11-21-2014

# Design and Development of Geographical Information System (GIS) Map for Nuclear Waste Streams

Sandhya Appunni
sappu001@fiu.edu

**DOI:** 10.25148/etd.FI14110763

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

# DESIGN AND DEVELOPMENT OF GEOGRAPHICAL INFORMATION SYSTEM (GIS) MAP FOR NUCLEAR WASTE STREAMS

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Sandhya Appunni

2014

To: Dean Amir Mirmiran
College of Engineering and Computing

This thesis, written by Sandhya Appunni, and entitled Design and Development of Geographical Information System (GIS) Map for Nuclear Waste Streams, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

_____
Leonel E. Lagos

_____
Shu-Ching Chen

_____
Tao Li, Major Professor

Date of Defense: November 21, 2014

The thesis of Sandhya Appunni is approved.

_____
Dean Amir Mirmiran
College of Engineering and Computing

_____
Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2014

DEDICATION

First, I would like to thank god almighty for overcoming all the obstacles that came in while working on my thesis. I would like to my parents, T.M. Appunni and Geetha Appunni, my parents-in-law, K.P. Balakrishnan and C. S. Girija and to my husband, Dr. Muni Rubens. Their support, patience, understanding and most importantly their love helped me in the completion of my thesis.

OM SRI GANESHAYA NAMAH

ACKNOWLEDGMENTS

I would like to thank members of my committee, Dr. Tao Li, Dr. Shu-Ching Chen and Dr. Leonel E. Lagos for their help and support. I'm heartily thankful to my mentor at the Applied Research Center (FIU-ARC), Mr. Himanshu Upadhyay, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. I would also like to thank United States Department of Energy for providing data for this study. Finally, I offer my regards to all those who supported me in any respect during the completion of this project.

ABSTRACT OF THE THESIS

DESIGN AND DEVELOPMENT OF GEOGRAPHICAL INFORMATION SYSTEM (GIS) MAP FOR NUCLEAR WASTE STREAMS

by

Sandhya Appunni

Florida International University, 2014

Miami, Florida

Professor Tao Li, Major Professor

A nuclear waste stream is the complete flow of waste material from origin to treatment facility to final disposal. The objective of this study was to design and develop a Geographic Information Systems (GIS) module using Google Application Programming Interface (API) for better visualization of nuclear waste streams that will identify and display various nuclear waste stream parameters. A proper display of parameters would enable managers at Department of Energy waste sites to visualize information for proper planning of waste transport. The study also developed an algorithm using quadratic Bézier curve to make the map more understandable and usable. Microsoft Visual Studio 2012 and Microsoft SQL Server 2012 were used for the implementation of the project. The study has shown that the combination of several technologies can successfully provide dynamic mapping functionality. Future work should explore various Google Maps API functionalities to further enhance the visualization of nuclear waste streams.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| AJAX | Asynchronous JavaScript and XML |
| API | Application Programming Interface |
| ARC | Applied Research Center |
| CSS | Cascading Style Sheets |
| DBLP | Digital Bibliographic Library Browser |
| DOE | Department of Energy |
| EM | Office of Environmental Management |
| FIU | Florida International University |
| GAO | Government Accountability Office |
| GIS | Geographic Information Systems |
| HLW | High Level Waste |
| HTTP | Hypertext Transfer Protocol |
| JSON | JavaScript Object Notation |
| KLM | Keyhole Markup Language |
| LLW | Low Level Waste |
| MIME | Multipurpose Internet Mail Extensions |
| MLLW | Mixed Low Level Waste |
| MTOM | Message Transmission Optimization Mechanism |
| REST | Representational state transfer |
| RSS | Rich Site Summary |
| SOA | Service-oriented Architecture |
| SOAP | Simple Object Access protocol |
| SQL | Structured Query Language |

| | |
|---|---|
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| WAS | Windows Activation Service |
| WCF | Windows Communication Foundation |
| WIMS | Waste Information Management Systems |
| WSDL | Web Services Description Language |
| XHTML | Extensible Hypertext Markup Language |
| XML | Extensible Markup Language |
| XSD | XML Schema Definition |

# CHAPTER 1: INTRODUCTION

Nuclear waste is the leftover product caused by nuclear weapons manufacturing, naval reactors and commercial nuclear power. Radioisotopes used for industrial, medical and scientific reasons also create nuclear waste by-products [1]. The majority of legal and regulatory definitions, specifically adopted for nuclear wastes, center on the actual techniques that create the wastes in the first place [1, 2]. This means the origins of the waste, rather than the waste's distinct scientific properties, determine how to properly dispose of the waste material.

