Project Title: **Wireless Data Collection Retrievals of Bridge Inspection/Management Information**

Research Reference Number: **MDOT # 2013-0067, Auth. No. 2 (R1, R2)**

Proposing Research Agency:

Michigan Technological University
1400 Townsend Drive
Houghton, MI 49931

Principal Investigator(s):

PI: Colin N. Brooks, MEM
Environmental Science Lab Manager, Research Scientist
Michigan Tech Research Institute (MTRI)
3600 Green Court, Suite 100, Ann Arbor, MI 48105
734-913-6858 ◆ cnbrooks@mtu.edu

Co-PI: Theresa (Tess) M. Ahlborn, Ph.D., P.E., FPCI, FACI
Professor, Civil and Environmental Engineering
Director, Center for Structural Durability
1400 Townsend Drive, Houghton MI 49931
906-487-2625 ◆ tess@mtu.edu

Contracting Authority:

Lisa Jukkala
Manager, Government Contracts/Training, Sponsored Programs
Michigan Technological University
1400 Townsend Drive, Houghton, MI 49931
906-487-2226 ◆ 906-487-2245 fax ◆ lajukkal@mtu.edu

Revised February 28, 2017

1. Report No. RC-1634	**2. Government Accession No.** N/A	**3. MDOT Project Manager** Rich Kathrens
4. Title and Subtitle Wireless Data Collection Retrievals of Bridge Inspection/Management Information		**5. Report Date** 2/28/2017
		6. Performing Organization Code 38-6005955
7. Author(s) C. Brooks, R. Sawtell, G. Sullivan, R. Dobson, T. Ahlborn, N. Jessee, H. Kourous-Harrigan, S. Aden		**8. Performing Organization Report No.**
9. Performing Organization Name and Address Michigan Technological University 1400 Townsend Drive Houghton, MI 49931		**10. Work Unit No. (TRAIS)** N/A
		11. Contract No. 2013-0067
		11(a). Authorization No. No. 2 (R1, R2)
12. Sponsoring Agency Name and Address Michigan Department of Transportation Research Administration Section 425 West Ottawa Street Lansing, MI 48933		**13. Type of Report and Period Covered** Final Report (Revised), including Phase II through 2/28/2017
		14. Sponsoring Agency Code N/A

15. Supplementary Notes
This final report includes the content of the first report delivered Sept. 30, 2015, with updates to reflect Phase II work completed through Feb. 28, 2017.

16. Abstract

To increase the efficiency and reliability of bridge inspections, MDOT contracted to have a 3D-model-based data entry application for mobile tablets developed to aid inspectors in the field. The 3D Bridge App is a mobile software tool designed to facilitate bridge inspection processes by enabling inspectors to enter element-level bridge condition data using 3G/4G network-enabled tablet devices. The system collects information from MDOT's bridge management database, and then renders a dynamic, interactive 3D model of the desired bridge. The bridge inspector is able to record the locations and attributes of new defects in an element-level form by touch interaction and manipulation of the 3D model. The interactive model, marked up with existing defects, also allows for bridge inspectors to better visualize past inspection data. The inspector can also take pictures of the defects using the tablet's camera, as well as record comments. This gives users further insight into the progression of the defect over time. The bridge inspector is able to navigate along the bridge model just as he or she would during a normal inspection. Results can be integrated into bridge management workflows.

17. Key Words Bridges, tablets, inspection, element, 3D, database, data, interactive, Unreal Engine 4		**18. Distribution Statement** No restrictions. This document is available to the public through the Michigan Department of Transportation.	
19. Security Classification - report Unclassified	**20. Security Classification - page** Unclassified	**21. No. of Pages** 101	**22. Price**

Acknowledgments

The project team would like to thank and acknowledge the MDOT program manager, Rich Kathrens, the research manager, Michael Townley, and the members of the research advisory committee for their advice and oversight during this project. The help of Michigan Department of Technology, Management and Budget staff in detailing database integration requirements was also appreciated. The PIs would like to thank their project team of Reid Sawtell (lead developer), Glenn Sullivan, Helen Kourous-Harrigan, Rick Dobson, Sam Aden, and Nate Jessee for their dedication to helping MDOT rapidly, safely, and effectively collect element-level bridge condition data through this new tool. The help of MDOT staff, especially bridge inspectors such as Janiene Devinney, and private companies such as Great Lakes Engineering Group in increasing our understanding of real-world inspection processes was critical to the success of the project. We thank everyone who advised the project and met with our team.

Disclaimer

Table of Contents

List of Figures

List of Tables

Executive Summary

Current bridge inspection practices at the Michigan Department of Transportation (MDOT) utilize paper forms followed by a manual data entry step to populate the Bridge Management System (BMS) database with information needed for bridge management and repair. Faced with an aging bridge inventory and increasing federal regulations regarding collection of element-level data, MDOT wishes to increase the efficiency and reliability of collected data. To achieve this, MDOT requested a 2D/3D application that can utilize mobile tablet technology to aid inspectors in the field.

To develop this application, a Michigan Technological University applied research team, led by staff from the Michigan Tech Research Institute (MTRI), first examined the state of practice across the nation to better understand currently available options. They found that as of 2014, no application assisted with collection of element-level data. Next, MTRI met with experienced bridge inspectors (from the consulting firm Great Lakes Engineering Group as well as MDOT staff inspectors) to better understand the needs of bridge inspectors so the application design could be tailored to their input.

Because MDOT does not have 3D bridge models available for all bridges, MTRI developed a server application using Django (a Python web framework) to generate Extensible Markup Language (XML) files using data from MDOT's BMS database. Each XML file provides a generic bridge model that is sufficiently representative for inspection purposes; it contains information about the element-level components of a bridge, including location and size. The server application includes a user tuning component to correct initial erroneous assumptions due to lack of information, such as placement of bearings per beam.

To produce the client application, MTRI selected the Unreal Engine 4 (UE4) game engine by Epic Games to provide cross-platform rendering capability. The application itself is built using C++ interfaced with the UE4 engine, as well as UE4 Blueprints for high-level functionality. It uses Java for integration with native camera functionality on Android devices, and Objective-C for iOS devices. The client application receives a XML file from the server application and constructs an interactive 3D model. Using a set of intuitive navigational views, the inspector can traverse the bridge and mark the surface of the model with element-level defect information, photos, and comments. Defect markers are proportionally sized based on the defect quantity and are color-coded to match condition states. The application also has a summary view for reviewing the aggregate defect information and for editing National Bridge Inventory (NBI) ratings.

The project's second phase focused on further development to bring the application closer to implementation. MDOT-requested enhancements included import/export XML functionality to enable integration of inspection results with MDOT's BMS database, NBI reporting functionality, and element transparency. A potential third phase would focus on moving the app into day-to-day usage

by MDOT, with the potential to bring the tool into national usage by working with the American Association of State Highway and Transportation Officials (AASHTO) to integrate it into AASHTOWare. Recommendations included in the Implementation Action Plan for a potential third phase include fully integrating the app with MDOT's BMS database, updating the app with key features suggested during user testing, enabling the app to support a wider set of bridges, and moving into the deployment phase so that MDOT can start using the tool as part of its standard inspection procedures.

1. Introduction

Collecting bridge inspection data is a key component of assessing bridge condition and managing MDOT's infrastructure. Regulations issued by the Federal Highway Administration require states to use a data-driven process to check the completeness and accuracy of bridge data and to verify compliance with the National Bridge Inspection Standards. States are also required to collect and maintain element-level inspection data as prescribed by the American Association of State Highway and Transportation Officials (AASHTO), a provision that increases the time and complexity of the inspection process.

Current inspection practices have inspectors using paper forms in the field to collect condition-state information and to provide historical reference data. These data must then be entered manually into the Michigan Bridge Inspection System (MBIS) and Michigan Bridge Reporting System (MBRS) (now both part of MiBRIDGE), which adds yet another task to the process and introduces potential for error. Photographs documenting bridge deterioration must be taken and stored as well, which requires additional documentation to be generated linking individual photographs with the locations they were taken. Finally, inspectors must carry relevant reference materials to verify the accuracy of the data they are collecting. Together, these demands burden inspectors with a growing load of devices and physical information that they must manage, often in unfavorable or hazardous conditions.

Given these issues, MDOT wishes to increase the efficiency and accuracy of the data collection process. Since mobile computing and wireless data transfer are now ubiquitous, these technologies offer a promising alternative to the current paper solution. Tablet devices are relatively inexpensive, can be made ruggedized for outdoor use or come ruggedized, can communicate directly with MDOT online services, and typically include cameras with acceptable resolution. A digital inspection process can leverage all of these features to streamline data entry, rapidly collect more detailed inspection information, and reduce the physical inventory needed by inspectors.

1.1 Objectives

This project had the following objectives:

1. Review and evaluate ongoing and recently completed research involving the bridge inspection process.
2. Review MDOT's process of collecting National Bridge Inventory (NBI) and AASHTO Element Level inspection data.

3. Develop an application to collect NBI and Element Level inspection data using visual methods and 2D drawings or 3D models of the bridge elements.
4. Develop and test a wireless data collection and display system to meet MDOT's bridge inspection and management needs which can be integrated with MDOT's existing web applications and database structure. Determine alternatives that will work on multiple mobile platforms.

1.2 Scope

To realize the overall project goal of developing an application that improves accuracy and efficiency of MDOT's bridge inspection process, the following 10 tasks were performed (Tasks 1-6 were part of Phase I, and Tasks 7-10 were added with Phase II):

Task 1: Literature Review Document
Task 2: Web/tablet application integrated with MDOT's current MBIS and MBRS Systems (now known as MiBRIDGE).
Task 3: Field demonstration of application
Task 4: Application User's Manual
Task 5: Complete documentation of the application and source code
Task 6: Final Report
Task 7: Integrate System With MDOT Database
Task 8: Finalize Cross Platform Support
Task 9: Finalize 3D Model User Tuning
Task 10: Add Support for collecting NBI Ratings

Task 1 was needed to evaluate what options currently exist. Determining how bridge inspections are carried out nationwide helped shape the application's features so it will meet or exceed MDOT's needs.

Task 2 included the development of the application itself and occurred throughout the project time frame. Task 3 was imperative for garnering feedback from inspectors and ensuring that the system was usable and successful. As Task 2 proceeded, Task 3 was executed from the first prototype of the application through the conclusion of the project.

