging Strike
3	78	12"	9"	Slider	In play, out(s)

Felix Pie grounds out, second baseman Ian Kinsler to first baseman Chris Davis. Three out.

Figure 3: MLB.com Gameday example: full 3D mode [8]

The Gameday visualization contains a realistic model of the infield of a Major League Baseball field, including the bases, pitcher's mound, and the fouls lines, as well as a batter model. The batter does not swing at pitches but the model switches its position according to whether the current batter is right or left handed.

In Gameday, strikes are red and balls are green. In the full 3D version, the paths are also color coded and appear to fade with time. When multiple trajectories are displayed simultaneously, the paths converge and it is difficult to discern different paths. The strike zone is displayed as a two-dimensional, hollow rectangle drawn around the strike zone area. Gameday also implements a scoreboard, pitcher and batter information, current game status, and strike out animations. Gameday defines the strike zone as a two-dimensional rectangle when the strike zone is a three-dimensional area. The MLB.com Gameday application only supports the visualization of live pitch data from a specific baseball game. Gameday does not support the visualization of user specified pitch data that can be used to analyze trends within specific data sets.

There are aspects of the PitchFX tool and MLB.com's Gameday that are used to create this baseball visualization project. These aspects include the 3D spatial representation of the baseball pitch and the pitch trajectory, taken from Gameday, as well as the user specified pitch selection and pitching graphs, taken from the PitchFX tool.

2.2 Visualization Techniques

Visualization includes images, graphs, diagrams, or animations that communicate some message. There are many different types of visualization, including scientific, and information visualization. There are a variety of techniques that can be used to enhance the visualization of pitch data, including flow visualization, collision detection, shading, and particle systems.

2.2.1 Flow Visualization

One possible method that can be used to visualize the baseball trajectory is flow visualization. Standard visualization mappings in flow visualization are arrows, stream lines (Figure 4), streak lines, path lines or particle traces (Figure 5), time lines, and contours (Figure 6) [11].

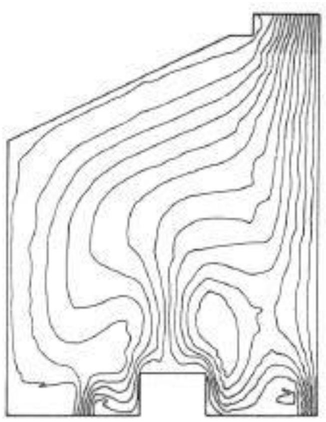


Figure 4: Stream lines in 2D [11]

Stream lines are generated as 2D or 3D curves derived from experimental data and characterize the general flow pattern. The distance between each line is inversely proportional to velocity [11]. Path lines, or particle traces, can be visualized as static curves that are interpreted in a single instant, or as a time interval [11].

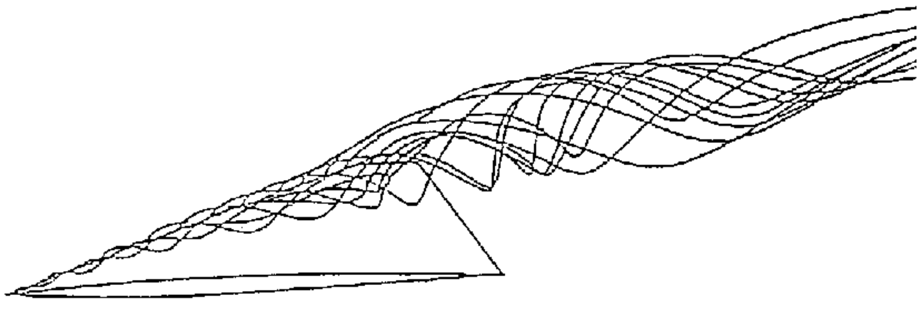


Figure 5: Particle traces [11]

Contours, also called iso-curves (in 2D) and iso-surfaces (in 3D), are used to visualize scalar data with a full range of visual depth.

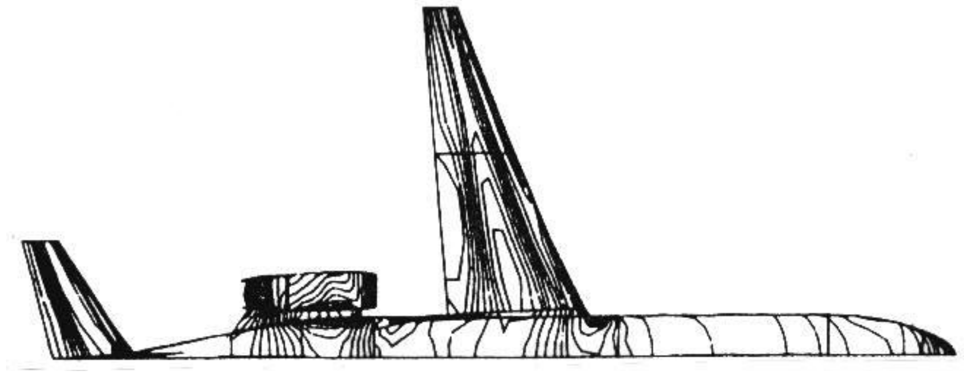


Figure 6: Pressure contours on a surface [11]

The process of computer visualization of data begins with data generation, then data enrichment and enhancement, visualization mapping, rendering, and finally display [11].

2.2.2 Collision Detection

Collision detection verifies the intersection of parts of one object with another object. Collision detection between an object and an implicit object is achieved by evaluating the implicit function at each sample point on the object [10]. Collision detection can be used to verify the baseball's intersection with the strike zone.

2.2.3 Shading

There are various shading models that use illumination equations in order to calculate the values used to render images. Figure 7 shows an example of a sphere without shading.

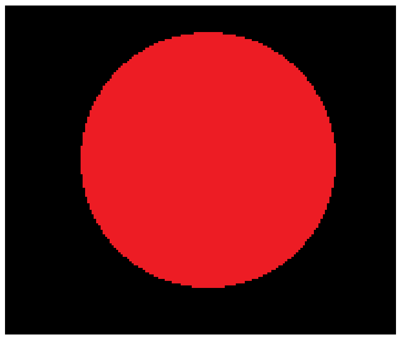


Figure 7: Uniform color example

One shading model is called flat shading. In flat shading the illumination calculation is done for each planar patch. The patch is rendered as one single color value. The object appears facetted because the patches are clearly visible [7], as shown in Figure 8.

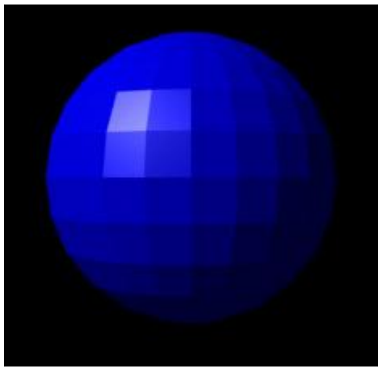


Figure 8: Flat shading example

A second shading technique is Gouraud shading, shown in Figure 9. In Gouraud shading, an interpolation method is used for smoother shading. The illumination value is calculated at each of the polygonal patch's vertices. The colors within the patch are determined by linearly interpolating the color values at the vertices [7].

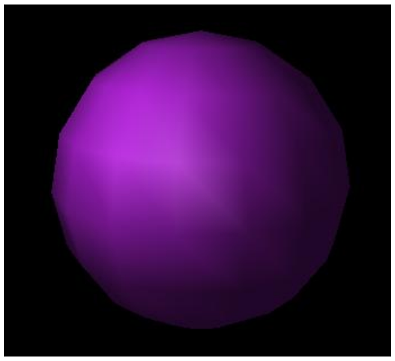


Figure 9: Gouraud shading example

Phong shading, Figure 10, improves upon Gouraud shading by interpolating the normal across the polygonal patch as opposed to the intensity, in order to pick up highlights. Phong shading is more computationally intense then Gouraud shading [7].

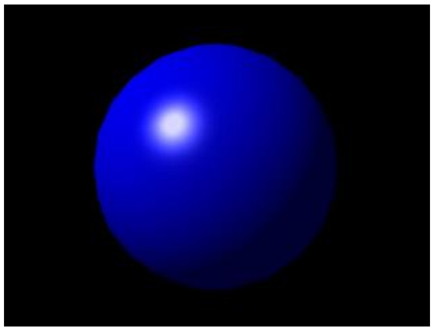


Figure 10: Phong shading example

2.2.4 Particle Systems

A particle system [10] is a technique used to simulate phenomena that are difficult to render, such as fire or smoke. Particle systems are easier to render because of the following assumptions. Particles do not collide with other particles; particles do not cast shadows, only as an aggregate body, and only on the environment; and particles do not reflect light [10]. Particles are also rendered as a point light source to simplify the rendering process. Here is an example of a particle structure, which is represented as a tuple holding the velocity, position, force, and mass, as defined in [10].

```
typedef particle_struct struct {
    vector3D        p; //position
    vector3D        v; //velocity
    vector3D        f; //force
    float           mass;
} particle;
```

The particle system’s structure is defined in [10] as

```
typedef particleSystem_struct struct {
    particle        *p; //array of state information for each particle
    int             n; //number of particles
    float           t; //current time (age) of the particle system
} particleSystem;
```

During the computation of each frame, new particles are generated and assigned attributes, particles that have reached the end of their lifetime are terminated, and all remaining particles are animated and rendered. Particles are randomly generated based on a user-specified distribution, which revolves around the predefined average number of particles desired at any one time [10]. Particle attributes include lifetime, position, velocity, color, transparency, and shade parameters. These attributes are used to determine the particle's life, motion, and appearance in the system [10]. Figure 11 shows an example of a particle system that was used to generate the effect of fire.

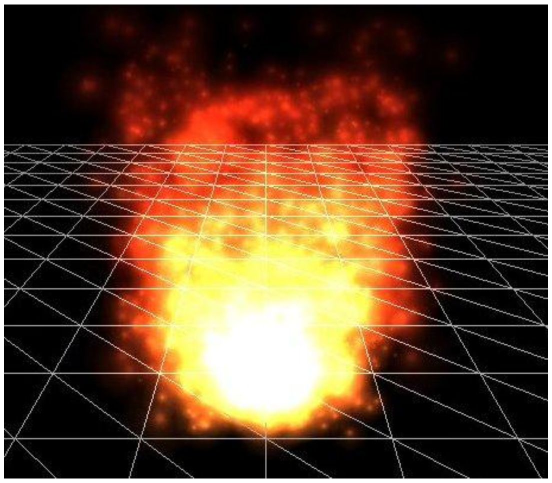


Figure 11: Particle system example [10]

A particle system could be used in the visualization to implement the generated path of each baseball.

2.3 Visualization Tools

There are a number of visualization tools that can be used to expedite and enhance the visualization development. These tools include the Prefuse Visualization Toolkit [12] and The Visualization Toolkit [13], as well as Open DX [9], JUNG [6], and The Flare Visualization Toolkit [3].

2.3.1 Prefuse Visualization Toolkit

The Prefuse Visualization Toolkit is a Java-based set of software visualization tools that uses the Java2D graphics library. Prefuse can be used to build applications and web applets, and integrates features for visualization, interaction, and data modeling. Prefuse is very useful for visualizing tables, graphs, and trees [12]. The Prefuse Toolkit offers many different tools such as data structures, integrated color maps, animation processing, and event logging. Figure 12 displays an example of a visualization created with Prefuse.