A nuclear waste stream is the complete flow of waste material from origin (where waste was formed) to treatment facility to final disposal [1]. In 2005, the United States Government Accountability Office (GAO) published a report on nuclear waste streams, providing many recommendations for the Department of Energy (DOE). In the report, GAO suggested the DOE to conduct proper life cycle cost analysis for the treatment and disposal of mixed low level waste (MLLW) as well as low level waste (LLW). A year later, in 2006, the DOE published an advanced summary of its National Low Level Waste and Mixed Low Level Waste Disposition Strategy [3]. This time, it carefully outlined its long-term plan for the management and disposal of MLLW and LLW. With a more proactive stance, the agency stated that it is paramount to develop key tools that incorporate proper and complete management of MLLW and LLW treatment and disposal. As a result, based on its own findings, the DOE's Office of Environmental Management (EM) established a new complex-wide MLLW and LLW database for facilitating the disposal of nuclear waste.

### 1.1 Different Nuclear Waste Types

There are several nuclear waste types. The first type of waste is called low level waste or LLW, and they are materials that have been exposed to radioactive material or have become radioactive through exposure to neutron radiation [4, 5]. Some examples of this type of waste include (but are not limited to) filters, wiping rags, reactor treatment water residue, and syringes. The radioactivity that is found in these items varies depending on their use. They can have the same radiation as in their natural state to the levels of a nuclear reactor's vessel inside a nuclear reactor plant. LLW usually are stored on-site, but require licenses depending on their storage specifications, the duration that the materials will be stored for, and plans regarding future transport to another facility, which must be approved by the Department of Transportation [5].

The second type of waste is called mixed low level waste or MLLW, and it comprises of hazardous as well as radioactive constituents [1, 4]. The majority of this waste comes from the research, development, and production of nuclear weapons. Over the next 20 years, waste management will acquire an estimated 226,000 cubic meters of MLLW that will need proper management and disposal [6].

The third type of waste is called high level waste or HLW. This radioactive waste that comes from used up nuclear fuel. Both solids and liquids that contain fission products and reprocessing, by law, should be permanently isolated [7]. In the United States, HLW is currently being stored in underground tanks at the Hanford, Savannah River, and Idaho Plant. The total amount of this waste is estimated to be around 70 million gallons. Since the waste disposal operation began over 30 years ago, no radiation

injuries or serious exposure has occurred which has affected the public, contrary to the highly publicized leaks and spills reported by the media [8].

### 1.2 Amount of Nuclear Waste Produced

The amount of nuclear waste produced is trivial in comparison to non-nuclear wastes. Annually, around 200,000 cubic meter of LLW and 100,000 cubic meter of HLW is produced globally from nuclear power reactors [9]. In the United States, for proper planning of nuclear waste disposal, EM established a new complex-wide MLLW and LLW database. The DOE works in tandem with the Applied Research Center (ARC) at Florida International University (FIU) to create, organize, disseminate, and maintain this complex-wide database system. Currently, the EM supplies all data on nuclear waste stream to ARC for integration of data through the Waste Information Management Systems (WIMS).

### 1.3 Waste Information Management Systems (WIMS)

Moving forward, waste managers at the DOE headquarters (and other DOE sites) require accurate and updated waste forecast and transportation data until year 2050 to find how much and what type of radioactive waste would actually be generated by specific DOE sites. In the past, each local DOE site gathered, organized, and published waste forecast data separately, and each facility stored its respective data in its own unique system [10]. The WIMS, an n-tier web-based system, was created to understand the complete and complex-wide picture, and to record all radioactive waste and shipment statistics from each DOE site. The WIMS increases the understanding of nuclear waste management for all site managers and improves the treatment and disposal forecast, and scheduling. The WIMS also gives officials a way to predict waste transport information

for proper planning of waste shipment transport. The WIMS was developed and maintained by the ARC at FIU [10]. The system lets users to filter through multiple selection standards such as disposal sites, waste sites, waste type, amount of waste and year to predict the waste disposal levels up until 2050. WIMS can be accessed at www.emwims.org.