Similarly, Task 4 was ongoing throughout the project lifetime (including Phase II) to reflect the evolving functionality of the application.

Task 5 provided a smooth transfer of the application from the research development team to MDOT ownership.

Tasks 7 to 10 were part of a supplemental development plan following the initial project to enhance the application's functionality and bring the application closer to release and integration with MDOT's inspection routine.

2. Literature Review

While federal guidelines for bridge inspection reporting must be met nationwide, individual states are free to meet those requirements in different ways. This has led to the use of diverse methodologies and a host of commercial solutions addressing the states' needs. The literature review for this project determined the state of the practice for bridge inspections across the country and summarized the tools currently available to facilitate the process, including devices that could be used to deploy a mobile bridge inspection application. Unfortunately, at the time of the project's literature review in 2014, none of these solutions, mobile or otherwise, were capable of handling AASHTO element-level data collection. The full state-of-the-practice report generated for this project is contained in Appendix 8.2.

In addition to evaluating current software solutions, the project team developed a survey to assess the methodologies used by bridge managers throughout the nation (Figures 2-1 and 2-2). Twenty-one responses were received from 21 states. This survey concluded that over 70 percent of the responding states used some electronic hardware in the data collection process, and over half of that hardware was laptops. Many agencies used custom software for inspection and management, including in-house software and modified or customized commercial solutions. See Appendix 8.3 for further details on the survey results.

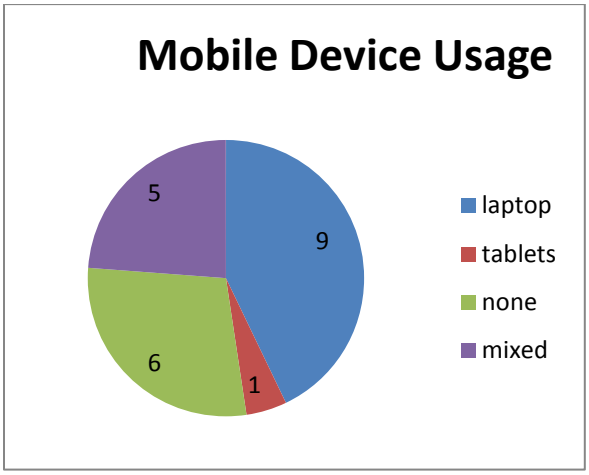


Figure 2-1: Mobile device usage of the responding agencies. The "mixed" category includes agencies that use both tablets and laptops.

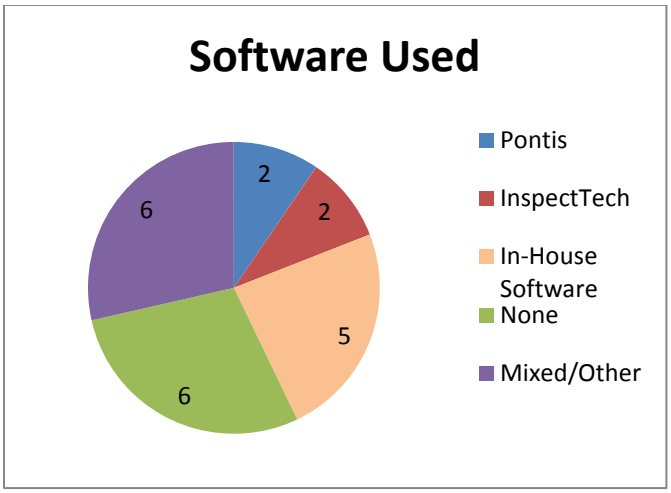


Figure 2-2: Types of inspection and management software currently being used. Many agencies use custom software. The "mixed/other" category represents modified or customized commercial solutions.

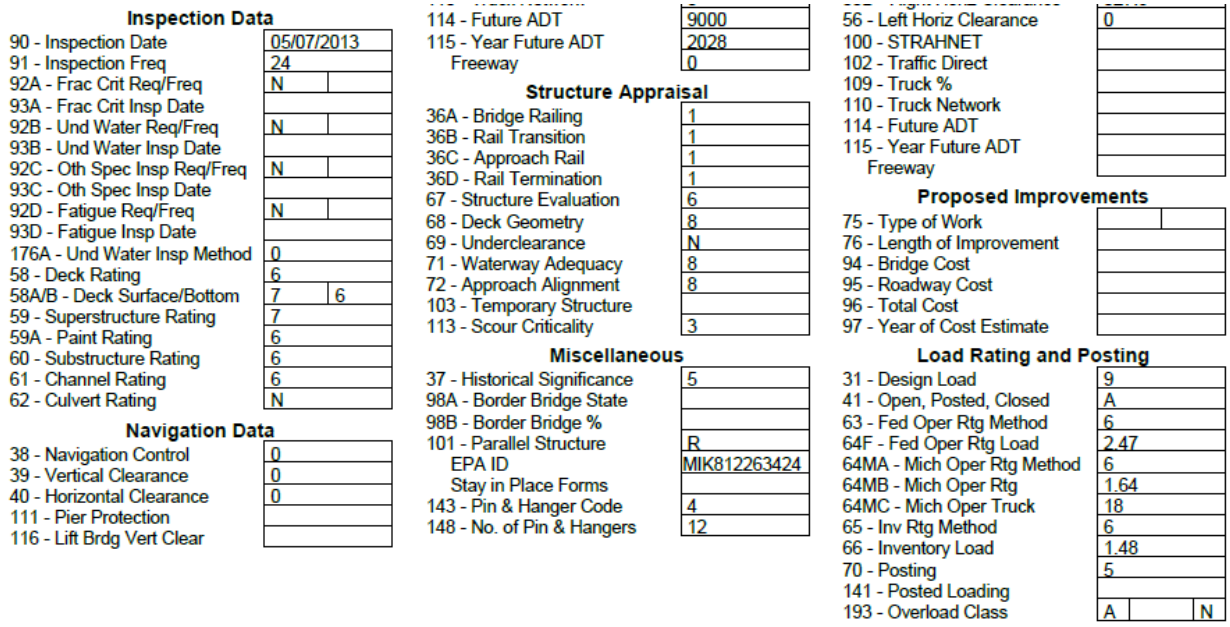
3. Review of MDOT Practices

The successful design and implementation of an application for collecting MDOT bridge inspection data hinged on understanding the current practices of MDOT bridge inspectors. By understanding the challenges and procedures inspectors deal with, the project team could develop an application with the functionality needed to help MDOT improve efficiency and accuracy. Project staff met with MDOT staff, including bridge inspectors, on several occasions to learn about and document current practices.

3.1 Inspection Forms

At the core of the inspection process are the forms that define what data must be collected to complete a bridge inspection. These forms include the NBI Safety Inspection Report, NBI CoRE Elements Report, and the Structure Inventory and Appraisal (SI&A) form.

The SI&A form (Figure 3-1) largely serves as a reference for the bridge being inspected by providing information such as component material types, dimensions, load ratings, and inspection frequency. It also contains a few fields for overall ratings of structure components such as superstructure, substructure, deck, and paint.



Inspection Data

90 - Inspection Date	05/07/2013	
91 - Inspection Freq	24	
92A - Frac Crit Req/Freq	N	
93A - Frac Crit Insp Date		
92B - Und Water Req/Freq	N	
93B - Und Water Insp Date		
92C - Oth Spec Insp Req/Freq	N	
93C - Oth Spec Insp Date		
92D - Fatigue Req/Freq	N	
93D - Fatigue Insp Date		
176A - Und Water Insp Method	0	
58 - Deck Rating	6	
58A/B - Deck Surface/Bottom	7	6
59 - Superstructure Rating	7	
59A - Paint Rating	6	
60 - Substructure Rating	6	
61 - Channel Rating	6	
62 - Culvert Rating	N	

Navigation Data

38 - Navigation Control	0
39 - Vertical Clearance	0
40 - Horizontal Clearance	0
111 - Pier Protection	
116 - Lift Brdg Vert Clear	

114 - Future ADT	9000
115 - Year Future ADT	2028
Freeway	0

Structure Appraisal

36A - Bridge Railing	1
36B - Rail Transition	1
36C - Approach Rail	1
36D - Rail Termination	1
67 - Structure Evaluation	6
68 - Deck Geometry	8
69 - Underclearance	N
71 - Waterway Adequacy	8
72 - Approach Alignment	8
103 - Temporary Structure	
113 - Scour Criticality	3

Miscellaneous

37 - Historical Significance	5
98A - Border Bridge State	
98B - Border Bridge %	
101 - Parallel Structure	R
EPA ID	MIK812263424
Stay in Place Forms	
143 - Pin & Hanger Code	4
148 - No. of Pin & Hangers	12

56 - Left Horiz Clearance	0
100 - STRAHNET	
102 - Traffic Direct	
109 - Truck %	
110 - Truck Network	
114 - Future ADT	
115 - Year Future ADT	
Freeway	

Proposed Improvements

75 - Type of Work		
76 - Length of Improvement		
94 - Bridge Cost		
95 - Roadway Cost		
96 - Total Cost		
97 - Year of Cost Estimate		

Load Rating and Posting

31 - Design Load	9	
41 - Open, Posted, Closed	A	
63 - Fed Oper Rtg Method	6	
64F - Fed Oper Rtg Load	2.47	
64MA - Mich Oper Rtg Method	6	
64MB - Mich Oper Rtg	1.64	
64MC - Mich Oper Truck	18	
65 - Inv Rtg Method	6	
66 - Inventory Load	1.48	
70 - Posting	5	
141 - Posted Loading		
193 - Overload Class	A	N

Figure 3-1: A section of the Structure Inventory and Appraisal form.

The NBI Safety Inspection Report contains the bulk of what the inspector must collect. It is organized first by overarching categories such as Deck, Superstructure, and

Substructure. Each of these categories is then broken into subcategories, such as Stringer, Paint, Section Loss, and Bearings (for Superstructures). The inspector must assign each subcategory a 0 to 9 condition rating that factors in all of the deterioration or flaws present in those components throughout the bridge. To aid the inspector's decision, a history of ratings for previous years is included, as well as past comments. When the report is completed, the combination of current and historical inspection information gives an overall picture of the progress and rate of bridge deterioration (see Figure 3-2 for an example).