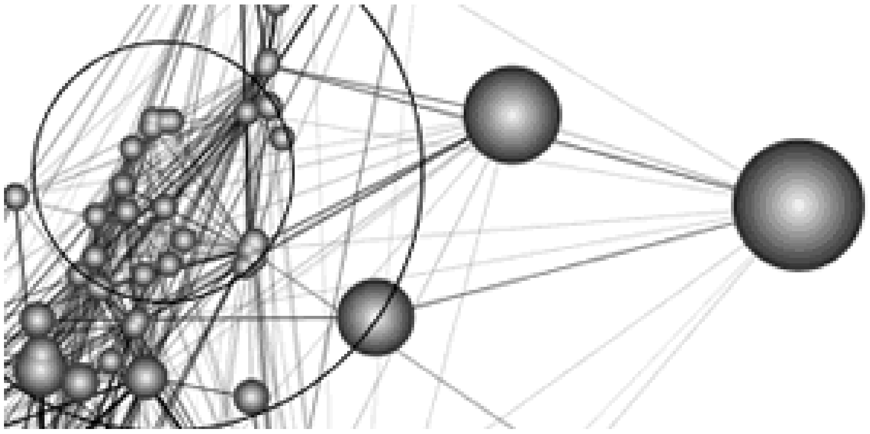


Figure 12: Prefuse example [12]

Other aspects of Prefuse that are helpful in visualizing data are that Prefuse contains a large library for force-based physics simulations, and user controls such as navigation and drag controls [5]. The Prefuse Visualization Toolkit is free to download and use, and there is a user's manual that describes in detail how to use the many different features of Prefuse.

2.3.2 The Visualization Toolkit

The Visualization Toolkit (VTK) is an open source software system implemented as a C++ toolkit, with support for Java, Python, and Tcl wrapping. VTK is used for 3D computer graphics, image processing, scientific visualization, modeling, volume rendering, and information visualization [13]. Figure 13 shows an image that was created with The Visualization Toolkit.

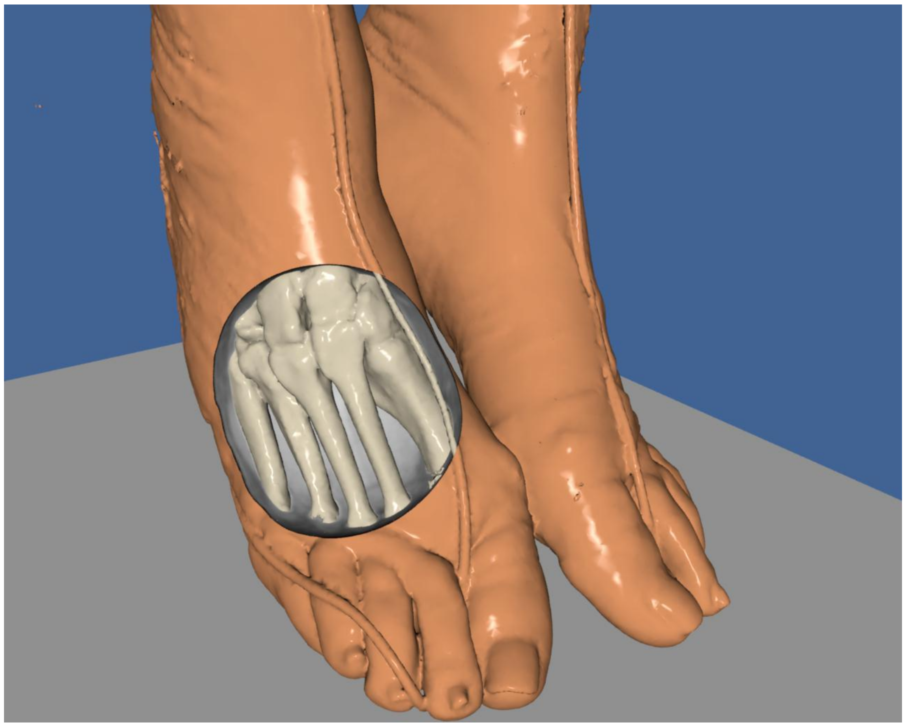


Figure 13: The visualization toolkit example [13]

2.3.3 Other Visualization Tools

OpenDX [9] is an open-source visualization tool that allows the visualization of primarily scientific, engineering, and analytical data [9]. Figure 14 displays two examples of visualizations produced with the Open DX software tool.

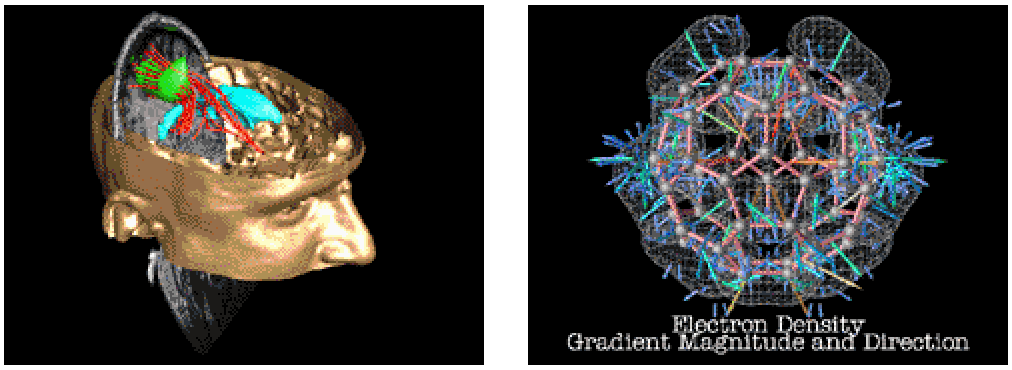


Figure 14: OpenDX examples [9]

The Java Universal Network/Graph Framework (JUNG) [6] is an open-source software library that supports the modeling, analysis, and visualization of directed and undirected graphs, multi-model graphs, hypergraphs, and graphs with parallel edges [6]. Figure 15 provides an example of a visualization created using JUNG.

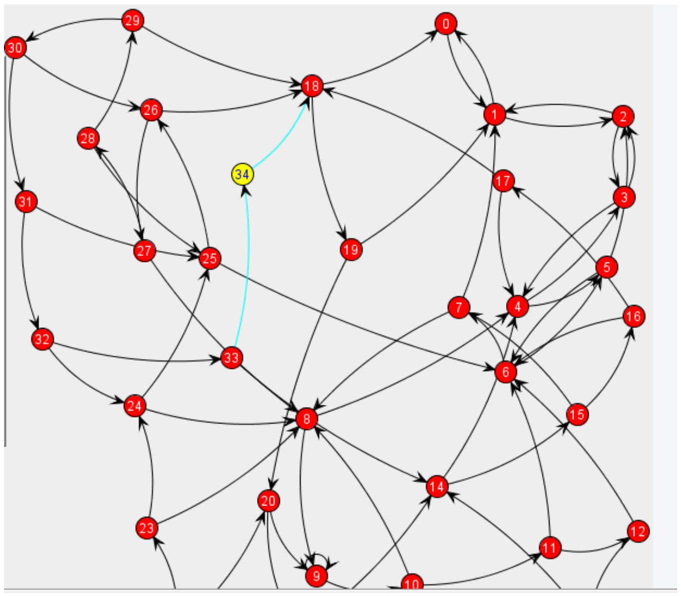


Figure 15: JUNG example [6]

The Flare Toolkit [3] is an ActionScript library, adapted from Prefuse, that can be used to create a wide variety of visualizations ranging from basic charts to complex interactive graphics [3]. Figure 16 shows two applications of Flare, a still of a smoke animation, and a simple tree graph. The toolkit supports data management, visual encoding, animation, and interaction techniques [3].

Figure 16: The Flare Visualization Toolkit examples [3]

There are many visualization techniques that can be used to represent data. There are also many software visualization tools that can be used to enhance the visualization. The background research, techniques, and tools were all considered throughout the design and implementation process of the baseball visualization project.

3. DESIGN

Upon determination of the project goal, namely designing, implementing, and evaluating a three-dimensional representation of user specified pitch data, the desirable features of the project were defined. The features include the methods for data storage, data selection, interface design, visualization design, the baseball animation, the baseball path, and the strike zone. This section describes the implementation options, advantages and disadvantages, and the final design decisions with respect to each desirable feature.

3.1 Data Storage

The method for storing the available data was defined in order to create the baseball visualization application. A desirable feature of the system is the ability to store, access and manipulate the data effectively. The implementation options for data storage include simple text files stored within a local file system, or a database. The storage of data in text files on a local file system does not provide the data organization necessary to query large subsections of the information. Databases are created for storing, organizing, and writing queries on large amounts of data. The most appropriate data storage system for the baseball application is a database.

3.2 User Selection Methods and the Interface

The primary feature of the baseball visualization application is the ability for the user to select any pitch based on specified data sets of pitches. There are a number of filters that can be applied to the data in order to produce different result sets. The data could be filtered by the pitcher, the batter, and the date the game was played. The type of pitch, such as fast ball, change up, or slider, the pitch call, strike, ball, or out, and pitch specific data, such as the speed and break, could also be filtered. Filtering the data in this way, allows the user to query a wide range of pitch subsets from the pitch database.

In order to implement the pitch selection filters, the user interface needs to support a simple pitch selection panel. The panel could be populated with a combination of GUI elements to enable pitch selection. For example, the pitcher, batter, and date selections could be implemented with combination boxes. Range sliders could be used to filter the pitch specific data, such as the pitch’s start and end speeds. To retrieve the specified subset of pitches, queries could be generated, specific to the selected search parameters. The potential subsets of pitch data supported by the user interface includes one or more pitches from one or more pitchers and pitches of a specific type or call from one or more pitchers. The interface also needs to support queries for all of the pitches that were pitched to a specific batter, the pitches a specific pitcher pitched to a batter, as well as any combination of the afore mentioned selections.

Filtering the data allows the user to select any supported combination of pitches to visualize and analyze. The creation of the pitch selection panel, as well as the selection filtering methods, is discussed in the implementation section.

A desired feature of the baseball application is that all pieces of the application are combined and displayed with one main window. The interface window must reserve an area for pitch selection, an area for the 3D pitch visualization, and an area for the 2D pitch graphs generated by the application. In order to implement the required features, the main window needs to contain three panels, the pitch selection panel, the visualization panel, and the graph panel. The separation of the interface into multiple panels is easier to use. The implementation of the three different panels is described in the implementation section.

3.3 Visualization Features

The 3D visualization is contained in the visualization panel of the baseball application. The pitch visualization requires a model of the infield of a Major League Baseball field, including the pitcher's mound and home plate. The visualization also requires the generation of pitches, and display of pitch animations on demand. Implementation options include creating the required models with modeling software and importing them into the Java3D visualization class, or creating the objects with Java 3D. Model implementation with Java3D is easier, and sufficient for the first implementation of the baseball visualization application. Model creation using the Java3D library will be pursued in the implementation phase of the project.

3.3.1 View Features

Predefined views are a desired feature of the project in order to analyze the baseball during and after the trajectory animation. There are five desired views (top, catcher, pitcher, side, and close-up) that allow a full range of baseball examination. The top view is an aerial view of the entire animation, the catcher view is the view facing the pitcher (the catcher's point of view), and the pitcher view is the view facing home plate (the pitcher's point of view). The side view is the view between third base and home plate, and can be used to see the rise and fall of the pitch trajectory. The close-up provides a zoomed view of the strike zone. Possible view implementation methods include defining the camera position at five locations and switching between the views, according to the user, or arbitrary viewing.

3.4 Pitch Animation and Features

A required feature of the baseball animation visualization is the animation of the pitch from the pitcher's mound to home plate. The baseball animation could be implemented similar to

MLB.com's Gameday. The animation of the pitch includes the generation of baseball objects, the generation of the pitch trajectory, and the speed and sizing options of the pitched baseballs.

3.4.1 Pitch Path

The path of the pitch provides a constant spatial representation of the pitch trajectory. Drawing the pitch's trajectory, in addition to the pitch animation, is a desired feature of the visualization. There are a number of different options available for implementing the pitch trajectory. The trajectory could be implemented as a particle system, in which the particles could follow the pitch's path, and exist for a specified amount of time. The trajectory could also be implemented as a conglomeration of objects used as place holders, and positioned at specified frame intervals along each pitch's path.