# CHAPTER 2: BACKGROUND

## 2.1 Service-oriented Architecture

Service-oriented architecture (SOA) is a design philosophy independent of any specific technology. Its main elements are services, which provide business functionality and have independent state and context [11]. According to Papazoglou, [12] "SOA is a way of reorganizing a portfolio of previously siloed software applications and support infrastructure into an interconnected set of services, each accessible through standard interfaces and messaging protocols". Therefore, core elements of SOA are interoperable services, i.e., software modules that can provide a particular functionality. Services is defined as, “self-describing, platform-agnostic computational elements that support rapid, low-cost composition of distributed applications” [12]. Services are used for many functions, ranging from simple request to complicated business processes. A service consists of service interface and service implementation. Services communicate with one another through messages from one service to other service, or by using coordinating activities among two or more services. Services are technology neutral, loosely coupled, and accessible by variety of clients irrespective of their location. Web service is defined as, “a specific kind of service that is identified by a Uniform Resource Identifier (URI) and exhibits the following characteristics: exposes its features programmatically over the Internet using standard Internet languages and protocols; and can be implemented via a self-describing interface based on open Internet standards (e.g., XML interfaces which are published in network-based repositories)" [12, 13].

### *2.1.1 Simple Object Access Protocol (SOAP) and RESTful Web Services*

SOAP is an XML language for definition of message architecture and formats. It uses XML Schema to define the structure of the message [14]. Web Services Description Language (WSDL), which is an XML language used for describing interfaces syntactically is a large set of WS-* specification for security, quality of service, and service interoperability. The aforementioned standards are used to implement classic SOAP web services [14, 15].

On the other hand, REST is grounded on 4 principles: self-descriptive messages, stateful interactions through hyperlinks, resource identification through URI, and uniform interface [14]. REST services are much simpler than SOAP stack, and they are based on HTTP, XML, URI, and MIME standards. This architecture was developed to resolve WS-* stack complexity, because many services can be implemented using only HTTP. Most commonly used HTTP methods are GET, HEAD, PUT, DELETE, and POST. These methods can identify 'retrieve some data' (GET), 'delete that same data' (DELETE), and 'overwrite it with different data' (PUT)" [15]. In the REST architecture, everything is a resource which can be located using URIs. Due to its simplicity, a very large number of service providers are switching to REST [16].

Many researchers have extensively studied on the advantages and disadvantages of SOAP and REST. For example, Flanders [17] tried to list the strengths and limits of each approach, and he concluded that, “both REST and SOAP can be used to implement similar functionality, but in general SOAP should be used when a particular feature of SOAP is needed, and the advantages of REST make it generally the best option otherwise” [17].

### *2.1.2 Windows Communication Foundation (WCF)*

Windows Communication Foundation (WCF) is Microsoft's framework for building service-oriented applications [18]. WCF enables simple development of Web services and Web services clients. The main features of WCF include: development of a loosely-coupled Web services, service interoperability and integration, publication of service metadata, support for multiple transport protocols and encodings, reliable message exchange, transactions, and REST support [18]. WCF contains runtime and APIs for message-based communication between endpoints (clients and services). WCF supports building of SOAP or RESTful Web services. The WCF model identifies between clients (application which initiate message communication) and services (application which answer to client's requests) [18, 19]. Endpoint defines where, how and what message should be send. Services expose these information in metadata format. Clients can use these data to produce appropriate WCF clients and communication stacks [18, 20]. WCF supports HTTP, TCP, message queuing, and peer networking. Messages can be encoded using text, Message Transmission Optimization Mechanism (MTOM) unstructured binary data, and binaries.

The main elements of WCF architecture are: contracts and descriptions, service runtime, messaging, and hosting [20]. Contracts define methods of the service, XML Schema Definition (XSD) input and output message parameters, the used transport and encoding, and security policies. The service runtime defines runtime behaviors of services. The messaging element consists of channels that process messages and their

headers. Services can be run on IIS and Windows Activation Service (WAS), as a Windows service or executable file [20].

WCF supports service workflows. According to Cibraro et al., [21] "a service can be implemented through a combination of primitive activities in a workflow, or a service can be the result of a workflow that orchestrates other services to accomplish a business goal". All workflow service has two components: the workflow itself; and an outer activity. The outer activity regulates how the execution should flow through different inner events [21]. Security of web services is one of the main concerns of users, and WCF offers basic and advanced security settings to protect SOA applications. It supports transport security and message security models, a hybrid model, and many authentication and authorization schemas [21, 22].