SUPERSTRUCTURE

	05/09	05/11	05/13	
9. Stringer (SIA-59)	7	7	7	Painted A588 steel I beams with staggered diaphragms. Stainless steel pins. Few areas of light LOS less than 10% near beam ends, cleaned and painted. (05/13) Painted A588 steel I beams with staggered diaphragms. Stainless steel pins. Few areas of light LOS less than 10% near beam ends, cleaned and painted. (05/11) Painted A588 steel beams and diaphragms. Tight vertical cracks in backwalls. Surface coat applied to backwalls. Staggered diaphragms. Stainless steel pins. (05/09)
10. Paint (SIA-59A)	8	8	6	Painted A588 steel I beams. Minor rust on few top flanges at leaching deck cracks. (05/13) Painted A588 steel I beams. (05/11) Painted A588. (05/09)
11. Section Loss	2	2	2	Few areas of light LOS 10% or less near beam ends, cleaned and painted. (05/13) Few areas of light LOS 10% or less near beam ends, cleaned and painted. (05/11) Few areas of light LOS 10% or less near beam ends, cleaned and painted. (05/09)
12. Bearings	7	7	7	Bearings cleaned and painted. Minor rust on few abutment bearings. (05/13) Bearings cleaned and painted. (05/11) Rockers and abutment bearings have been cleaned and painted. (05/09)

Figure 3-2: A section of the NBI Safety Inspection Report. The report combines historical and current ratings and comments to fully document deterioration.

The NBI CoRe Elements Report captures AASHTO element-level information on condition state. Each component of the bridge is assigned an element type number (there are approximately 158). For a given bridge, applicable element types have a total quantity and a unit of measurement (linear, area, or both). When inspectors look at a bridge, they must quantify the units and condition states of defects for each element type for the whole bridge. The condition states are Good, Fair, Poor, and Severe. To aid in the inspection process, each element type has a table listing the possible defects that can be associated with it and descriptions of the defect for each condition state (see Table 3-1 and Figure 3-3).

Table 3-1: Condition State Table for Prestressed Concrete (from the Michigan Bridge Element Inspection Manual).

Michigan Bridge Element Inspection Manual

CS TABLE 2 – PRESTRESSED CONCRETE

Defects	Condition State 1	Condition State 2	Condition State 3	Condition State 4
	GOOD	FAIR	POOR	SEVERE
Spalls/ Delaminations/ Patch Areas (1080)	None.	Delaminated. Spall 1 in. or less deep or less than 6 in. diameter. Patched area is sound.	Spall greater than 1 in. deep or greater than 6 in. diameter. Patched area is unsound or showing distress. Does not warrant structural review.	The condition warrants a structural review to determine the effect on strength or serviceability of the element or bridge; OR a structural review has been completed and the defects impact strength or serviceability of the element or bridge.
Exposed Rebar (1090)	None.	Present without section loss.	Present with section loss that does not warrant structural review.	
Exposed Prestressing (1100)	None.	Present without section loss.	Present with section loss that does not warrant structural review.	
Cracking [1] - PSC (1110)	Insignificant cracks or moderate-width cracks that have been sealed.	Unsealed moderate-width cracks or unsealed moderate pattern (map) cracking.	Wide cracks or heavy pattern (map) cracking.	
Efflorescence / Rust Staining (1120)	None.	Surface white without build-up or leaching without rust staining.	Heavy build-up with rust staining.	
Settlement - Substructure (4000)	None.	Exists within tolerable limits or arrested with effective actions taken to mitigate.	Exceeds tolerable limits but does not warrant structural review.	
Scour - Substructure (6000)	None.	Exists within tolerable limits or arrested with effective countermeasures.	Exceeds tolerable limits but is less than the limits determined by scour evaluation, and does not warrant structural review.	
Damage (7000)	Not applicable.	The element has minor damage caused by vehicular or vessel impact.	The element has moderate damage caused by vehicular or vessel impact.	The element has severe damage caused by vehicular or vessel impact.

AASHTO ELEMENTS (English Units)

Element Number	Element Name	Total Quantity	Unit	Good CS1	Fair CS2	Poor CS3	Severe CS4
Decks/Slabs							
803	Conc Deck - Coated Bars	8262	sq.ft	0	8226	36	0
				0%	100%	0%	0%
815	Rigid Overlay	8262	sq.ft	8240	0	22	0
				100%	0%	0%	0%
811	Conc Deck - Btm Surface	8262	sq.ft	0	8226	36	0
				0%	100%	0%	0%
812	Reinf Conc Fascia	360	ft	270	90	0	0
				75%	25%	0%	0%

Figure 3-3: A section of the NBI CoRe Elements Report. Deterioration is classified by element type, quantity, and condition state.

3.2 Inspection Procedures

While the inspection forms determine which data need to be collected, of equal importance is how those data are collected. There are no rigid rules that define how a bridge inspector should go about collecting the necessary information to fill out the forms, so there is a natural variability in how individuals and organizations will handle the process. However, guidelines and the physical nature of the task ensure that there

should be sufficient overlap in practices to define a generalized procedure. Capturing this process was essential to the design of the inspection application since it directly reflects the needs of the application's users, who are in turn trying to meet the needs of bridge managers.

MTRI staff began by meeting with Amy Trahey, president of Great Lakes Engineering Group, LLC, and a former MDOT bridge inspector. Trahey provided a virtual walk-through of a bridge inspection. (Figure 3-4 represents the inspection process as Trahey described it.) The process is nonlinear—inspectors do not simply go down the list of items on the form and evaluate each one. This is largely a matter of efficiency. For example, evaluating the railings on a bridge requires walking both sides of the bridge, and in doing so the inspector will pass many other components. Trahey also provided a listing of tools and materials an inspector would require during the inspection, such as manuals, ratings guides, cameras, previous inspection reports, and pencils. Another important consideration is that inspections are routinely performed by two inspectors. Typically, one inspector will proceed with the inspection itself, filling out the forms, while the other inspector will photograph bridge deterioration and areas of concern.

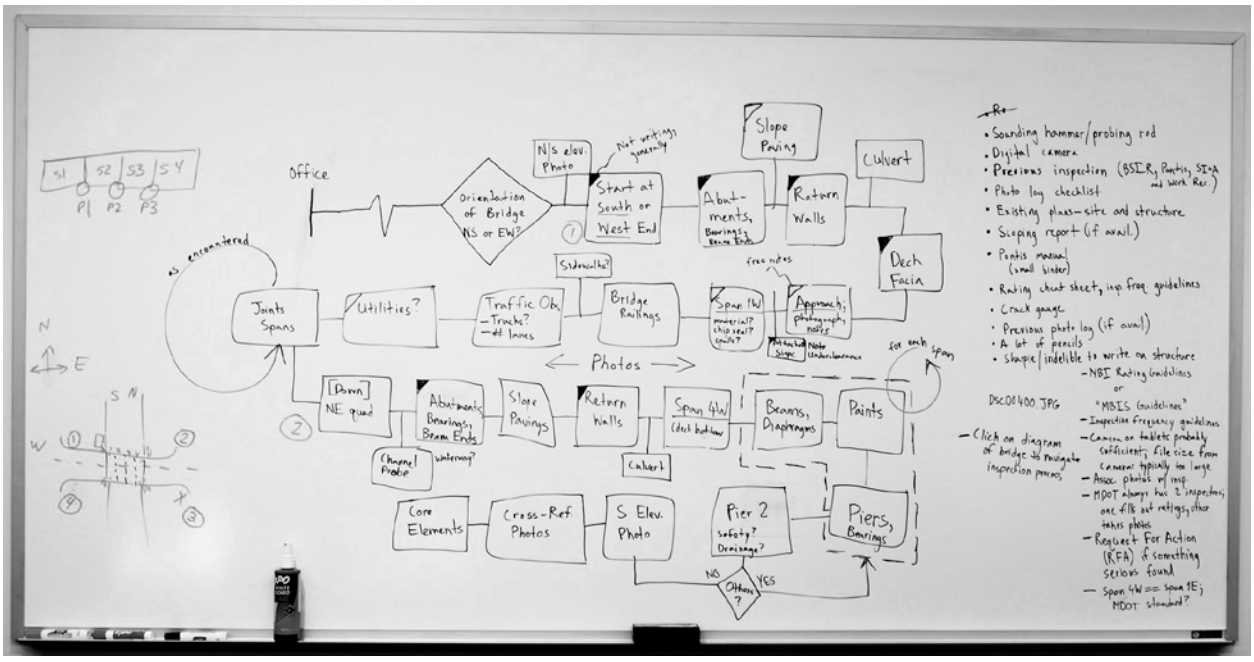


Figure 3-4: Inspection flow diagram and tool/material listing.

To supplement their understanding of the bridge inspection process, MTRI staff accompanied MDOT inspectors Janiene DeVinney and Lindsey Renner for a mock inspection of the Curtis Road Bridge over M-14 northeast of Ann Arbor (Figure 3-5). This served as a very useful demonstration of the workflow process outlined by Amy Trahey and gave the application developers a chance to see firsthand what a bridge inspector deals with. Of particular note, inspectors write a great deal of information on scratch paper or in the margins of the paper forms, since the generalized ratings are formed from a comprehensive view of the bridge while the inspection process itself must proceed piecemeal. The group also discussed office practices, because inspectors must transfer information from paper forms to MDOT's database after the inspection is completed. They also discussed task assignment authentication/security practices since inspectors are responsible for the quality of their inspections.

Figure 3-5: MTRI staff observe an MDOT inspector examining joint condition.

4. Application Design and Requirements

Using the information gained from the literature review and from observing MDOT's current practices, the project team formulated requirements and design parameters for the application. The software requirements specification is designed to encapsulate what the application will and will not do. Its primary purpose is to ensure clear communication between the client and the developer concerning the application's functionality. It is not meant to be a rigid constraint; it can be revised as needed given further clear communication between parties. The original document can be seen in Appendix 8.4.

4.1 Requirements

The primary requirement of the application was that it collect and aggregate AASHTO element-level inspection data. It was MDOT's desire that this would involve a 2D or 3D interface (preferably 3D) depicting the bridge elements, which could then be tagged with relevant information such as element type, defect type, condition state, and defect quantity. Such an application would have the advantage of not only capturing element-level data, but also capturing the location and size of individual defects, which opens up new opportunities for monitoring deterioration. This primary requirement was of keen importance since at the time of the literature review, no software or procedure existed to efficiently gather element-level data.