Drawing the baseball trajectory allows the spatial comparison of two or more pitches. One observable disadvantage of drawing the baseball's trajectory is path convergence and screen clutter, resulting from drawing multiple pitches. There are a few different methods for mitigating screen clutter and trajectory convergence. The first method considered involves screen clearance after a predetermined number of pitches are drawn. The method prevents screen clutter, but interferes with the user's requests. The second method considered was the fading and eventually disappearance of each path based on a defined path lifetime. The method does not effectively reduce screen clutter if the user draws a plethora of baseballs within a short timeframe. The final method considered was a user defined screen clearance option, implemented by a simple "clear screen" button that removes all baseball objects and trajectories when clicked.

A desired feature of the baseball and baseball trajectory is the representation of additional information about the pitch using color. Color could be used to create a distinction between balls and strikes, to display different pitch speeds, or to display different pitch types. The path visualization, “clear screen” option, and path coloring were selected to pursue during the implementation phase of the project.

3.4.2 Pitch Size and Speed

There is approximately 60 feet between the pitcher’s mound and home plate. A baseball traveling at 90 mph would take slightly less than half a second to travel from the pitcher’s mound to home plate. It is difficult to track the pitch’s trajectory when the pitch is displayed at full speed. The option to change the animation speed is a desired feature of the baseball visualization application. The baseball speed is set by the equations. One option to implement animation speed alteration is to modify the baseball speed variable in the equation. An interface element needs to be provided for the user to specify the speed change. Implementing an animation speed mechanism allows the user to see the baseball animation at the actual speed of the ball, displaying the intensity of the pitch, as well as a slower speed to analyze the pitch’s movement.

Another requirement of the visualization project is that the baseball size reflects the standard size of a MLB ball, which is approximately 2.9 inches in diameter. To implement this, the size of the sphere object created to represent the baseball, should be set to 0.0029. The units being used are 1/1000 of an inch. Maintaining a consistent default baseball size creates more realism in the application. Major League Baseball standard baseballs are difficult to locate within the field because the baseballs are small. An additional desired feature of the visualization project is to include baseball sizing options. A possible implementation of sizing options could entail creating an interface element that allows the selection of multiple baseball sizes. The baseball objects created could reflect the selected size. Sizing options allow the visualization to maintain realism and allow the pitch to be easily spotted on the screen.

The pitch size and animation speed modifications were both selected to pursue during the implementation phase of the project.

3.5 Strike Zone Features

In baseball, the strike zone defines the area that determines whether a pitch is declared a strike or a ball, and is an integral part of the game. The left, right, front, and back boundaries of the strike zone correspond to the edges of home plate. The upper bound is the midpoint between the batter’s shoulders and belt. The lower bound is defined as the area at the hollow of the batter’s kneecap. The upper and lower boundaries of the strike zone differ to some extent depending on the batter’s proportions.

There are two primary methods for defining the strike zone in this application: static and dynamic. A static strike zone maintains predefined upper and lower boundaries that remain consistent throughout the visualization. A possible method of implementing a static strike zone includes creating a box object with predefined dimensions based on average batter heights that do not change throughout the entire visualization. Although the static representation of the strike zone is easier to implement, the representation sacrifices pitch call accuracy.

A dynamic strike zone, in which the upper and lower boundaries change to correspond to the current batter’s proportions, is a desired feature of the visualization application. The top and bottom boundaries of each pitch are specified within the dataset of collected pitch information. In order to implement a dynamic strike zone, the boundary information is collected and stored. The boundary information is used to manipulate the size of the strike zone object before a specific pitch is drawn. After considering both representations of the strike zone, the dynamic strike zone was selected to pursue during the implementation phase of the project.

One possible use of the application includes confirming the accuracy of the umpire’s call by comparing the umpire’s call with the resulting location of the drawn pitch. Accurate strike zone collision detection can be used to validate the umpire’s call of a specific pitch. Baseball collision with the strike zone is another desired feature of the application. Implementing collision detection between the strike zone and the baseballs allows the user to confirm whether the ball intersected the strike zone. Collision detection can be implemented by creating a bounding object around the strike zone, and checking at different time intervals if any of the baseball objects intersect the bounding object. If an intersection is detected, a responding event can be triggered, such as color change. The advantages of implementing strike zone collision include a higher probability of umpire call verification.

The required and desired features are pursued during the implementation phase of the project, which is described in the following section.

4. IMPLEMENTATION

The required and desired features, outlined in the Design section, were considered during the implementation phase of the project. A number of steps were taken to implement the visualization project in accordance with the necessary features. The programming language and environment were selected. The Major League Baseball data was acquired and stored in the baseball visualization database. A Java3D program was created containing a baseball field, and different views of the drawn pitches. Calculations were performed on the data and the pitch trajectory was drawn to the field. The user interface was created in order to combine pitch selection and the field. The following section explains the implementation of each of the different parts of the baseball visualization Major Qualifying Project.

4.1 Programming Language

In order to implement the baseball visualization project, a programming language and environment were selected. Java (Java 3D), and C++ (Open GL) were considered as implementation languages because of the extensive 3D graphics libraries provided. The entire project was implemented in Java and Java3D using the eclipse programming environment, as a result of programmer familiarity and learning curve reduction.

A variety of visualization tools were considered to enhance and expedite the visualization process. In particular, The Visualization Toolkit, providing 3D capabilities and support for both C++ and Java, was pursued. The toolkit was not used in the baseball visualization project because of system build difficulties during the initial stages of project implementation.

4.2 The Baseball Visualization Database

MLB.com provides a plethora of Major League Baseball data in XML files available to the general public. Joseph Adler [2] provides in his book, *Baseball Hacks*, the Perl script, hack_28_spider.pl, that was used to download all of the game data for April 7th, 2007, April 2nd, 2008, and October 9th, 2008, totaling 3,705 different pitches. The dates were randomly selected from games played in different months and years to allow the verification of pitch selection from different games. The number of games downloaded was limited by space.

A baseball visualization database was created using MySQL. Mike Fast [2] provides a database schema that was used to create the tables necessary for organizing the pitch data. The tables created were "atbats", "games", "pitches", "players", and "umpires". The "atbats" table contained references to information regarding the entire at bat, such as the pitcher, batter, inning, game, the strike count, and the ball count. The "games" table contains the home and away teams, a reference to the umpire, and the game conditions, such as wind speed and direction. "Pitches" contains the position, velocity, and acceleration values of each particular pitch. The "players" and "umpires" tables contain references to the players or umpires names and ids. The database was populated with the downloaded MLB.com data using a Perl script provided by Mike Fast.

4.2.1 Connection

To establish a connection between the Java project and the baseball visualization database, the Java Database Connectivity (JDBC) driver was used. A *DBConnect* class was created to provide a connection to the database using the JDBC driver and the server, username, and password parameters specific to the baseball database.

4.2.2 Pitch Selection Queries

After a connection with the baseball database was established, queries were created to retrieve baseball information. A *dbqueries* class was implemented to contain the queries created to retrieve the pitchers, batters, dates, and pitch trajectory information. The first set of queries was created in order to populate the interface with the information necessary to select specified pitches, such as the list of pitchers, batters, and game dates, as well as the minimum and maximum values of the start speed, end speed, and the break. Each query was written in a separate function; for example *getPitchers* returns an array of strings containing the entire pitcher's name on the following specified query:

"select distinct first, last from (select pitcher as eliasid from atbats) as b natural join players"

The eliasid is the unique id of each pitcher and batter stored in the database. In this query, the first and last name of the pitcher whose id matches the id provided by the at bat is returned. The "players" table and the "atbats" tables are joined so that the specific pitcher is returned, as opposed to any player.

The second set of queries was created to populate the "available pitches" table with the database results from a search specified with the user selected search criteria. Multiple queries were created in which the selection criteria changes depending on the user specified search criteria. For example, if the user does not select a pitcher or batter, the query will not specify that criteria.

A *Pitch* class was created to store the "pitchID", "pitcher", "batter", and "des" of a pitch that was used to populate the "Available Pitches" table. The "des" of a pitch represents the description of the outcome of the pitch; for example "Hit by Pitch".

4.3 Pitch Trajectory Calculations

There are two Microsoft Excel spreadsheets that are provided at *The Physics of Baseball* [4]. The first spreadsheet, *template_for_cd*, calculates the drag coefficient and lift coefficient given the 9-parameter fit. The 9-parameter fit contains the baseball's initial position, velocity, and acceleration in the x, y, and z directions. The second spreadsheet, *full_3d_trajectory*, uses the drag and lift coefficients to calculate the position, velocity, acceleration, and other pitch trajectory information, with respect to time.

In order to store and manipulate pitch information, a *PitchData* class was created, containing instance variables for each of the columns in the “pitches” table. These variables included the pitch id, pitcher, batter, the initial x, y, and z positions, velocities, and accelerations, as well as the start and end speed. For each pitch added to the “Visualize Pitches” table, the *Pitch ID* was used to select the corresponding tuple from the baseball visualization database. A *PitchData* object was created for each pitch, and the variables were set to equal the result values from the query. A list of *PitchData* objects, labeled “pitches”, was maintained, containing the pitch information for every pitch in the “Visualize Pitches” table. When the “draw” button is clicked, the pitch trajectory was calculated for each *PitchData* object in the “pitches” list.

A *Calc* class was created to calculate the trajectory information for each pitch. The *Calc* class contains functions that represent the calculations defined in the *template_for_cd* and *full_3d_trajectory* files. The functions created include *calcAX*, *calcAY,* and *calcAZ,* which are used to calculate the acceleration in the x, y, and z directions, respectively. The *Calc* class defines a multitude of additional functions that produce values for the additional variables involved in the calculation defined in the *full_3d_trajectory* file. The calculated x, y, and z positions of each *PitchData* object at every time delta are stored in integer arrays that are passed as parameters to the animation, in order to be displayed.

4.4 The Baseball Field and Baseball Animations

The baseball field was created to provide a setting for the baseball pitch animations. The field contains the ground, bases, strike zone, baseballs, and baseball trajectories. The baseball field was implemented in the *BVAnimation* class using the Java3D library.

The infield was created using two separate box shape instances named the *in_green* and *in_dirt*. The length and width of the *in_dirt* box was specified to reflect MLB standards. The height of the box was set to a minimal value. The *in_green* box was implemented as a slightly smaller version of the *in_*dirt shape, and positioned in the center of the *in_dirt* box. Two separate appearances, dirt and grass, were created and applied to each of the boxes. The two boxes define the ground of the infield, and were rotated 90 degrees to reflect the infield of a baseball field.

The three bases and home plate were modeled as simple boxes and scaled to represent the actual size of the bases. A simple, white appearance was applied to each of the bases. The base positions were translated to place each base in the proper location in the field. The pitcher's mound was modeled and placed in the world, similar to the bases with the exception that the mound's width is smaller. The mound is not higher than the field, straying from MLB field standards. Figure 17 shows the generated baseball field from the catcher's point of view.