## 2.2 Web 2.0

Web 2.0 is a term used to describe World Wide Web sites that use advanced technologies enabled by Ajax and other applications. Earlier, static pages were used for web sites. Web 2.0 technologies include: blogs, RSS, wikis, mashups, tags, folksonomy, and tag clouds.

### *2.2.1 Ajax*

The classic web application model forwards the entire HTML page from server to client computer. Ajax attempts to merge the advantages of traditional and Web-enabled applications aiming to enable interactivity for web application users. Ajax is an important constituent of Web 2.0 applications. According to Murugesan, [23] "Ajax-style programming makes Web pages more responsive by exchanging small amounts of data with the server so that the entire Web page doesn't have to be reloaded each time the user

requests a change". Ajax integrates asynchronous data retrieval (with XMLHttpRequest), standards-based presentation (with CSS and XHTML), JavaScript, data interchange and manipulation (with XSLT and XML), and dynamic display and interaction (with Document Object Model) [24].

### *2.2.2 JavaScript and JavaScript Object Notation (JSON)*

JavaScript is interpreted programming language used by all modern web browsers on many devices [25]. JavaScript specifies the behavior of web pages, and it is mostly used for the client side programming of Web applications. The European Computer Manufacturers Association TC39 committee has standardized the core language of JavaScript as ECMAScript [26].

JSON is a type of format used for interchanging of data and storing it in an organized manner that is easy to access. It is a light weight format that helps the machines to easily format and parse the data. JSON is language independent and makes use of conventions that is familiar to programmers doing programming in languages like C, C++, C#, Java etc. This property of JSON makes it an ideal data interchange format/ language.

## 2.3 Google Maps API

Google Maps is a Web-based service providing geospatial information and includes road maps, satellite views, and street views. Google Maps provide a set of APIs, and the most important API for web application development is Google Maps JavaScript API v3. To use this library, API key should be obtained from Google APIs web page [27, 28]. On the same page, there are couple of examples showing how to work with Google APIs. To initialize a map, "Map options" object must be created to define the center of

the map and zoom level [27]. MAP class is the JavaScript class which characterizes a map. Objects of the MAP class delineate single map on a web page. If more than one MAP class are created, each object will be delineated as separate map in the web page [29].

Some JavaScript objects within Maps API can respond to mouse or keyboard events. The Maps API contains a number of built-in controls: pan control, zoom control, map type control, scale control, etc. Users can programmatically draw on the map using [29]:

- Markers for single location
- Info windows to add text or images at a given map's location
- Polylines to display lines on the map
- Polygons to define a region, and
- Circles and rectangles

A marker identifies a location on a map, and responds to various events, e.g., mouse click [27]. A marker can be animated, its image can be customized, and it can be set to allow or to not allow users to drag it to another location. Info windows are used to display text or some image at a given location on the map, and are typically attached to a marker [27, 28]. Polyline object has an array of locations defined by latitudes and longitudes, and it is used to draw a line connecting those locations. Similar to Polyline, Polygon object consist of array of locations and represents an area enclosed by a closed path [27, 29]. In addition to Polygon class, API includes classes for rectangles and circles to simplify their depiction on a map [28]. To place chosen image on the map, GroundOverlay object from JavaScript API should be used. Layers consist of one or

more elements, but are manipulated as a single unit on a map. Google Maps API can display road map view, satellite images, a combination of standard and satellite views, and map on terrain information [28, 29]. The API supports 45° imagery for some locations. Custom map types are supported from version 3 of JavaScript API. Google Maps API supports mobile devices (Android and iOS). Maps can be localized by setting language and region.

API also offers some useful services. *DirectionServiceObject* enables calculation of routes from origin to destination for a specified mean of transportation. Google's Distance Matrix service calculates the distance travelled and the total time taken for the journey between origin and destination depending on the mode of travel chosen [29]. Elevation data for a location on the earth can be calculated by using *ElevationService* object. Geocoder object can convert addresses into geographic coordinates and vice versa. Users can arbitrarily change visualization (style) of map elements as well. Creating styles by hand is not trivial, so users can use Google's Styled Map Wizard to select features and choose style, and save styles to JSON, and use in client web application [27, 29].