Of secondary importance was the collection of comments and photographs concerning the defects, preferably utilizing a device's built-in camera. This information, along with the element-level data, is vital to maintaining a historical record of the bridge's condition so appropriate deterioration monitoring can occur and response decisions can be made. Additionally, it was desired that the application automatically compile the recorded information into the broader categories used in the various forms, thereby eliminating the need for inspectors to keep track of it themselves. Compiling individual defects also would dovetail well with the inspectors' practice of recording information as it is observed.

MDOT was also interested in viewing historical information during the inspection process. This feature would be similar to the previous ratings available on the NBI Safety Inspection Report, which provide additional input for the inspector to consider. Finally, since the application would already be in a digital format, it should be designed to enable communication with the MDOT BMS database to store finalized inspection data and photographs, eliminating the need for inspectors to do so manually.

4.2 Design Considerations

In developing the application, several important design decisions had to be considered. The first of these was device compatibility, since a wide range of portable electronic devices are available, including laptops, tablets, and smartphones. MDOT was primarily interested in tablet devices as a good compromise between the bulk and power of a laptop and the portability but small screen size of a smartphone. However, the tablet operating system (OS) universe is quite diverse, and different options are often incompatible with one another: Any application developed natively for one device would need to be completely reprogrammed to work on another OS. Web applications are promising in that they run via browsers instead of natively, but they require an active Internet connection. This may not be available in rural areas, rendering the application useless. Additionally, the desire for either 2D or 3D interaction is not well-suited to a Web application, primarily for performance reasons. Fortunately, MTRI was able to identify an alternative development strategy that sidesteps these challenges: software packages used to design video games for multiple mobile platforms.

Game design software, referred to as game engines, are software packages used by game developers to create interactive applications. They can be either 2D or 3D, and many of them promise cross-platform compatibility. With such packages, the task of device interoperability falls to the engine creators rather than the individual developers. The MDOT application is not a game, but it does share many common elements with video games, such as a need for 2D/3D rendering, geometry modeling, touch-based interaction and Web access. MTRI investigated a variety of available engine platforms to select one as the foundation for the MDOT bridge inspection application. From the large pool of available platforms, MTRI narrowed the list to three for final consideration, detailed below in Table 4-1.

Table 4-1: Game Engine Comparison.

Library	License/Cost	Pros	Cons
OSG	Based on Lesser General Public License (LGPL), a free software license	Free, low-level access, open-source code	Small community, poor documentation/support, low cross-platform compatibility (must develop natively)
Unity	$3,000 per developer	Very large community, good support, game industry standard, feature-rich, great performance	Expensive, must purchase licenses per developer, must purchase per additional platform supported, closed-source code
Unreal Engine 4	Initially $19/month for MTRI (unlimited seats, and now free), plus 5% of revenue if selling on market under standard license	Cheaper than Unity, large community, feature-rich, cutting-edge development, source code available	Early in product life cycle (software bugs, low support initially for some features), 5% of revenue if selling on market

Based on the low cost, list of features and promise of cross-platform support, MTRI chose to proceed with application development using Unreal Engine 4 by Epic Games. While being on the cutting edge of development is always a risk, Epic has a long history of successful development (Unreal Engine 3 is widely used even today). Additionally, Unreal Engine 4 subscribers are granted access to the source code of the engine, a huge advantage in shaping the application to MDOT's needs and ensuring that MDOT and MTRI will always have access to the platform for future development. As an added bonus, Epic entirely dropped the monthly subscription fee in March 2015, so MTRI and MDOT were able to receive software updates at no charge during the remainder of the development period.

5. Server Implementation

Since previous 3D models of the state's bridges were not consistently available, a model had to be created from scratch. Given the large amounts of descriptive information within MDOT's Bridge Management System database, MTRI decided to build a model utilizing all of the relevant data. from the database This way, any bridge being inspected could be viewed with a sufficiently representative model. The data were retrieved from the database, missing information was derived from the data collected, and then a representative model was created as an XML file (See Figure 5-1). When requested, the XML file is then sent to the client application to render the 3D model.

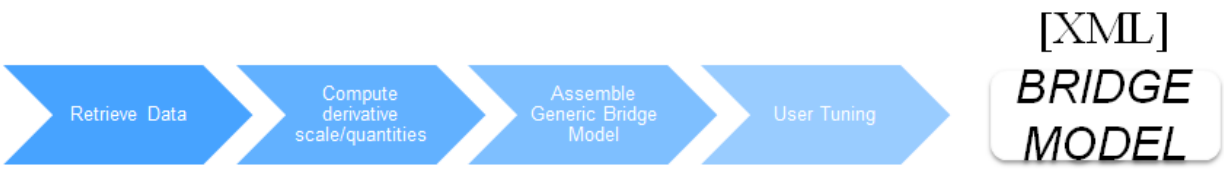


Figure 5-1: Bridge model generation.

5.1 Review of Bridge Fundamentals

To gain a better understanding of bridges, the MTRI team met with Tess Ahlborn, Co-PI and Michigan Technological University Professor of Civil and Environmental Engineering. Ahlborn gave a two-hour lecture on basic bridge fundamentals and addressed any of the staff's questions or misunderstandings about bridges. During the lecture, Ahlborn covered how a generic bridge works, explaining the function of the deck, superstructure, and substructure. The lecture also covered more specific bridge parts (such as pin and hanger assemblies, bearings, diaphragms, and girders) to provide further details about the basic components of a bridge. Ahlborn concluded the lecture by explaining all of the bridge elements that composed the Curtis Road Bridge over M-14 near Ann Arbor, which MTRI has been using as a test bridge for development (since the time MTRI staff visited it for the mock inspection). This in-depth explanation of bridges was instrumental in the project's development, as it provided further insight into how bridges work and fit together, allowing the programmers to better understand the process of making generic 3D bridge models.

5.2 BMS Database

The first step in building the 3D model was the retrieval of data from MDOT's BMS database. The database is composed of 16 tables. These tables were not intended to be used to generate 3D models, but they contain a wealth of information including bridge dimensions, bridge measurements, bridge form data and AASHTO element-level data. After copying the database onto MTRI's local server for development and testing, MTRI added one additional table to the database that would store all of the information needed to create a proportionally accurate representation of the bridge. This Bridge Model table draws from almost all of the other tables within the database and incorporates several new fields that MTRI, using generic assumptions about bridge construction, derived from the information in the database. To simplify the XML generation process, the application only pulls data from this new Bridge Model table. The Bridge Model table is very large, simplifies the process of exporting database information into an XML format, it also means that individual bridges can be modified without making changes to the rest of the BMS database.

Additionally, the BMS database contains a wealth of ancillary information such as sidewalk dimensions, traffic flow information and presence of water beneath the bridge The application uses some of this information to collect the most recent element and NBI report information, though there is other information that has not been utilized yet due to other tasks being prioritized to improve the functionality of the application first. However, this information lends itself to future improvements of the application that could make the model even more realistic.

5.3 Computing a Generic Bridge Model

After the Bridge Model table is created, MTRI utilizes Django, an open-source Python Web framework for managing websites while incorporating large amounts of data from databases (https://www.djangoproject.com/). This server application will output the requested XML file for the desired bridge when contacted by the client application. To generate the XML file, the server will query the appropriate information from the Bridge Model table, derive necessary quantities from the queried data, convert all variables to the appropriate units, and generate a list of member components (See Figure 5-2). The data needed for the NBI report information is shown in Figure 5-3. The client connects to the server using HTTP over a Wi-Fi or cellular connection. This Internet connection will be necessary for the application to load the appropriate Bridge Model XML file, but after the initial download of the XML file, no further Internet connection is necessary as the file can be stored on the tablet device.

The initial generation of the Bridge Model utilizes a set of assumptions to create values for variables that cannot be derived from the database, such as placement of bearings per pier, number of beams, and joint locations. These derived quantities should work for the majority of bridges, and all the necessary information to render the 3D model will be within the XML file. If these assumptions result in an erroneous model, administrative users can tune them to improve model fidelity (discussed in section 5.5).

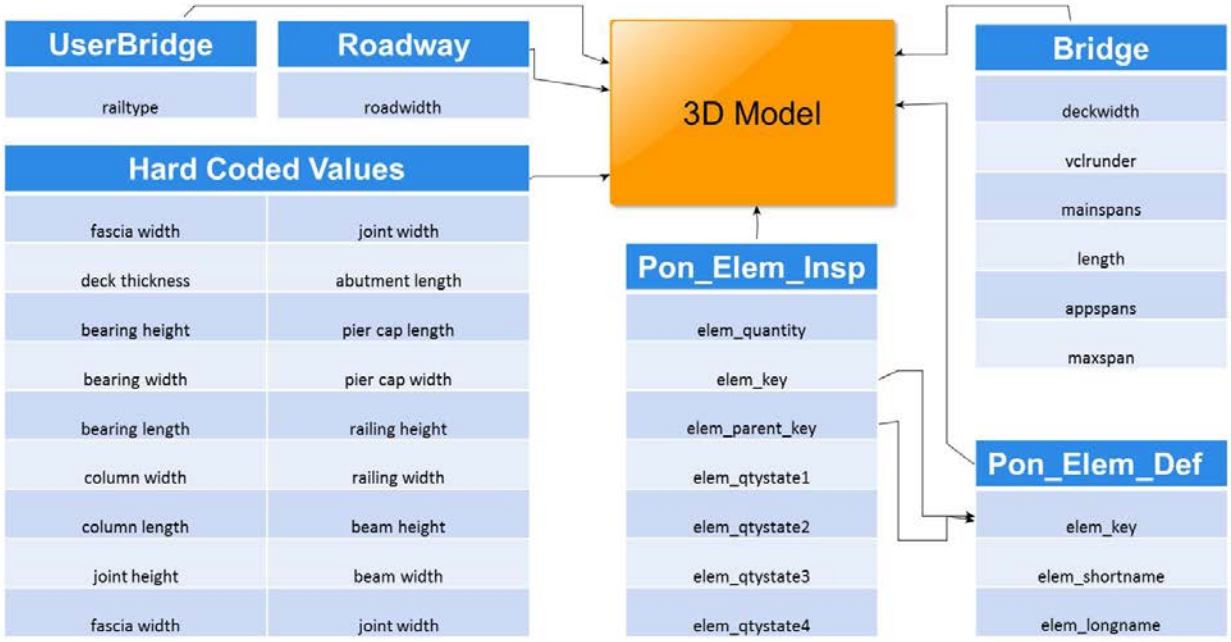


Figure 5-2: Flowchart of the back-end obtaining all of the data necessary to create the 3D model.