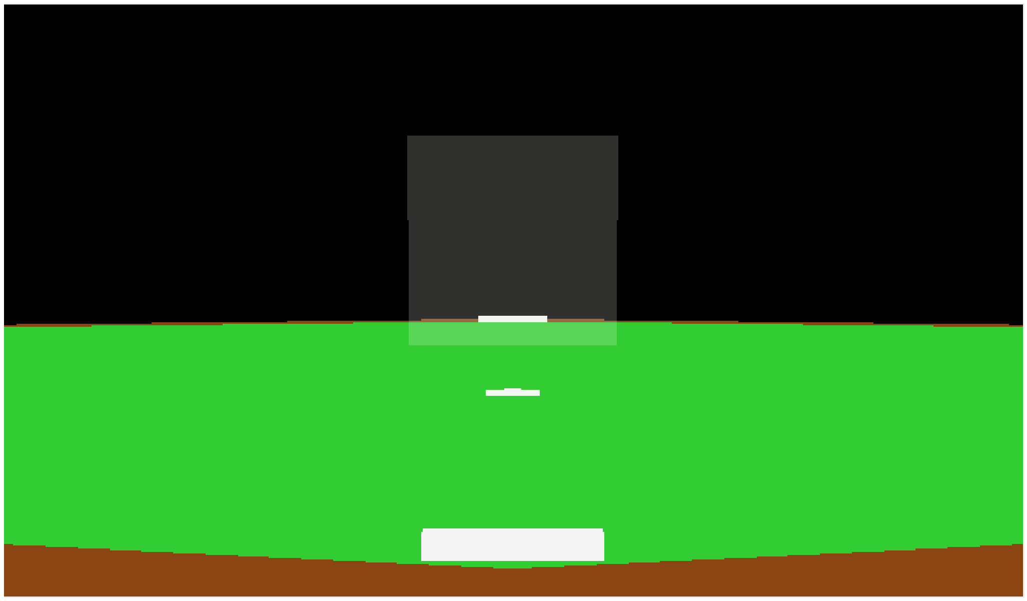


Figure 17: Baseball field as seen from the catcher's view

4.4.1 Views

Five predefined views: top, side, catcher, pitcher, and zoom, of the visualization were implemented. Figure 17 shows the objects seen from the catcher's point of view, which is the default view when the application is first started. Figure 18 shows the remaining four views: top, side, pitcher, and zoom, respectively. In order to implement each view the plane of projection and camera location and orientation was transformed to produce the separate views. The user interface allows the user the ability to instantly switch between any of the defined views.

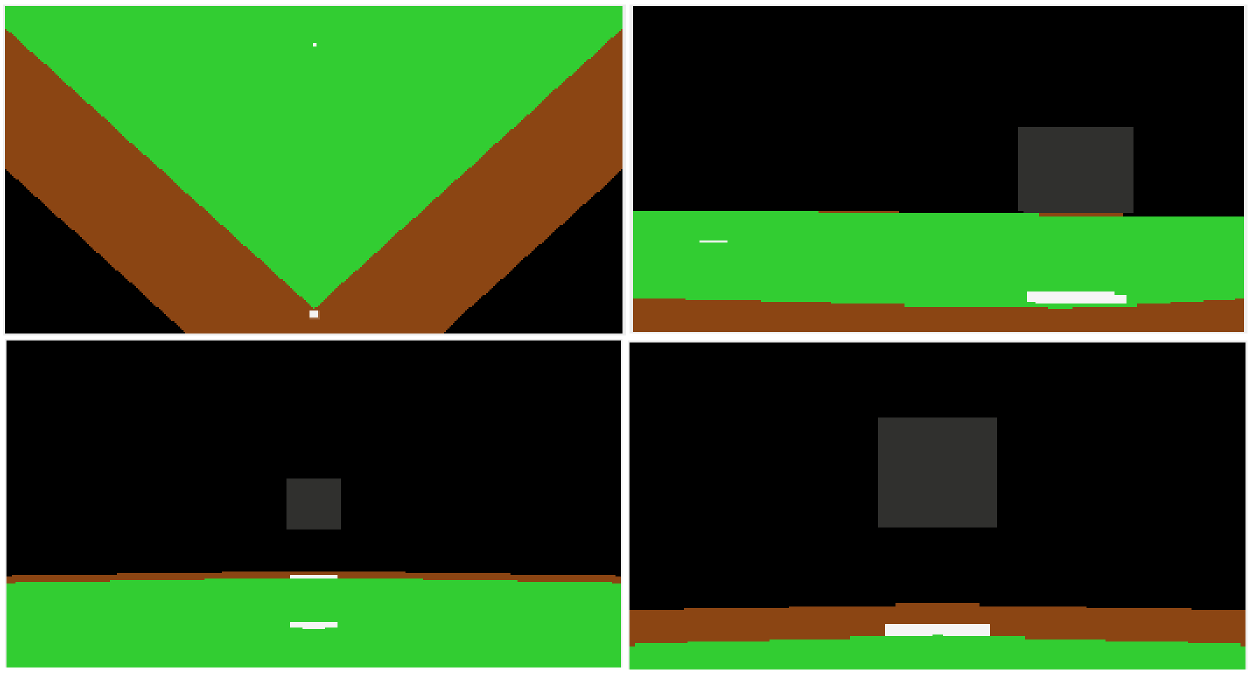


Figure 18: Predefined views (top, side, pitcher, zoom)

4.4.2 Baseball Animations

The function *setTransforms*, defined in *BVAnimation,* sets the pitch data, creates the pitch transformation groups and the baseball objects, and starts the baseball animation. During each frame the baseball object's position is transformed according to the positions calculated in the *Calc* class.

In order to draw the baseball trajectories, the *addPathSphere* function is defined in

BVAnimation. At every tenth frame of the baseball animation a path sphere is generated, located at the current position of each pitch. The generation of the path sphere at every 10th frame was selected because generating the path sphere every frame is expensive, and every tenth frame is less expensive and still draws a congruent path. The path sphere maintains the coloring options of the original pitch, and is slightly smaller. At animation completion the full pitch trajectory is displayed.

The application allows drawing of multiple pitches to the screen at a single time, an example of this is shown in Figure 19. The strike zone in the figure appears modified based on the average upper and lower strike zone boundaries of the pitches drawn.

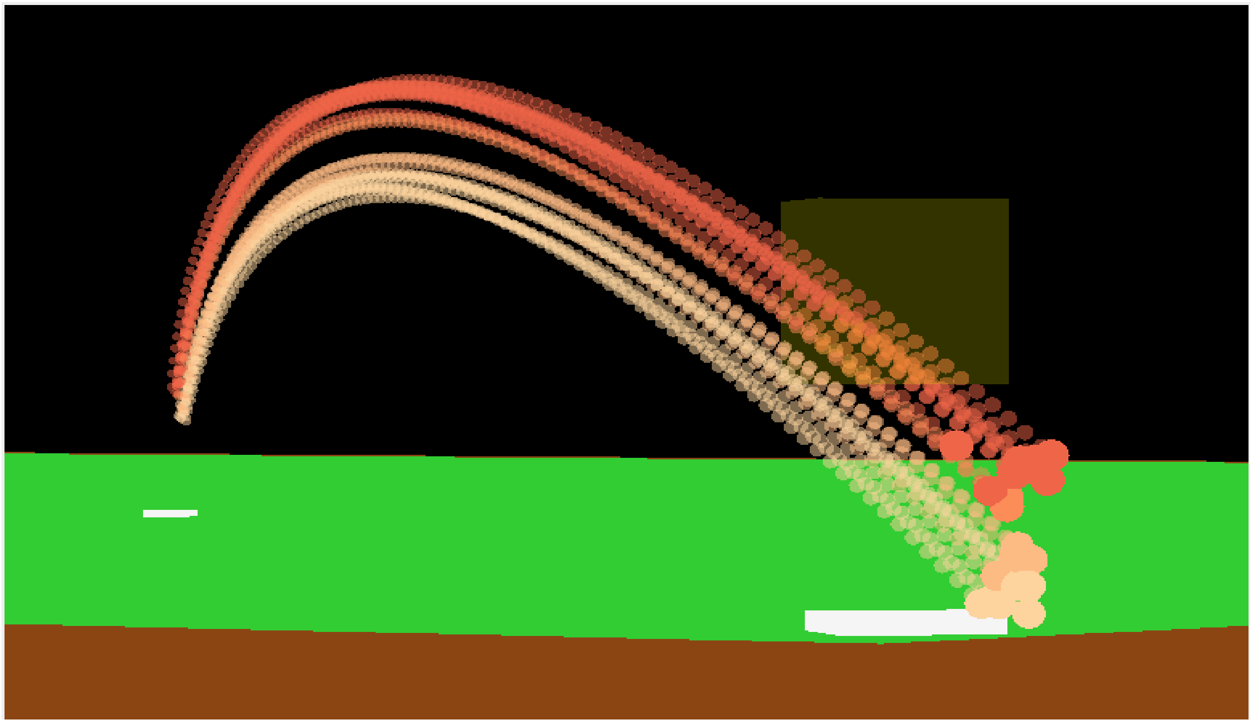


Figure 19: Multiple pitches drawn at once, Color = Avg. Speed

The *highlightPath* function was implemented in the *BVAnimation* class. The function searches the objects in the scene graph for the pitch that is clicked by the user. The pitch and pitch trajectory's appearance is changed to a light green color, and the pitch information is displayed in the text area below the visualization frame. This pitch information includes the information that could not be included in the visualization, such as the wind speed, temperature, and umpire during the selected game.

4.4.3 The Strike Zone

The top and bottom position of the strike zone was stored in the *PitchData* class of each pitch in the “Pitches to Visualize” table. In order to implement the strike zone, the top and bottom positions of each pitch in the “Pitches to Visualize” table were averaged. The average value was used to change the size and position of the strike zone box whenever the “Draw” button is clicked.

4.5 The User Interface

The user interface of the baseball visualization application was created using Netbeans, which was easier to define the placement of the different graphical elements. An initial prototype was created and was modified as additional features were added to the project. Figure 20 is a screen shot of the first prototype of the user interface.