### 2.4 Relational Database Management Systems

According to Ramakrishnan et al., [30] database is, "an integrated collection of data, usually so large that it has to be stored on secondary storage devices such as disks or tapes. This data can be maintained as a collection of operating system files, or stored in a database management system (DBMS)" [30]. Database management system can design logical and physical schemas, secure the data, and recover from possible failures. A relational database management system is based on the relational model. In the

mentioned model, data has tabular organization consisting of rows of items and columns of attributes. Tables can have relationships with each other using foreign keys [31].

### *2.4.1 SQL*

SQL is a standardized query language for relational database systems. It is used to retrieve, manipulate, and modify the data. The four basic statements of SQL are SELECT, UPDATE, INSERT, and DELETE. The most common operation of SQL is SELECT query which is used to recover data from tables [32].

## 2.5 Current Waste Information Management System

EM needed to make an accurate estimation of the amount, quantity and type of present and future radioactive waste streams for the management of LLW and MLLW treatment and disposal. In order to meet this requirement EM had to develop a complex-wide LLW & MLLW database. So, EM started working with the ARC to develop a system that will forecast waste stream related data. EM took the responsibility of collecting and validating the waste forecast data from every DOE sites and then forwarded the information in the form of data to the ARC for deployment and integration. When the proposal of developing such a waste forecast system was put forward, the initial course of action taken was to collect data from each DOE site and publish the information on web as forecast data table, disposition pathway map and GIS map. Later, it was decided that instead of publishing information on each DOE site, it will be better to gather waste forecast information from all DOE sites and facilities and make it available on web for the site mangers and public. This resulted in the development of a custom system software named ‘Waste Information Management System’.

The WIMS is a web-based information management system, designed, developed, deployed and maintained by the ARC at FIU for DOE site waste managers. The main objective of developing this system was to provide the DOE headquarters and the site mangers with a system that was easy in visualizing, understanding, and managing waste volumes and categories related to waste stream. The WIMS is a user friendly online tool that helps in gathering, organizing and showing the forecasted data from different DOE sites. This system provided a way for the identification of forecasted volumes of waste, different material classes, various disposition routes, and likely choke points and barriers to final destination.


**Figure 1.** User interface screen of the WIMS.

Figure 1 shows the user interface of WIMS with five modules. First module is the Forecast Data module, second the Disposition Map module, third GIS Map module, forth Transportation module and last the Report module. The Forecast Data module is used to

forecast the waste information from different sites to facilities in the form of a table on the basis of desired filtration choices present on the user interface. Disposition Map module is used to display the disposition pathway from a particular 'waste from' location to 'waste to' locations. The Disposition Map will give the site name (where the waste is produced) and facility name (where the waste is being taken), type of waste being carried, physical form of the waste, and the volume of waste carried. Transportation module is the module in WIMS which gives information on the mode of transportation being used to transport the waste from a reporting site to different disposition facility. Main modes of transportation are truck, intermodal and rail. The transportation forecast data will display information about the number of truck, rail and intermodal shipment in addition to the reporting site and facility name. GIS Map module forecasts the disposition of waste from a particular point of origin to the intended treatment or disposal sites on static map image (Figure 2).

### *2.5.1 GIS Map Module of the Existing WIMS*

As mentioned above, the GIS map module generates a static map that forecast waste transfer from a particular reporting site to an intended disposal site. Information can be displayed in both directions i.e., from reporting sites to different disposal sites or from a disposal sites to different reporting sites. The user interface of this module has different filters to display information. Different filtering queries available on GUI are 'Waste disposed from (reporting sites)', 'Waste disposed to (disposal facility)', 'Fiscal year', 'Waste type' and 'Base quantities' and 'American Recovery and Reinvestment Act (ARRA) quantities'.