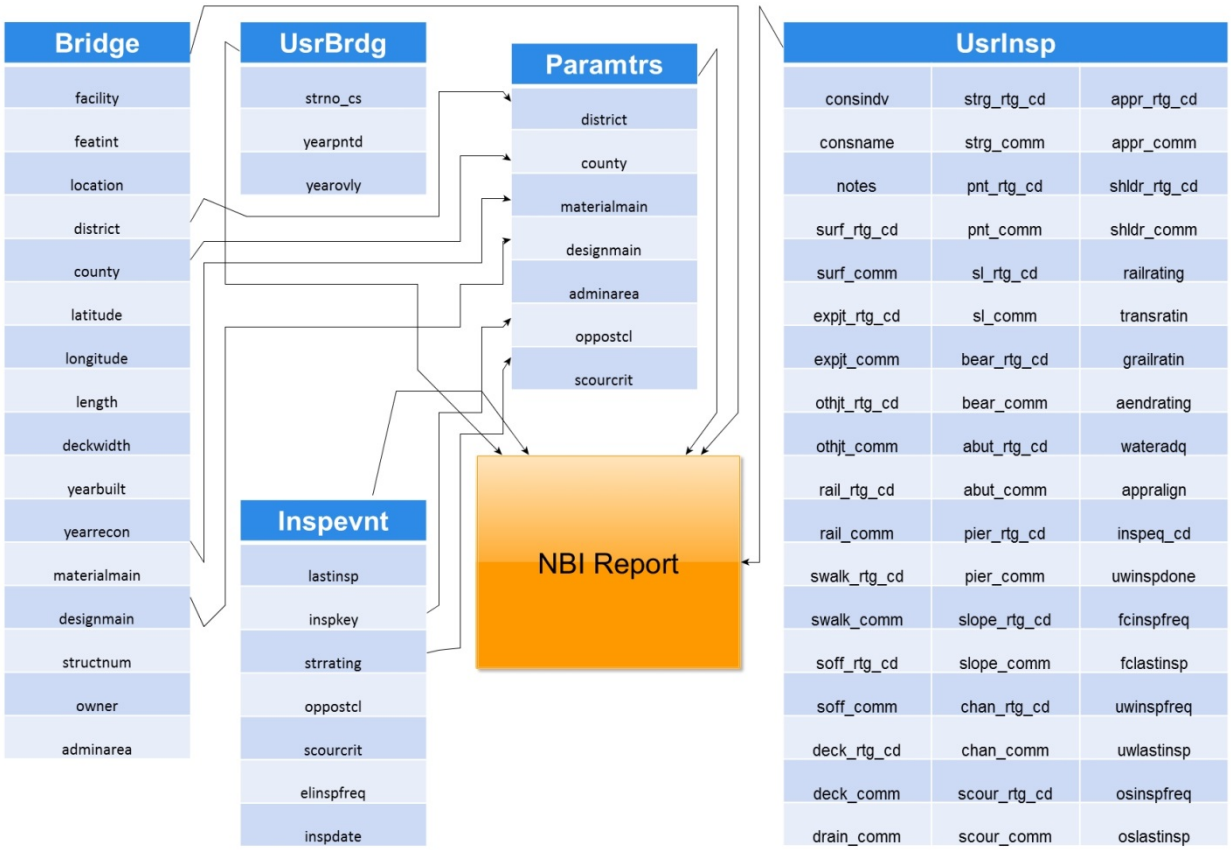


Figure 5-3: Flowchart of BMS data integrated into the NBI Safety Inspection Report.

5.4 XML File Structure

The entire XML file is arranged into six categories: basic bridge information, deck, superstructure, substructure, bearings, and culvert. Other than basic bridge information, all of the categories are created using AASHTO element-level data from the BMS database, which gives very specific details about all of the bridge parts that compose that bridge. The individual pieces of the bridge that will be rendered as parts of the 3D model are represented by the term Member in the XML file.

Each Member contains data for the Role, Type, Name, Length, Width, Height, X-coordinate, Y-coordinate, Z-coordinate, and the AASHTO Element Number associated with that Member. The Role is the category an individual member falls into, the Type is the exact name specified by the AASHTO element-level data within the database, and the Name is the identifier associated with the standard bridge inspection labeling schemes for elements such as 2 South or 1 West 2 South. The labeling scheme changes per element, and also depends on the bridge orientation, such as whether the bridge runs north and south or east and west. Each Member is associated with one

label, so if the deck bottom surface is labeled as 1 South 3 West, that will be an individual bridge piece that will be rendered separately from 1 South 2 West. When rendered, the bridge parts will appear seamless, as if they were one bridge part, but they actually are multiple pieces that make up the entire deck bottom surface. The Length, Width, and Height are all values derived from the database to render a proportionally accurate representation of the bridge. The only information in the database relevant to member height is the vertical clearance of the bridge. All of the element Heights below the substructure (pier, pier cap, and abutments) are inferred, using fixed height for most elements and extending the pier and abutment heights to cover the remaining distance. Other dimensions are also inferred if they are not found in the database. The X-, Y-, and Z-coordinates are based on the Length, Width, and Height of the individual element as well as its location relative to the other components to get an exact centeral location for that element. The AASHTO Element Number is provided so the client application can determine the context of the member. (See Figure 5-4 for examples of the above data contained within the bridge XML file.)

Figure 5-4: Example of a bridge member variable in XML format.

Using Member variables to represent individual bridge pieces is critical since the unavailability in the database of some of the required information imposes certain limitations on creating a 3D model from the database. The Member variables are self-defining (they do not rely on relative information from any other part of the XML) so the client is more flexible for future model improvements. This will be helpful in the case of more unusual bridges such as those that have a varying number of beams per span, or those where the bearing placement per pier is abnormal.

5.5 User Tuning

As noted, MTRI made some generic assumptions in calculations for key variables used to render the 3D bridge model. To address the issue, an administrative website (separate from the client application) was developed through Django that enables the inspector or bridge engineer to verify and/or modify these assumptions to create a more accurate model. For the generic concrete overpass-style bridge, the calculations should be reasonably accurate. However, there are several outliers where key pieces of information about how the bridge is composed—such as numbers of beams per span and placement of bearings per pier—are abnormal. These bridges would be modeled incorrectly and therefore the inspector could not record defect data accurately. The website's administration tool allows bridge inspectors to fix any errors in the model (usually ahead of the inspection) to create a better replica of the bridge, and allows them to make any necessary changes to the data as they see fit. Within the administration tool, the information is divided into eight categories: Assumptions, General Bridge Information, Deck, Superstructure, Substructure, Bearings, Bearing Placement, and Culvert. The most important information that the bridge inspector will need to review are the Assumptions and Bearing Placement sections. These are the two areas where data are not present in MDOT's database but are derived from calculations and assumptions. In the future, more fields and categories may be incorporated in the administrative tools to make a more accurate model.

5.6 Other Services

The application requests different URLs for past NBI CoRe Element and NBI Inspection reports. Each URL sends back an XML file with the most recent report information for the bridge that was selected. Additionally, the server can accept newly collected NBI data to store to the database. When the user finishes an inspection, he or she can press the "Push" button, and the front-end application will send all of the element-level defect and report information in an XML file to the back-end. The back-end will then appropriately store the data in the correct variables to use in the future.

5.7 Limitations

As models do not exist for every bridge potentially needing inspection, MTRI needed to use information from MDOT's BMS database to create each 3D model. The current application is optimized to accurately model generic overpass-style bridges but will inaccurately model bridges that are irregular. This limitation is ameliorated using the Django administration site, which can correct many simple errors in the models. Another limitation is that the application does not yet handle "exotic" bridges such as cable

bridges, culvert bridges, or truss bridges. These bridges will not cause the application to crash or behave improperly, but they will not be rendered properly in the current version. This limitation could be addressed through a future enhancement-focused project phase. Such bridges are not particularly common, and modeling them would be time-consuming; time was instead spent on higher-priority tasks during the project's first two phases. A final key limitation is that bridges that are not monitored by MDOT are particularly challenging to model properly, as no AASHTO element-level data have been captured for them. The application's model for these bridges would be limited by a lack of structural information and would be unlikely to represent the bridge accurately.

6. Client Implementation

Implementation of the client application, whose name has been changed from MDOT 3D Wireless Bridge Inspection System/3DWBIS to 3D Bridge App, is the primary product of this research. It is built on Epic Games' Unreal Engine 4 (UE4) and can work both in Windows desktop environments and on Android mobile devices such as tablets or smartphones. Taking advantage of UE4's rendering capabilities, the 3D Bridge App parses XML files delivered by the server and creates 3D representations of the bridge being inspected. Then, inspectors can dynamically tag the surface of the bridge with defects.

6.1 Coding

UE4 is primarily based on the C++ programming language using the Microsoft Visual Studio development environment. The engine relies heavily on macro functionality, adding its own particular flair of coding as well as an extensive application program interface (API) for interfacing with the engine. Any software development projects utilizing the engine include an Unreal-specific build program that automatically sets up the Visual Studio environment and pre-compiles specialized header files that prepare the macro interface. There is also a UE4 plug-in for Visual Studio that allows tighter integration with UE4 projects. The bulk of the new application is coded in this environment, but there are several important exceptions.

The first is UE4's Blueprint language (see Figure 6-1). This is essentially a visual coding language defined within the UE4 editor that allows for high-level interaction with game mechanics. This higher abstraction level, as compared to coding in C++, benefits certain tasks such as user interaction with objects and camera navigation. Functions, operators, events, and variables exist in Blueprint as blocks on the screen with inputs and outputs as tie-in points on the blocks. Different code blocks are then strung together, linking like variables across blocks as well as tying the execution flows together to form the program.

The second exception is native device coding. This is done within the UE4 source code rather than project code and is specific to the operating system targeted. In this case, use of the built-in cameras available on mobile devices must be developed separately for iOS and Android. For example, Android's native language is Java, so the camera functionality exists as a Java plug-in for UE4. While it is inconvenient to have to reproduce this functionality for each supported operating system, the extra effort

needed is rather small compared to developing the entire application for multiple systems.