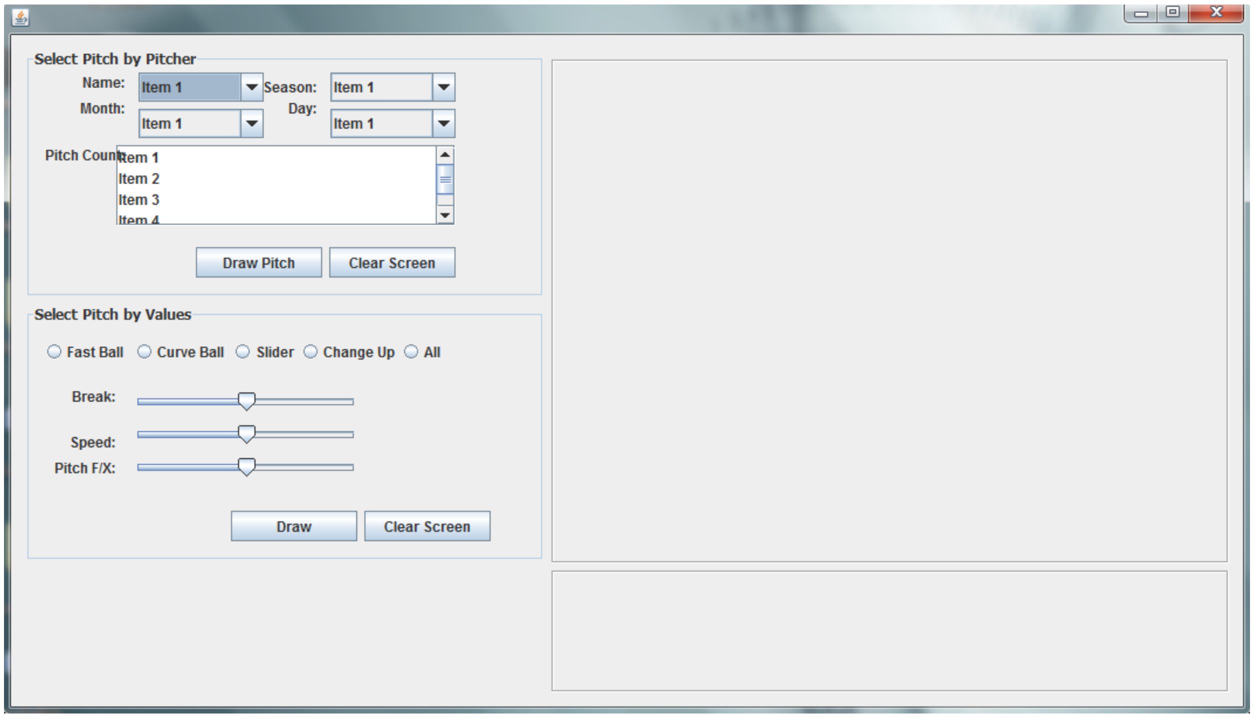
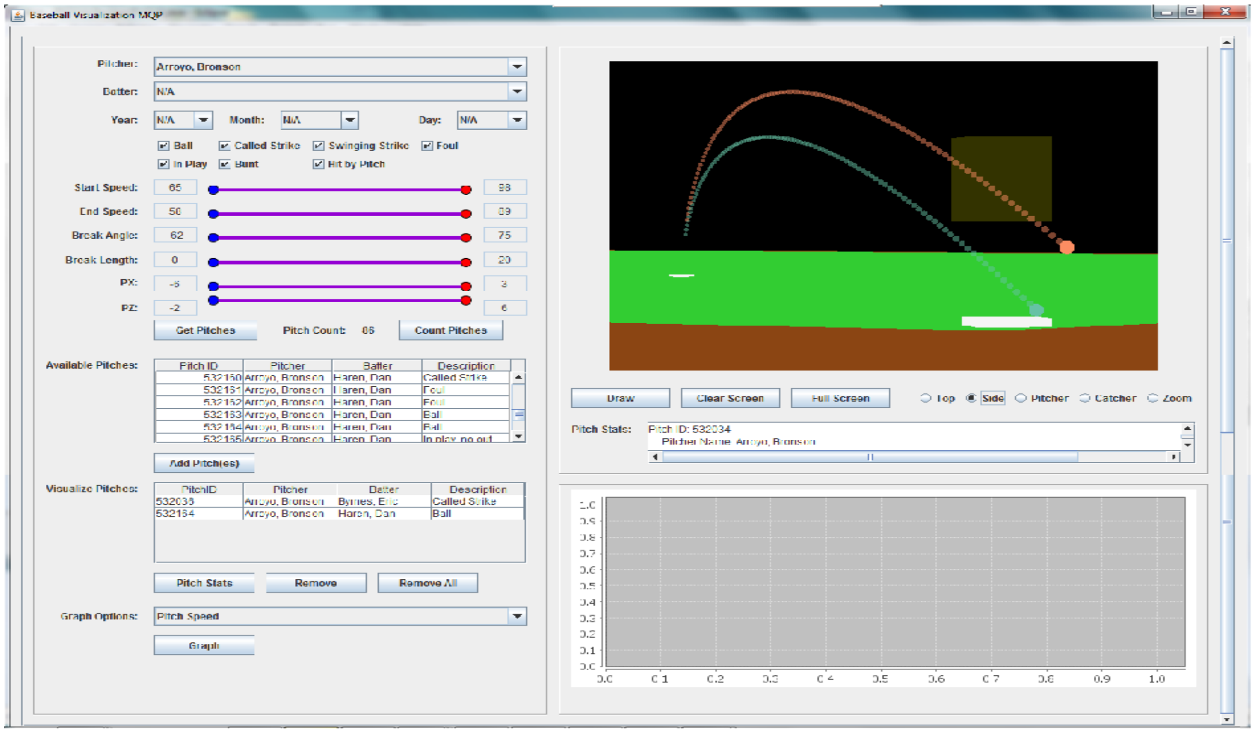


Figure 20: User interface prototype

The prototype contains two frames on the left to allow the user to select a specific pitch, by pitcher or by value. The prototype contains two panels on the right, for the visualization and the generated graphs. The user interface changed drastically from the original prototype to the final version. The basic panel layout remained consistent except for minor size changes. The pitch selection mechanism was changed drastically. Figure 21 shows the final user interface of the baseball visualization project.



Pitcher: Arroyo, Bronson
Batter: N/A
Year: N/A Month: N/A Day: N/A
[✓] Ball [✓] Called Strike [✓] Swinging Strike [✓] Foul
[✓] In Play [✓] Bunt [✓] Hit by Pitch
Start Speed: 65 – 98
End Speed: 56 – 89
Break Angle: 62 – 75
Break Length: 0 – 20
PX: -5 – 3
PZ: -2 – 6
Pitch Count: 86

Available Pitches:

Pitch ID	Pitcher	Batter	Description
532160	Arroyo, Bronson	Haren, Dan	Called Strike
532161	Arroyo, Bronson	Haren, Dan	Foul
532162	Arroyo, Bronson	Haren, Dan	Foul
532163	Arroyo, Bronson	Haren, Dan	Ball
532164	Arroyo, Bronson	Haren, Dan	Ball
532165	Arroyo, Bronson	Haren, Dan	In play, no out

Visualize Pitches:

PitchID	Pitcher	Batter	Description
532036	Arroyo, Bronson	Byrnes, Eric	Called Strike
532164	Arroyo, Bronson	Haren, Dan	Ball

Graph Options: Pitch Speed

( ) Top [● Side] ( ) Pitcher ( ) Catcher ( ) Zoom

Pitch Stats: Pitch ID: 532034
Pitcher Name: Arroyo, Bronson

Figure 21: Final baseball visualization user interface

The interface was created with three different panels placed inside a frame. The pitch selection panel is located on the right, the visualization panel on the top left, and the graph panel on the bottom left. The pitch selection panel allows the user to select a subset of pitches from the baseball database. The visualization panel displays the baseball animations, and the graph panel contains graphical representations of the data.

4.6.1 The Pitch Selection Panel

The Pitch Selection Panel allows the user to find and select specific sets of pitches to visualize. The panel was separated into two different sections; the top of the panel allows pitch selection from the database, the bottom of the panel allows the user to specify which pitches are drawn. Figure 22 shows a screen shot close-up of the top portion of the pitch selection panel.

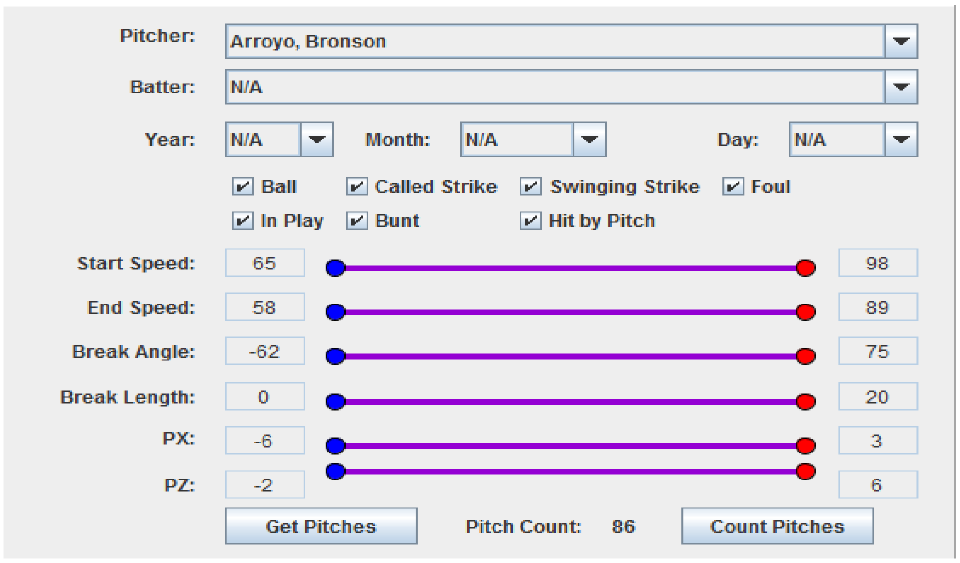
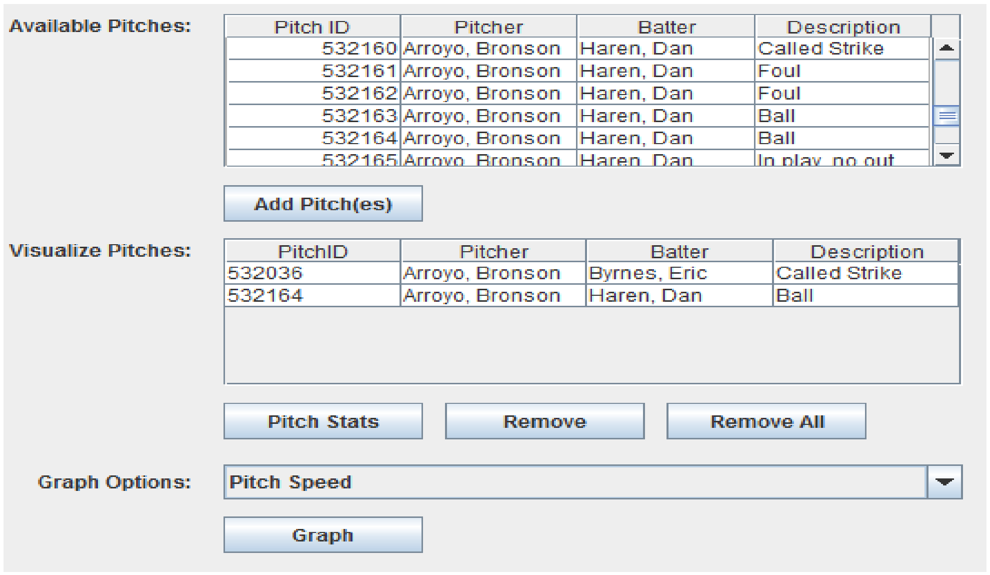

Pitcher: Arroyo, Bronson
Batter: N/A
Year: N/A Month: N/A Day: N/A
✓ Ball ✓ Called Strike ✓ Swinging Strike ✓ Foul
✓ In Play ✓ Bunt ✓ Hit by Pitch
Start Speed: 65 98
End Speed: 58 89
Break Angle: -62 75
Break Length: 0 20
PX: -6 3
PZ: -2 6
Get Pitches Pitch Count: 86 Count Pitches

Figure 22: Screenshot of the top half of the pitch selection panel

This portion of the panel contains the search parameters that are required to find specific subsets of pitches. The panel allows the selection of pitches by pitcher, batter, year, call, and the pitch data values, such as start speed, px and pz. Px is the location of the pitch as it crosses home plate in the x direction, pz is the location of the pitch as it crosses home plate in the z direction. The pitcher, batter, and year combination boxes are populated with the current list of all of the pitchers, batters, and dates that are retrieved from the database. For the slider values, the minimum and maximum values of each variable are retrieved from the database. A unique controller class is attached to each combination box that listens for changes. If the pitcher is changed to a pitcher other than “N/A”, the list of batters changes to represent only the batters that have batted against the specified pitcher. If no search parameters are specified, all of the pitches in the database are returned. The interface set up allows the user to create pitch subsets, such as the pitches of one pitcher, one batter, the strikes, the pitches in 2008, the pitches with a start speed of at least 95 mph, or any combination of the different selection attributes. The "Count Pitches" button returns the count of the pitches that match the search criteria. The "Get Pitches" button returns the specified subset of pitches. The pitches that are retrieved from the database replace the old contents of the "Available Pitches" table, contained in the bottom half of the pitch selection panel that is displayed in Figure 23.



Pitch ID	Pitcher	Batter	Description
532160	Arroyo, Bronson	Haren, Dan	Called Strike
532161	Arroyo, Bronson	Haren, Dan	Foul
532162	Arroyo, Bronson	Haren, Dan	Foul
532163	Arroyo, Bronson	Haren, Dan	Ball
532164	Arroyo, Bronson	Haren, Dan	Ball
532165	Arroyo, Bronson	Haren, Dan	In play, no out

PitchID	Pitcher	Batter	Description
532036	Arroyo, Bronson	Byrnes, Eric	Called Strike
532164	Arroyo, Bronson	Haren, Dan	Ball

Graph Options: Pitch Speed

Figure 23: Screenshot of the bottom half of the pitch selection panel

The bottom half of the pitch selection panel contains two tables, the "Available Pitches" table and the "Visualize Pitches" table. The two tables contain a list of *Pitch* elements. The pitches in the "Available Pitches" table can be sorted based on any column, Pitch ID, Pitcher, Batter, and Description; any number of these pitches can be added to the "Visualize Pitches" table. The reason for two separate tables is so pitches from different pitchers can be retrieved and drawn to the screen. The "Available Pitches" table changes depending on the search results. The

"Visualize Pitches" table remains consistent until pitches are added or removed. The bottom half of the pitch selection panel also contains the graphing options.