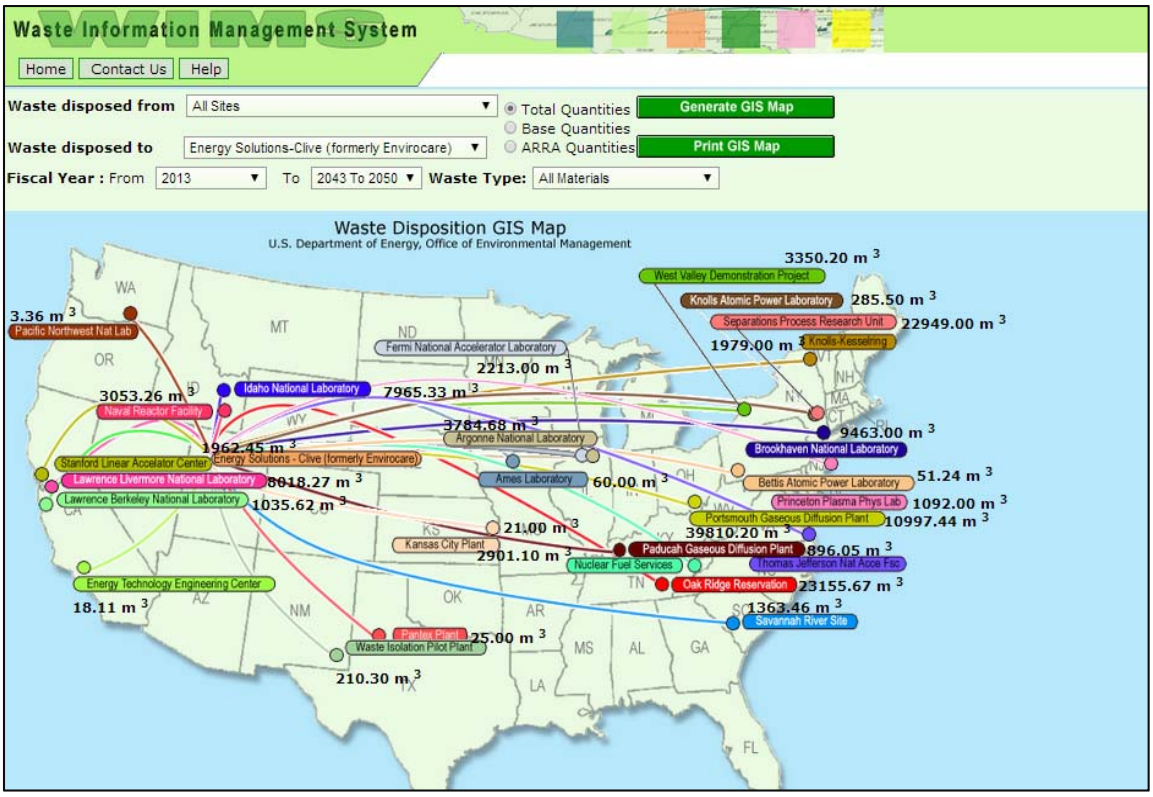


**Figure 2.** GIS Module of WIMS.

### *2.5.2 Uses of current WIMS*

- Assist DOE and local site managers in preparing for goals and milestones of individual sites
- WIMS is used to share resources and treatment capabilities among different sites. This allows sites to influence expertise and capacities
- WIMS provides information to technology vendors on waste disposal needs. This is used to plan and prepare for forthcoming technology capacities and capabilities

- WIMS provides detailed information on complex-wide waste streams. This is used to improve the efficacy of scale during the outsourcing of treatment and disposal of waste

## CHAPTER 3: RESEARCH OBJECTIVES AND METHODS

WIMS is a complex-wide, high performance program that includes forecasting data, geographic information system (GIS) maps, disposition maps, and transportation and custom reports based on DOE requirements. The WIMS system was constructed using SQL Server Reporting Services® and SQL Server® 2005 Integration Services on a SQL Database. Based on specific components and attributes, WIMS can provide critical waste stream data to users in the forms of disposition maps, data tables, and GIS maps. However, the current nuclear waste stream GIS map module has the following drawbacks: static image display through C# programming; addition and/or deletion of waste origin location (site) and/or waste destination location (facility) that require redesigning of image; and changes in existing disposition paths that require redesigning of images. Redesigning of images is a time consuming process and an alternative is a dynamic Google map-based system. This study designed and developed a dynamic Google map to display nuclear waste stream that identified and displayed the waste origin location, the waste destination location, volume of waste transported, year of transport, and type of nuclear waste transported.

In addition, the study enhanced the visualization of the map. Since there are multiple locations, the polyline connecting “waste from” location with “waste to” location on the Google map may intersect with other nearby polylines. This might confuse the viewers about the direction of waste transfer and hence affect the visualization of the map. In order to fix this issue, this study developed an algorithm which prevented the overlapping or intersecting of the polylines through quadratic Bezier curve to make the map more understandable and usable.