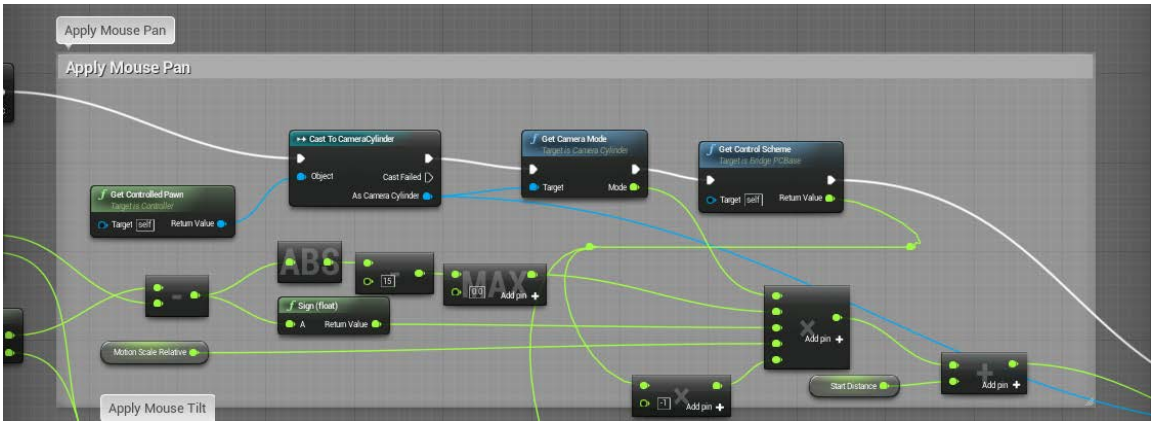


Figure 6-1: UE4's Blueprint coding language as used to implement the Client Application's user interface.

6.2 Loading Bridge XML files

The first step in using the application is to load the XML model for the bridge being inspected. The sidebar menu of the application includes a Load Bridge button, which polls the server for a list of bridge models available (see Figure 6-2). The user then selects the bridge of interest and can either load it or download it. The Download option copies the XML to the device's internal storage for offline use; such bridges will have their menu item display in green instead of blue. The Load option will use the downloaded XML if available or, if not, will ask the server for the XML instead. While in offline mode, only bridges with downloaded XML files will appear in the list. Once the server has responded, the application will parse the XML and generate a list of all the bridge member elements. Each member element is assigned appropriately scaled and positioned geometry within the application world, effectively constructing the bridge from its individual components. Each of these elements retains context-sensitive information about itself, such as the member's name,

Figure 6-2: Load Bridge menu.

that is displayed when the user interacts with the element.

6.3 Navigation

Navigation in a full 3D environment can be daunting since it involves motion with six degrees of freedom (6-DoF), three-axis translation and three-axis rotation. This problem is exaggerated in touch-based environments, which are limited to a 2D plane. Multi-touch, gestures where more than one finger is used, can help, but overreliance makes the user experience unintuitive. For the client application, multi-touch is limited to the familiar pinching gesture often used for zoom. Since this limits the application to 3-DoF input for a 6-DoF environment, some constraint on allowable motion is needed. To cope with this problem, two viewing methods have been implemented to allow for natural viewing of the bridge geometry while keeping user interaction simple and intuitive.

The first viewing method has been dubbed Camera Cylinder (see Figure 6-3). Essentially, the camera, or view angle of the user, is constrained to a cylindrical orbit along the bridge. Swiping left or right with mouse or touch interaction pans the view, while swiping vertically changes the orbit angle of the camera around the bridge. Since a full 360-degree orbit of the bridge would result in the camera viewing the bridge upside down, or, if the camera were flipped, would cause a control inversion that would be frustrating and confusing for users, viewing is limited to 180-degree arcs. However, the compass widget in the upper right of the application heads-up display (HUD) can be clicked to switch to the opposite arc. The final pinching gesture allows the camera to zoom in on a target area of interest to the inspector. The Camera Cylinder viewing mode is the default and allows the inspector to intuitively navigate most of the bridge, while the zoom option makes it easy to get close-in views.

The second viewing method is called Camera Rail (see Figure 6-4) and was created in response to feedback from MDOT bridge inspectors during a demonstration of the application. In this view, the camera is constrained to a box volume centered on the bridge. Vertical and horizontal swipes pan the camera in their respective directions, while the pinch gesture translates the camera forward or backward along the bridge. The compass widget switches the camera view direction 180 degrees. This viewing method is convenient for reproducing some of the viewing angles inspectors use in the field, such as looking at an abutment head-on.

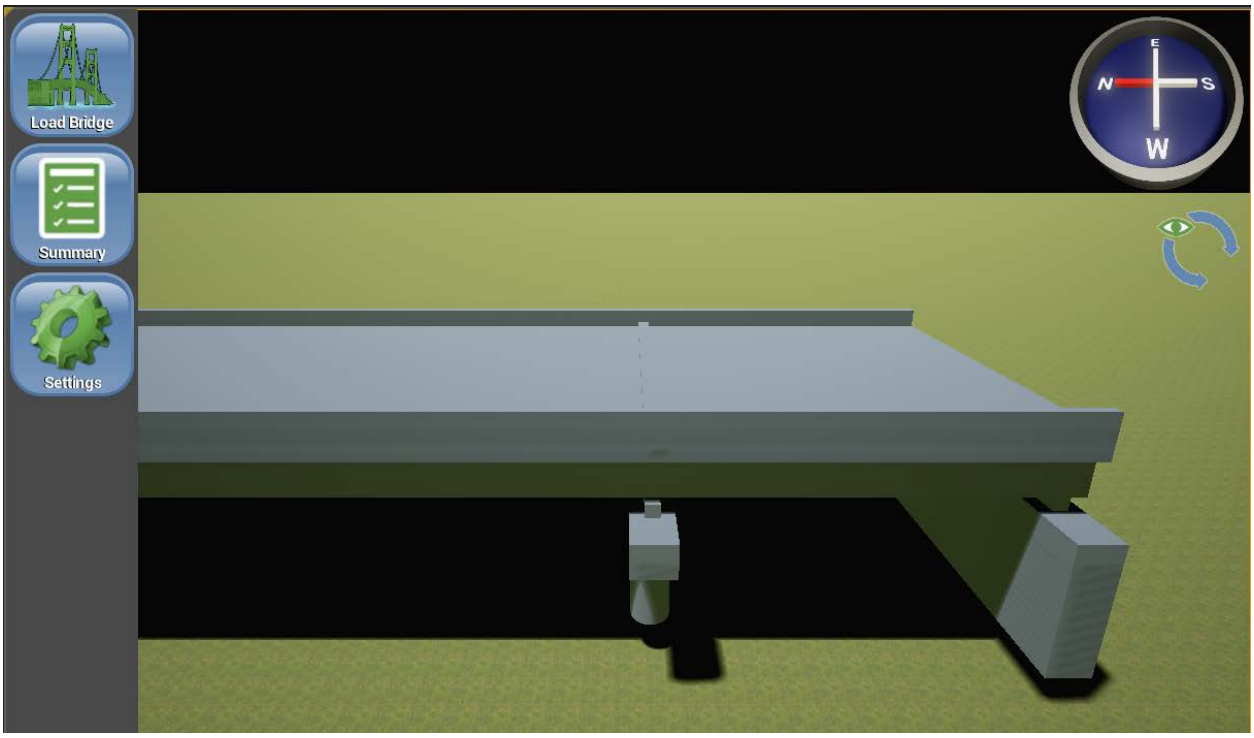


Figure 6-3: Camera Cylinder view orbits around and along the bridge.

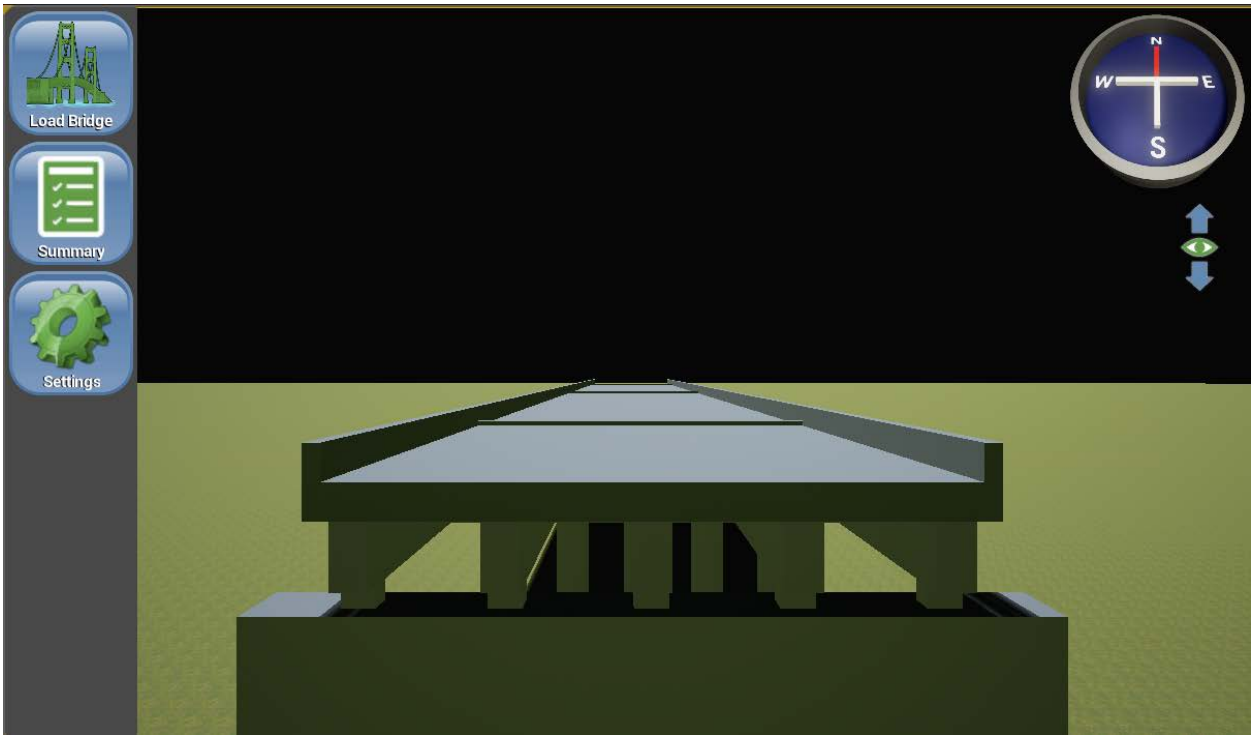


Figure 6-4: Camera Rail view allows head-on inspection of the bridge.