In order to prevent the user from exhausting the available memory used to run the application, limits were placed on the number of pitches that can be drawn and graphed. When the user attempts to draw more than 25 pitches or graph more than 500 pitches, a warning window appears preventing the user from drawing or graphing those pitches.

4.6.2 The Visualization Panel

The visualization panel was created to display information and options regarding the pitch animation. Figure 24 shows an example of the visualization panel.

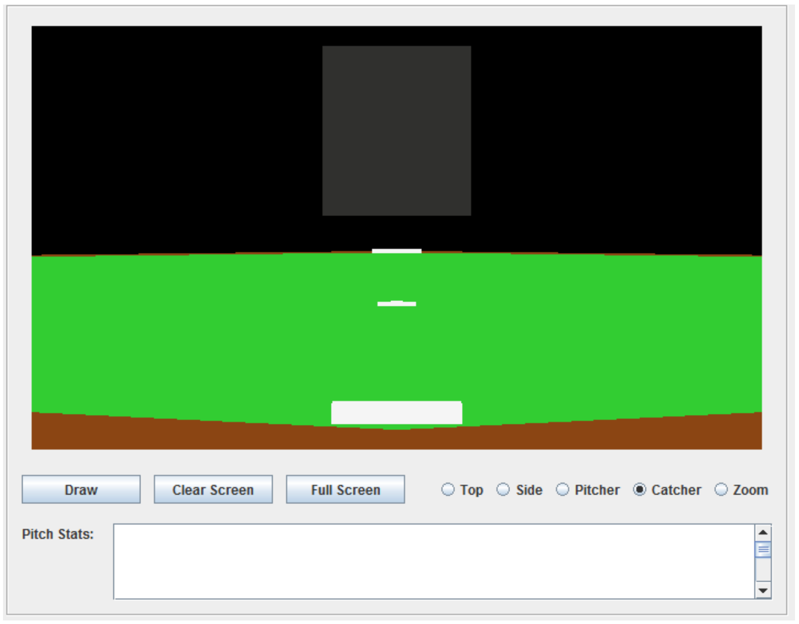
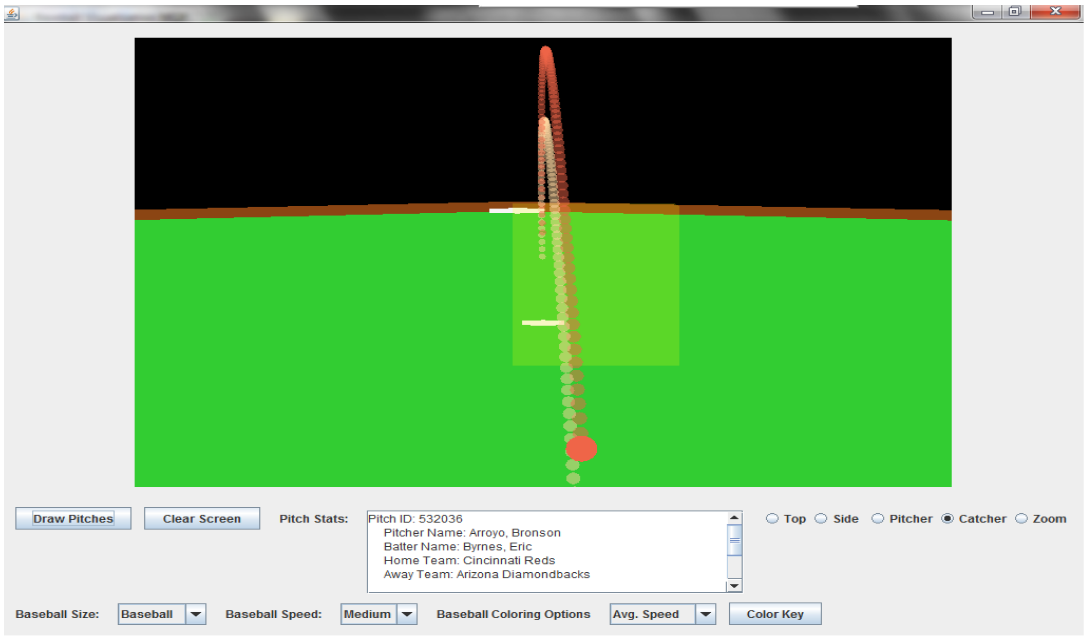


Figure 24: The visualization panel

Simple Java Swing class elements, such as JButtons, JRadioButtons, and JTextAreas, were used to implement the desired features of the visualization panel. A *DrawPitchesClick* class was created to handle clicking on the "Draw" button. The *DrawPitchesClick* reads the information in the "Visualize Pitches" table, calls the *Calc* class to perform the trajectory calculation on the set of *PitchData,* and transmits the results to the *BVAnimation* class, that animates those pitches. A *ClearScreenClicked* class was created to remove all of the pitch objects from the animation, when the "Clear Screen" button is clicked. The *ViewButtonGroupChange* class was created to listen for any changes to the current view. If a change to the view is detected, the camera position of the animation is transformed. The visualization panel is limited by space constraints; therefore the full screen window was implemented. The *FullScreenClick* class listens for a mouse click on the "Full Screen" button and opens the full screen window if the button is clicked.

4.6.2.1 The Full Screen Window

The full screen window was implemented to emphasize the pitch animation and introduces additional features, such as animation speed and ball sizing. Figure 25 provides a screen shot of the full screen window of the baseball visualization application.



Pitch ID: 532036
Pitcher Name: Arroyo, Bronson
Batter Name: Byrnes, Eric
Home Team: Cincinnati Reds
Away Team: Arizona Diamondbacks

Top Side Pitcher [✓ Catcher] Zoom

Baseball Size: Baseball
Baseball Speed: Medium
Baseball Coloring Options: Avg. Speed

Figure 25: Full screen window example

A new window was created containing two panels. The top panel contains the animation and the bottom panel contains both standard and additional pitch options. The standard pitch options include the "Draw Pitches" and "Clear Screen" buttons, as well as the view radio button, and are directly derived from the corresponding features in the visualization panel. The additional pitch options include animation modifications that are unique to the full screen window, such as changes in the baseball size, animation speed, and coloring options.

Three different baseball size options are supported by the visualization: baseball, softball, and beach ball. The user changes the baseball size by selecting another option from a simple drop down menu. The baseball size is a parameter that is passed to the *BVAnimation* class by the *DrawPitchesClick* class. The *BVAnimation* class provides the function *addSphere*, which uses the size parameter in order to create new baseball spheres reflecting the selected size. The different sizes are implemented to increase pitch visibility. Figure 25 shows the baseballs drawn with the default size “Baseball”. Figure 26 shows the baseballs drawn with the largest size, “Beachball”, selected. The shading appearance added to the generated spheres does not work correctly, so the spheres appear without shading.

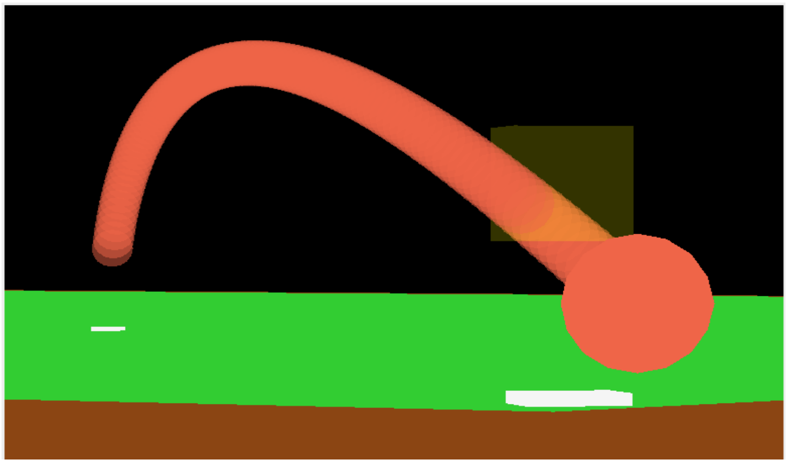
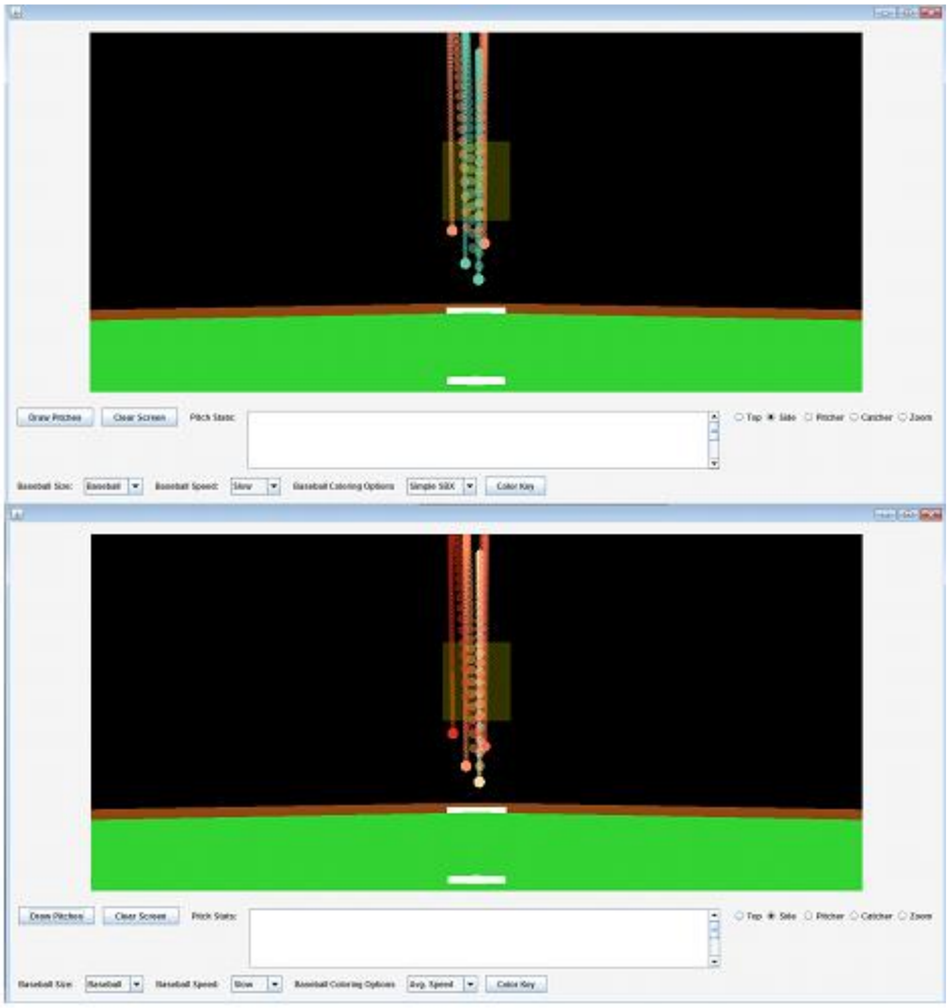


Figure 26: Beachball size option

The user selects the baseball’s speed and coloring options using the same method as baseball size selection. The speed and coloring options are passed to the *BVAnimation* class. The three different speed options supported by the system are slow, medium, and fast. The fast speed option is the full speed of the pitch. The medium and slow speeds are implemented as fractions of the full speed. The *BVAnimation* class extends the time of each frame depending on the currently selected speed option. Pitch speed variation allows the analysis of pitch trajectories at full pitch speed, as well as a slower speed to extract different types of data from the path.