6.4 Element-Level Defects

The primary feature of the application is its ability to tag the bridge model with defects. After navigating to the bridge location being examined, the inspector can tap on the bridge surface to place a defect marker (see Figure 6-5). A menu pops up that allows the inspector to select an element type from a shortlist of elements most likely applicable based on context-sensitive information from the bridge XML file. A check box exists to disable the filtering and present the full list of elements should the inspector not find the one being examined. Once an element type has been selected, the inspector chooses the defect type. The defect type drop-down menu is populated only with types applicable to the selected element type. The inspector can also choose the condition state of the defect (the default state is Fair) and enter the unit quantity for the defect. The defect description is automatically updated according to the combination of defect type and condition state, allowing the inspector to quickly confirm that the option selected matches MDOT guidelines. The Add Picture button allows inspectors to attach an existing photograph or take a new one; clicking on an attached photo will display a full-screen image of the photo. At the bottom is a comment box where inspectors can add any additional information they wish to record.

Figure 6-5: Defect pop-up menu. Title and element shortlist are context-sensitive according to the bridge location touched.

Also part of the defect pop-up menu is the option to switch to the Edit Marker mode; this view removes the HUD and pop-up overlays to offer an unrestricted view of the bridge (see Figure 6-6). A minimal interface at the bottom presents the user with options to resize (according to unit quantity), relocate, and rotate the defect marker. The user also can manipulate the aspect ratio of the marker, allowing for an infinite variety of

rectangular markers. Setting the aspect ratio to 0 will convert the marker to circular from rectangular.

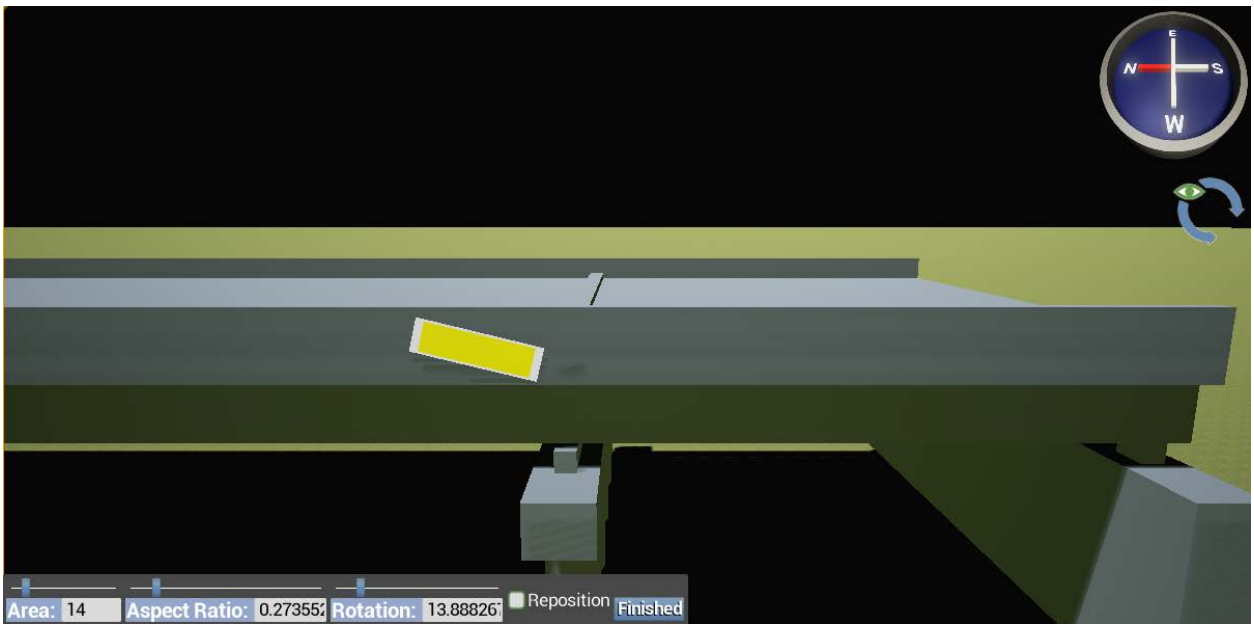


Figure 6-6: Marker Editor offers an unrestricted view so the inspector can position and manipulate the defect.

A button on the bottom left of the defect menu links the defect to the NBI rating entry menu, through which the inspector can pull up the NBI section most relevant to the current defect (see Section 6.6).

6.5 Bridge Review

The Bridge Review menu offers several choices for reviewing the data collected during the inspection process. The Element Review mimics the format available on the MiBRIDGE website (see Figure 6-7), listing the percentage of condition states for each bridge element but also providing a breakdown of the individual defects contributing to that score. Totals are updated as the inspection continues, relieving inspectors of having to perform the calculations themselves.

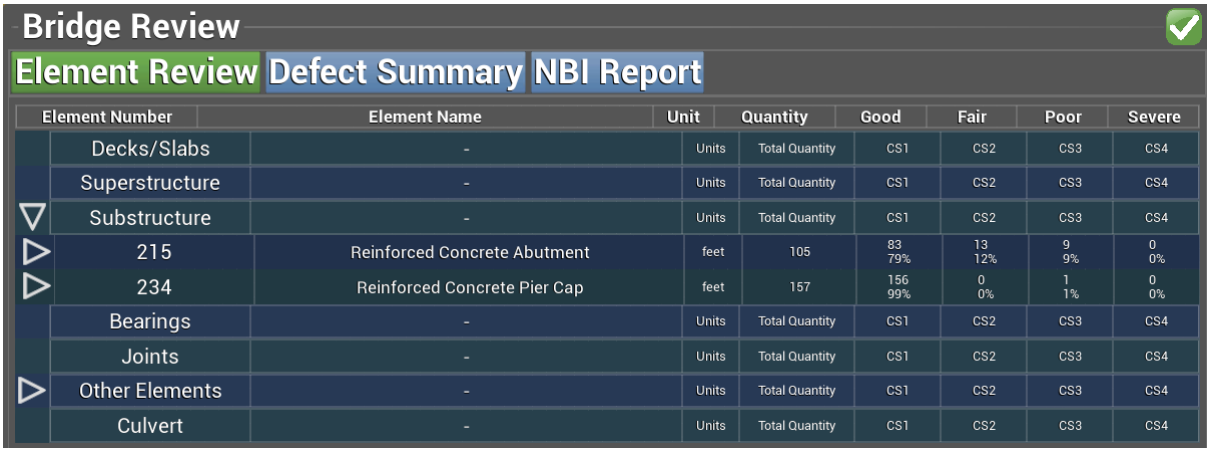


Figure 6-7: Element Review mimics MiBRIDGE format.

The Defect Summary menu offers an alternative breakdown of the defects (see Figure 6-8). The top level of the drill-down shows the condition rating, while subsequent levels show the category, then element type, defect type, and finally individual defects. Quantities are automatically summed for each level of the drill-down, and comment boxes and icons for photographs are available.

Figure 6-8: Defect Summary drill-down to individual element-level defects.

6.6 NBI Safety Inspection Report

As part of the supplemental Phase II work plan, the project team added the capability of entering and reviewing NBI safety inspection report information to create a more integrated solution to bridge inspections. The full NBI safety inspection report

information can be accessed through the Bridge Review menu. The display mimics the paper form but includes a few appropriate upgrades for a digital format (see Figure 6-9).

The top section of the display is identical to the paper form, listing bridge information such as location, dimensions, materials, last inspection date, and current inspector, and providing an entry box for general inspection notes. Below that, the NBI rating entries are found, divided into structural categories (such as deck, superstructure, etc.) The categories are collapsible, facilitating navigation between sections on limited screen space. The final section, Miscellaneous, contains data entry fields for all applicable items including guardrail ratings, water adequacy, approach alignment, high-load hits, and underwater inspection method.

Figure 6-9: Digital NBI Report form.

General NBI sections pertaining to the bridge structure and approach all follow the same entry format and can be accessed from the full report form or by clicking the NBI Ratings shortcut button in any bridge defect menu (see Figure 6-10). The shortcut menu option will infer which NBI category the inspector is interested in reviewing based on the current defect context, but any category can be selected from the drop-down menu. This context-sensitive shortcut system allows inspectors to move quickly between entering detailed element-level information and entering information in the broad NBI categories, facilitating an enter-as-you-go approach.

At the top of the shortcut form, the previous three ratings are listed along with a button to enter in the current rating. Below that, the previous three comments are listed, each one accompanied by a button that will copy that comment into the current comment box

at the bottom of the entry form. Once copied, the comments can be edited, freeing the inspector from having to entirely rewrite the comments each time. When selecting numeric NBI ratings, the inspector sees a ratings wheel displayed which depicts a pie graphic with the ratings N and 0-9 (see Figure 6-11). This format allows the inspector to quickly select the desired rating on a mobile device with or without the use of a stylus. The N rating was included at MDOT's recommendation to allow for a "not applicable" option when a bridge does not contain that particular component.

Figure 6-10: NBI Rating shortcut entry form, accessible from any bridge defect menu.

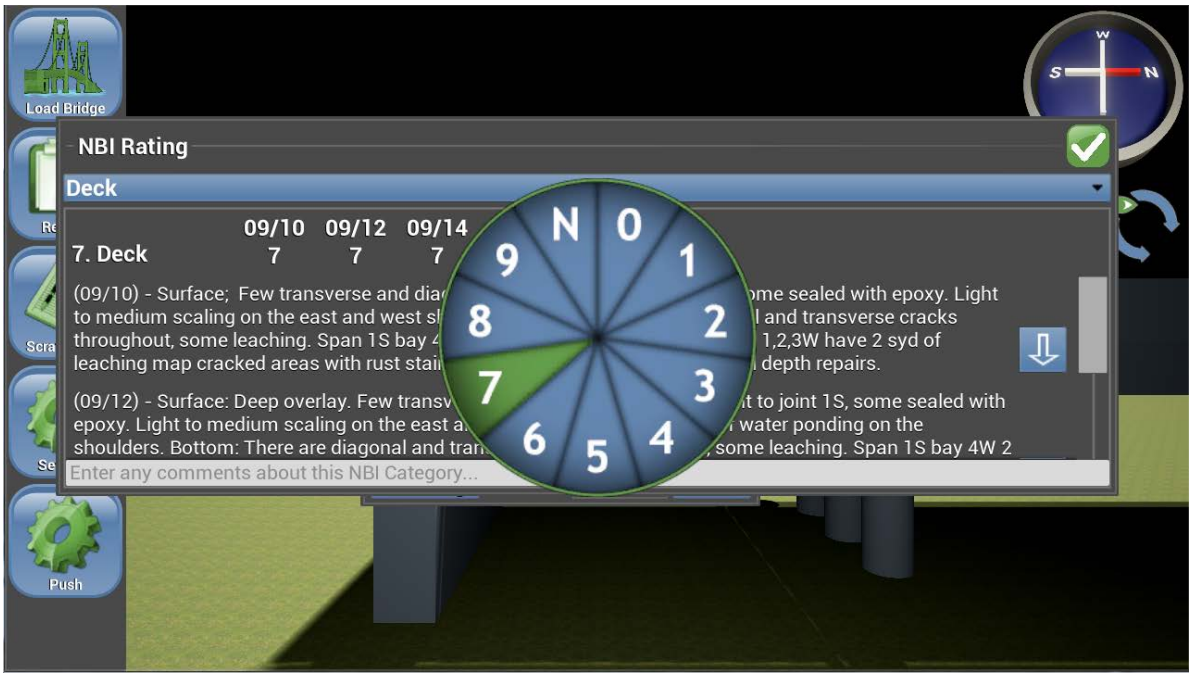


Figure 6-11: The ratings wheel is a touch-friendly interface for quickly selecting NBI ratings.

6.7 Scratch Pad

At the request of inspectors following field demonstration reviews, a scratch pad interface was implemented. The interface consists of a white space upon which the inspector is free to draw or write something of interest (see Figure 6-12). The interface includes several sizes of brushes for drawing and erasing as well as a Clear Screen option. Writing is best done with a stylus since fingers are too large for small text, but drawing can be done easily with either tool. Currently, the scratch pad's content is not recorded within the inspection and is purely for the inspector's personal use. Future development work could include extending the scratch pad tool set to create overlay drawings for pictures associated with bridge defects, allowing inspectors to highlight problem spots or write comments. Such photo overlays could be included with the photo data uploaded to the server to facilitate management review of inspection data.

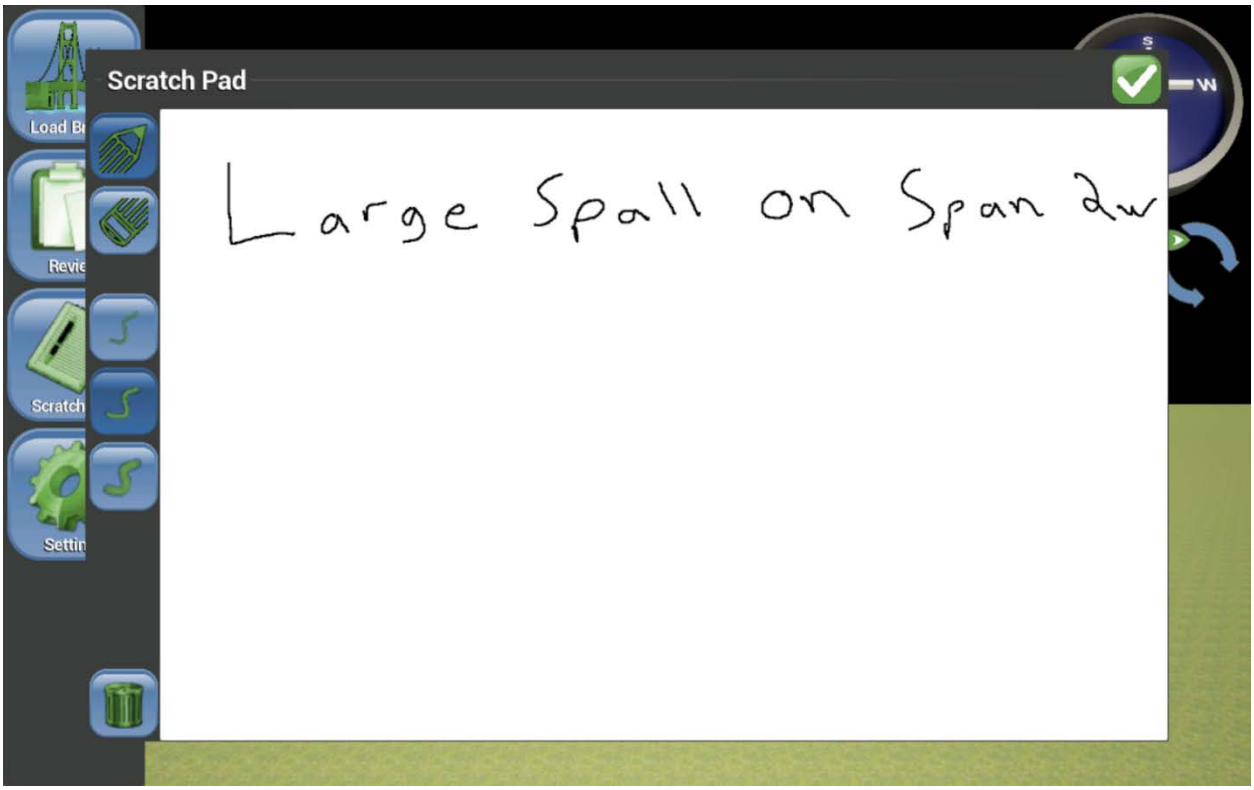


Figure 6-12: The scratch pad gives inspectors a place to write/draw notes that are not included in the report.

6.8 Linear Defects and Defect Aggregation

Certain bridge elements, such as railings and abutments, are measured in linear feet rather than area. Since all defect information is handled by placing area defects on a

surface, there must be a method for converting area-based defects to linear quantities. The application handles this by projecting the polygons of the area defects onto a one-dimensional line at the base of the elements. For example, defects on the inside, outside, or top of the railing will be projected onto a line parallel with the long dimension of the railing before aggregation, while defects that are placed on the ends of the railing will be excluded from the aggregate value since they do not contribute to the linear quantity. Aggregation proceeds from Severe defects to Poor and then to Fair. At each condition-rating stage, when the aggregate quantity for that stage is computed, area that overlaps with regions that have a more severe condition rating are excluded. The result is a total linear quantity for the element in which all area defects are included but in which overlapping quantities are not counted multiple times: A severe defect located spatially below a poor defect will supersede the poor defect in the aggregate quantity. In the defect pop-up menu, the inspector may choose to define a particular defect as linear; however, such a defect will be represented in the application as a quadrilateral polygon with an assumed width of 6 inches. These “linear defects” serve as a quick way to represent cracks, but it is the projection algorithm that truly computes the linear quantity.

The bridge deck is the largest element in any bridge model, and typically will have defects on the top and bottom surface. The bridge deck does use an area-based metric, so area defect aggregation must occur in a 2D plane. All defects are projected into the 2D plane, and then aggregation proceeds analogously to the 1D case (described in the previous paragraph), in which the most severe defects are aggregated first and then the combined region is excluded from overlapping but less severe defects. The application uses a polygon operator library, Clipper Lib, to perform the necessary polygon union and intersection operations.

6.9 Saving/Loading and Importing/Exporting

As with any computer-based application, it is vitally necessary for the users to be able to save and load their work at any time to the local device. Such capability is a hedge against software failure and user error. To this end, the application includes both a named-save file system and an autosave feature. At any time, the user may enter a unique name identifying a particular inspection and then save the current progress of that inspection as a file on the local device bearing the chosen identifier. These save files may be restored at any time, and will return the loaded bridge model, all defects, and NBI report data to the state they were in at the time the save file was created, allowing the user to undo inadvertent changes or to resume the inspection at a later time. The autosave feature activates every time the user modifies a defect on the bridge

surface. There is only a single autosave for the entire application, so it is not a reliable way to save data for future us as it is frequently overwritten, but it does provide a way to recover quickly from a software failure such as an application crash, or from a limited hardware failure such as a depleted battery. Once the device is operating properly and the software is running, the autosave may be loaded, restoring the inspection to the state it was in as of the most recent modification to any bridge defect.

The final critical element is importing and exporting inspection information so that it may be integrated into the MDOT BMS database. Exporting an inspection generates an XML file that includes the original bridge model and NBI information, but included are all the NBI values as well as each individual defect and its location on the bridge model surface. As an XML file, this information can be uploaded to MDOT servers and processed into database entries documenting the inspection. When the same bridge is inspected in the future, the same XML format may be used to generate a new inspection that includes the previous inspection data, which can then be imported into the bridge inspection application. This import/export system was implemented as an interim substitute for full integration of the 3D Bridge App with the MDOT BMS database. Full integration is awaiting MDOT approval that fits into its schedule of database upgrades.

7. Conclusions

After reviewing nationwide bridge inspection practices and discussing current practices and needs with bridge inspectors, MTRI staff developed the 3D Bridge App to render 3D bridge models and interactively tag them with AASHTO element-level defect information. Currently, bridge models are generated using information gleaned from MDOT's BMS database and then tuned with user input. The new system will allow bridge inspectors to gather element-level information efficiently while eliminating the manual data entry present in the current state of practice.

While this project had a specific scope, future development of the 3D Bridge App would be a logical and very promising follow-on to the first two phases of development and implementation-focused improvement. Should MDOT develop a more detailed set of bridge models (such as by obtaining the engineering design files used in bridge construction) that have the necessary metadata, such as element type and category (Deck, Substructure, etc.), then the application could be modified to work with those models rather than the generic models derived from database attributes. The digital nature of the application also makes it ripe for integration with other operations such as remote sensing overlays and GPS tracking. The app could be extended to work with larger, more complex bridges. Finally, the app's per-defect approach to bridge markup opens up new possibilities for bridge management decision-making and represents a step beyond the current inspection regulations, since the app captures the location of defects in addition to their quantities.

Altogether, MDOT's 3D Bridge App affords cutting-edge improvements in the bridge inspection process, enhancing the efficiency and quality of data collection and interpretation.