There are three different pitch trajectory coloring options currently defined in the baseball visualization system: simple SBX (strike, ball, out), detailed SBX, and average speed. An example of pitch trajectories colored using the simple SBX coloring option is displayed at the top of Figure 27. The getColor function defined in the *BVAnimation* class retrieves the appropriate color of each baseball, depending on the selected coloring mode. The bottom of Figure 27 displays an example of pitch trajectories that are drawn using the average speed coloring option.



Baseball Size: Baseball
Baseball Speed: Slow
Baseball Coloring Options: Simple SBX

Baseball Size: Baseball
Baseball Speed: Slow
Baseball Coloring Options: Avg. Speed

Figure 27: Baseball coloring options. Simple SBX top, speed bottom

Color charts, or keys, are provided as a separate pop-up window so that the user can understand the different pitch colors.

Closing the full screen window allows the user to return to the original baseball visualization screen. The changes made to the animation on the full screen window are reflected in the original window.

4.6.3 The Graph Panel

The graph panel was created in order to provide a graphical representation of the data, similar to the information provided by the Pitch FX tool. The *ScatterPlot* class, that uses the JFree Chart library, was created to generate the specified charts. The pitch selection panel allows the selection of two different graph options, “Pitch Speed” and “Release Point”. The *GraphClick* class was created to populate and display the specified graphs when the “Graph” button on the pitch selection panel is clicked. The *GraphClick* class creates a *ScatterPlot* object and specifies the horizontal and vertical axis labels, as well as the plot’s title. Within the *GraphClick* class, the required data fields are extracted from each *PitchData* object in the “Pitches to Visualize” table. The fields are added to the data sets of the *ScatterPlot*, and the graph is generated and displayed within the graph panel. Figure 28 shows an example of the graph panel, in which a “Pitch Speed” graph, populated with pitch data from approximately 50 different pitches, is displayed.

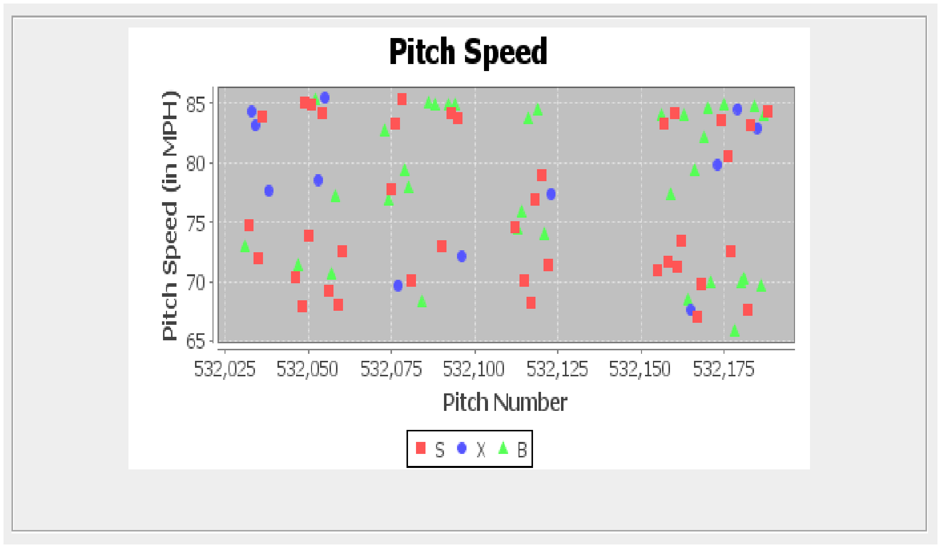


Figure 28: Graph panel, pitch speed graph

The final baseball visualization application was exported from eclipse as an executable jar file. The application can be run on any system with Java3D installed. The next section provides a description of the user testing strategy to quantify the effectiveness and validity of the baseball visualization application.

5. RESULTS

User testing was conducted to determine the effectiveness of the baseball visualization application. Due to time and resource constraints, the participants of the testing were selected based on proximity and willingness to participate. A total of ten test subjects participated in the project tests. More extensive user testing should be conducted with a user group primarily consisting of baseball players and coaches, in order to quantify the project's effectiveness at providing useful information.

The user testing consisted of three different phases: pre-test, test, and post-test. The test was administered in a quiet, secluded location to reduce distractions, and took approximately 20-25 minutes.

During the pre-test phase, the user was given a consent form and a brief description of the application. The project description included a short introduction to the baseball visualization project, the project purpose, an explanation of the purpose of the testing, and a summary of the upcoming tasks. A pre-test questionnaire was administered to determine the subject's background, experience in baseball, and familiarity with baseball visualizations. The results of the pre-test questionnaire are discussed in section 5.1 Pre-Test Results.

During the second phase of the user testing scenario, the user was introduced to the baseball visualization application and asked to spend a few short minutes to explore the program. The subject's actions were observed, particularly in response to how the test subject explored the system, and recorded for analysis. The user was then given a sequence of tasks to follow that were designed to introduce the different aspects and features of the project. The tasks consisted of selecting a pitcher, selecting a batter, changing the slider information, drawing pitches, and exploring the options included in the full screen window, such as changing the ball size. Each test subject was monitored as the subject explored the visualization application and completed each of the tasks. The observed behavior of each of the test subjects was recorded for analysis and discussion. The results of the project exploration and task administration phase are discussed in section 5.2 Task Discussion and Observation.

During the third and final phase of the testing, a post-test questionnaire was administered. The purpose of the questionnaire was to give the test subject a chance to evaluate their experience, and suggest improvements that could be made to the project. Section 5.3 discusses the post-test questionnaire results.

Examples of the forms, questionnaires, and tasks that were administered to the participants are provided in Appendix A for reference.

5.1 Pre-Test Results

The pre-test was administered to understand the test subject's familiarity with baseball in order to compare the results between players, fans, and other users. In order to understand each test subject's background, various questions were asked and the subject was expected to give a number ranking of their experiences. The possible answers to the following questions were 1: never, 2: rarely, 3: occasionally, 4: often, 5: daily.

Question 1: How often do you use computers?

Question 1 was administered to ensure that the test subject's computer knowledge was adequate to use the information, in order to understand any possible outliers in the task administration phase of the testing. Figure 29 shows a simple bar graph representation of the user response to the question. The general trend in the graph shows the test subjects are proficient with computer applications.

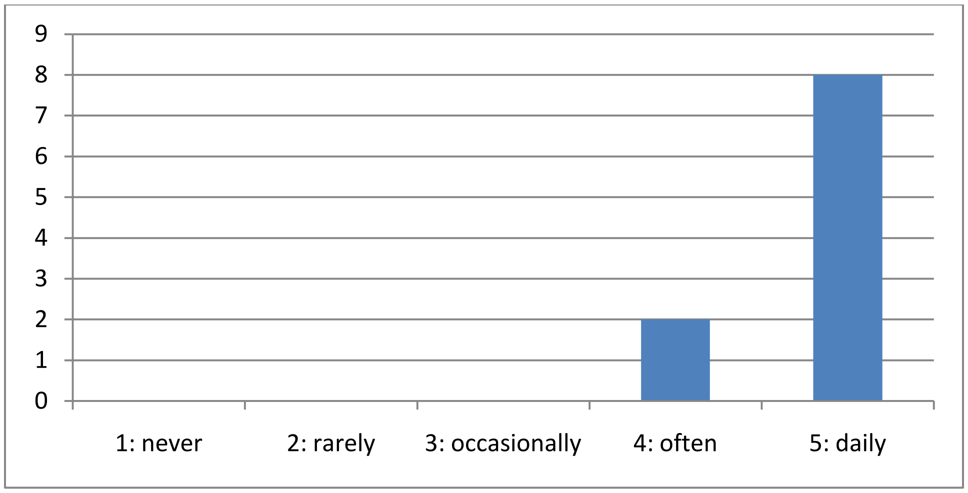


Figure 29: Pre-test question 1

The vertical axis represents the test subject's response; the horizontal axis represents the number of subjects that responded to each option. The horizontal and vertical axes remain consistent in all of the pre-test question response graphs.

Question 2: How often do you watch baseball games?

The second question in the pre-test questionnaire was created understand the test subject's familiarity of the game. The responses to the second question are used to compare the task results between subjects with various levels of familiarity to baseball. Figure 30 shows that there is approximately an equal distribution of baseball fans, and individuals that rarely watch baseball.

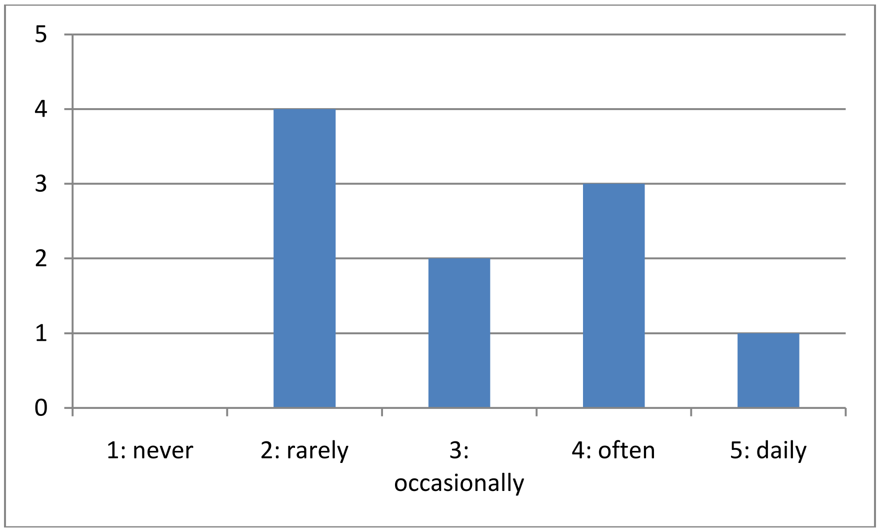


Figure 30: Pre-test question 2

Question 3: How often do you visit MLB.com?

The third question was designed to evaluate the test subject's experience with MLB.com to analyze the subjects interest in baseball statistics, and possible interest in a baseball visualization application.

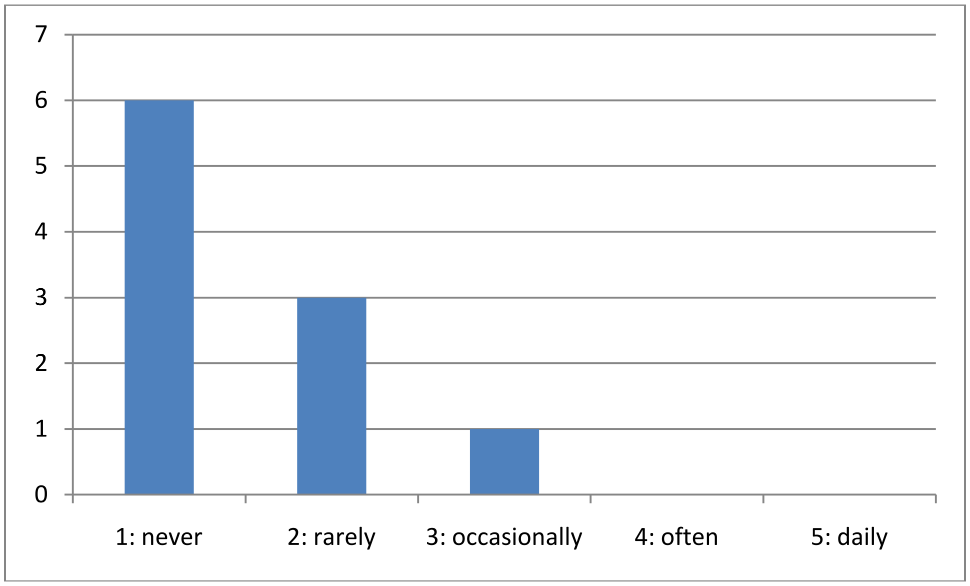


Figure 31: Pre-test question 3

The majority of the test subjects do not use MLB.com

The remaining two questions in the pre-test questionnaire warranted an open ended response.

Question 4: Have you ever played Baseball? If so what position?

The fourth question was created in order to validate the effectiveness of the baseball visualization application to baseball players. Fifty percent of the test subjects answered that they had some experience playing baseball. Three subjects played in the outfield, one subject played second base, and one subject was a pitcher. The remaining five test subjects have never played.

Question 5: What do you expect from a baseball visualization application?

The final question allows the test subject to disclose what basic expectations they have of baseball visualization applications, prior to seeing the application. These expectations included fun, a visual application that is easy to understand, a way to watch the baseball, and a way to understand the different pitch types because they all look the same.

5.2 Task Discussion and Observation

I observed each of the test subjects as they explored the visualization application. The set of tasks started with selecting a pitcher and drawing a strike pitched by that pitcher. The tasks then provided an exploration of the full screen window, as the user was asked to change the strike's size, speed and coloring options. The later tasks involved selecting pitches based on a specific pitch subset, retrieving and reading pitch information to find the inning, deleting the pitches from the screen, and generating pitch speed graphs. The tasks tested the user on all of the basic functionality of the system. After completing the prescribed tasks, the user was given the option to explore the system in greater detail.

For the first task, selecting a pitcher and drawing a strike pitched by that pitcher, various test subjects had different difficulties in completing the task. All ten subjects were able to locate the pitcher field, scroll through the list of available pitchers, and select a pitcher relatively quickly. Drawing a strike proved more difficult for a portion of the test subjects. Three subjects needed help retrieving the pitches, and preparing the pitches to be drawn. Four subjects were able to work through the interface to draw the pitches, but took a longer amount of time. The remaining three test subjects were able to draw the strike with relative ease.

The second task involved opening the full screen window, and manipulating the pitch based on the different size, animation speed, and coloring options. This task proved relatively easy for the majority of the test subjects. Two subjects took longer than the others to locate the full screen window, but once the window was open they were able to change the options with relative ease. The greatest issue that all ten subjects had in completing this task involved switching between the two windows. A subject would open the full screen window, return to the original window and the visualization would disappear. The observation lead to the conclusion that the window changing mechanism is confusing, and a different implementation option for screen switching should be explored.

The third set of tasks that were administered during the task phase of the user testing appeared easier for the subject to complete. The subjects were more familiar with the simple structure of the baseball application; therefore it was easier to use the more advanced options, including graphing and removing pitches. Four of the subjects found the mechanism for reading pitch information hard to locate because of the separation of the mechanism and the produced result on two different panels.

Based on the observations throughout the task phase of the testing, one major area of the application that confused the user was view manipulation. It appeared difficult for the user to move the view around with the panning and zooming functions. The current settings of the panning functionality are not intuitive. The camera only rotates about a single point. This observation, in addition to post-test questionnaire results, leads to the conclusion that the panning functionality should be modified to be more intuitive, or removed completely.

5.3 Post-Test Questionnaire Results

A post-test questionnaire was administered to collect the test subject's responses and suggestions in regards to the baseball visualization application. The questions asked in the post-test questionnaire included a measure of the subject's experience using the application, any suggested improvements to the application, any confusing aspects or pieces of the application that seemed out of place, the difficulty of using the application, as well as suggestions for future additions or deletions from the application. These questions were administered as open response questions. The following describes the overall trends formed from combining the results of the questions from the ten test subjects.

The majority of test subjects found the application confusing at first. “It was a little confusing. I had a little trouble at first, but figured it out.” One user suggested that a short introductory tutorial explaining the controls of the project would be very helpful to remove any confusion. For future releases of the project there should be an accompanying tutorial in order to reduce the initial confusion of the application. Once the test subject figured out how to use the application it was relatively easy to use. Nine of the subjects answered that no part of the application seemed out of place. The tenth responded that beach balls seemed out of place.

There were multiple suggestions for improvements that can be made to the baseball visualization application. One subject suggested that when baseballs are added to the “Visualize Pitches” table, they should be removed from the “Pitches to Visualize” table, because the pitches are no longer available.

The suggestions for additions and deletions were analyzed and used to make suggestions for future additions to the application. The next section describes some of the changes and future additions that can be made, as well as summarizes the entire project.

6. CONCLUSION

There are aspects of the project that require further improvements. These aspects were discovered during the analysis of the user testing data, and include the stopping location of the baseball in the animation, and the camera views. There are also features that can be implemented in future releases of the project, including increased performance, enhanced graphics, and new models.

The baseball's position is calculated past the location of home plate, portraying the baseballs lower than they should. One possible solution to the baseball end position issue could be to stop the trajectory calculations prematurely. The current camera positions of the top and side views are calibrated incorrectly. In the top view, it is difficult to see the pitches, and in the side view, the angle of the camera skews the visualization. The camera position and the position of the plane of projection could be translated and rotated in order to create a closer, more realistic view of the pitch animations.

In future implementations of the visualization project, methods for increasing system performance could be considered. Currently, speed is not a prominent concern, but the issue is enlarged when the database is populated with all of the available pitch data. Methods to reduce the processing time of the application include optimizing the queries, and using alternate, more efficient data structures to store the pitch information.

The pitch type (e.g., curve ball, fast ball, slider, and change up) is not explicitly defined in the baseball visualization database of pitch information. There was no easy way to distinguish the different pitch types; therefore the pitch types could not be included in the visualization. In future implementations of the visualization, pitch type classification could be supported, and the user would be able to specify search queries on the pitch type. An additional coloring schema based on pitch type could also be added to the project in a future implementation. Pitch selection based on inning could also be included.

In future implementations of the visualization project, the graphics could be enhanced. For example, a baseball stitch texture could be attached to each baseball's corresponding appearance, creating a more realistic baseball model. Shadows could also be added to the scene to increase the realism of the field. The realism of the pitch trajectory could also be increased by implementing the rotation of the pitch. Additional objects could be inserted into the scene, including a catcher, pitcher, and batter model.

Ultimately the visualization could be extended to show the location of the pitch after interaction with the batter. The data was not available at the time this visualization project was implemented. It would also be interesting to create simulations or short animations of batter and pitcher movement, as well as the pitch being released from the pitcher's hand. Motion capture can be demonstrated for hand motion during pitching, as well as the flight trajectory of a baseball [1], therefore adding pitcher hand movement animations to the visualization would be a wonderful addition to the visualization, as well as a step closer to realistic representation of the pitch.

The purpose of this Major Qualifying Project was to create an application to display 3D pitch information. Visualization techniques and tools to enhance the baseball visualization project were researched. A baseball visualization database was created to store the pitch information that was downloaded from MLB.com. Queries were specified to search the database for specific pitches. An interface was created that allows the user to specify the search criteria and to view the resulting visualization of the selected pitch information. After the construction of the baseball visualization application, a small user testing scenario was conducted. System modifications and future features were suggested based off user testing results. The baseball visualization application can be used by fans to analyze and visualize user selected subsets of pitch data.

7. WORKS CONSULTED

[1]. BrooksBaseball.net PitchFX Tool. (n.d.). Retrieved September 27, 2009 from http://brooksbaseball.net/pfx/index.php

[2]. Fast, Mike. (2007). How to Build a Pitch Database. Retrieved April 16, 2010 from http://fastballs.wordpress.com/2007/08/23/how-to-build-a-pitch-database/

[3]. Flare Data Visualization for the Web, (n.d.). Retrieved April 11, 2010 from http://flare.prefuse.org/

[4]. Gueziec, A. (2002). "Tracking Pitches for Broadcast Television". *Computer*, 35(3), 38-43.

[5]. Heer, J., Card, S., & Landay, J. (2004) Prefuse: A Toolkit for Interactive Information Visualization. 1-10. Retrieved September 27, 2009 from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.80.3032

[6]. JUNG Java Universal Network/Graph Framework, (n.d.). Retrieved April 2010, from http://jung.sourceforge.net/

[7]. McConnell, Jeffrey J. (2006) *Computer Graphics Theory Into Practice.* Jones and Bartlett Publishers: Sudbury, Massachusetts. 77- 93.

[8]. MLB.com Gameday. (n.d.). Retrieved September 27, 2009 from http://www.mlb.com/mlb/gameday/

[9]. OpenDX. (n.d.). Retrieved April 10, 2010 from http://www.opendx.org/

[10]. Parent, Rick, (2008) *Computer Animation Algorithms & Techniques*. Elsevier: Boston. 241 -246, 425.

[11]. Post, F., & Walsum, T. (1993) "Fluid Flow Visualization". *Focus on Scientific Visualization*. Springer Verlag: Berlin, 1-40.

[12]. The Prefuse Visualization Toolkit. (January 24, 2009). Retrieved September 18, 2009 from http://www.prefuse.org/

[13]. Visualization Toolkit. (n.d.). Retrieved September 18, 2009 from http://www.vtk.org/VTK/project/about.html

8. APPENDIX A: User Testing Documents

CONSENT FORM

WPI Baseball Visualisation MQP Testing

Thank you for considering participating in this study. Research is being conducted to test the qaulity of this application. This study is being conducted by Jennifer Hunt from WPI.

Over the next few minutes you will be asked to follow a provided set of tasks. You will also be given a few minutes for free experimentation. Following this, you will be asked some questions about the difficulty of the tasks and any suggestions you think would make the application easier to use.

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 researcher 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.

By signing below you acknowledge that you may not gain anything personally by participating in the experiment.

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.

YOUR SIGNATURE BELOW INDICATED THAT YOU HAVE READ THE INFORMATION ABOVE AND YOU ARE CONSENTING TO PARTICIPATE IN THE EXPERIMENT DESCRIBED ABOVE.

Participant's Signature Date

Participant's email address

I have explained in detail the procedure for this experiment to the participant and, if asked, have made a copy available for the participant to keep. The participant has agreed to participate by signing above. My signature also confirms that the experiment was carried out as described.

Researcher Signature Date

Pre-Test Questionnaire

This feedback form is being used to determine the effectiveness of the baseball visualization application developed during the project. Please complete the questionnaire to the best of your abilities. Thank you.

1. How often do you use computers?

1: never
2: rarely
3: occasionally
4: often
5: daily

2. How often do you watch baseball games?

1: never
2: rarely
3: occasionally
4: often
5: daily

3. How often do you visit MLB.com?

1: never
2: rarely
3: occasionally
4: often
5: daily

4. Have you ever played Baseball? If so what position?

5. What do you expect from a baseball visualization application?

Tasks

These tasks are designed to determine the effectiveness of the baseball visualization application developed during the project, and to determine any changes that can be made to make the application better.

Please spend a few minutes experimenting with the application. Note: This application only contains pitches from a limited number of games for testing purposes.

1. Select your favorite pitcher.
2. How many pitches did this pitcher pitch?
3. Draw a strike.
4. Open full screen.
5. Change the ball size, speed and color.
6. Select the trajectory.
7. Select a batter.
8. Find all the pitches pitched to that batter that are 85 – 90 mph.
9. Select a pitch that was a ball, what inning was this pitched in?
10. Delete all the pitches drawn to the screen.
11. Graph the pitch speed of all the pitches in the pitches to draw table.

Post-Test Questionnaire

This feedback form is being used to determine the effectiveness of the baseball visualization application developed during the project. Please complete the questionnaire to the best of your abilities. Thank you.

1. How was your experience?

2. Can you make any suggestions to improve the application?

3. Was there any part of the application that was confusing? If yes, what was confusing?

4. Was there anything that seemed out of place?

5. How difficult was the application to use?

6. What would you like to see added to the application? (If anything)

7. What would you like to see removed from the application? (If anything)

Comments?