## 3.1 Research Objectives

The main objectives of the study are: (1) research and analyze WCF on Microsoft .NET platform; (2) design and develop waste streams services using WCF; (3) technical research on the features of Google Maps API and associated technologies; (4) analyze and evaluate the identified Google Maps API features to implement GIS module of WIMS; (5) design and development of GIS module using Google API and waste stream services; and (6) develop an algorithm to prevent overlapping of polyline for enhanced visualization of the map in presence of multiple locations.

## 3.2 Methods

### *3.2.1 Microsoft Visual Studio 2012 and Microsoft SQL Server 2012*

For developing the application, Microsoft Visual Studio 2012 and Microsoft SQL Server 2012, tools were used. Microsoft Visual studio 2012 is the latest integrated development environment from Microsoft. It was used in developing a GUI application. Microsoft SQL Server 2012 was used as the relational database management system.

### *3.2.2 Designing and Development of Database Using SQL Server 2012*

A database was designed based on the information to be shown on the Google map. The database helped to retrieve information as requested by the user.

### *3.2.3 Development of Waste Stream Services Using Windows Communication Foundation (WCF) Framework*

The back end of the application had the waste stream services developed using WCF, which acted as a mediator to communicate with the database. WCF is a web based framework for creating service-oriented applications [19]. Its main features are that it is service oriented, interoperable, sends multiple message patterns and enables security

[18]. On request from the client, the waste stream services will grab the requested information from the database and send it to the client and it will be displayed on the Google map. The technologies to be used for data interchange are the following:

**(a) Asynchronous JavaScript and XML (AJAX):** AJAX is a group of technologies used in the client-side [33]. An HTTP (hypertext transfer protocol) request is given to the waste stream services using this technology. It prevents the reloading of the webpage after each function. It exchanges only the required information, thus it is a quick responding interface with faster web processing time and load time for a particular event.

**(b) JavaScript Object Notation (JSON):** JSON is a data-exchange language, which can be read by humans and easy for systems to analyze, describe and use. Inside the JavaScript, JSON is directly supported and well matched for JavaScript applications [33]. In return to the HTTP request to waste stream services, a JSON response is given which has the serialized data of requested information.

### *3.2.4 User Interface Development Using Google Maps API*

The front end of the application have a GUI consisting of query lists like "waste disposed from" location (36 locations), "waste disposed to" location (31 locations), fiscal year (2013 – 2050), waste type (5 types), and total quantities of nuclear waste (in cubic meter). To integrate Google Maps into the application, Google Maps API's are required. The latest Google Maps API was selected for integrating the Google map on the system by undergoing a brief research on Google Maps API. Google Maps API is unique in a way that all its components are located in JavaScript container in a XHTML page. When Google Maps web page is opened, these components are loaded. The final outcome

display showed “waste disposed from” and “waste disposed to” locations with markers connected by a polyline on the Google map. For easy identification, “waste disposed from” locations were displayed in red and “waste disposed to” locations were displayed in blue markers. “Waste disposed from” and “waste disposed to” locations also displayed InfoWindow showing the name of the facility, year, and type of waste. The amount of waste disposed was displayed on the polyline.

#### *3.2.5 Development of Algorithm for Clear Presentation of Polyline on the Google Map*

The algorithm is a step-by-step procedure to solve the problem of overlapping/ intersecting of the polylines. It made use of mathematics to determine which lines was closer to each other and formulated a relationship between the lines and spaced them accordingly for making the map more understandable and usable. Then, the quadratic Bezier method was directed for forming the curve between the locations. Quadratic Bezier was best method to use when there are clusters of location [34].

### 3.3 Functional Flow Block Diagram

Figure 3 is the functional flow block diagram, displaying the system’s functional flow in a step-by-step manner. According to the functional flow block diagram, the application has two parts, one is the client browser and the other is the server. Client side includes the JavaScript controls and logic, google API and the AJAX routine. The server sides include WCF services and content database. When a user makes selections in the query lists on the GUI developed by this study, Ajax routine will send an XML request to the WCF services to get the requested data from the database. WCF services will then retrieve the data from the database and then send the received data as a response to Ajax

routine in the JSON format. Once the required data is received, then with help of Google API the data is shown on the Google map.

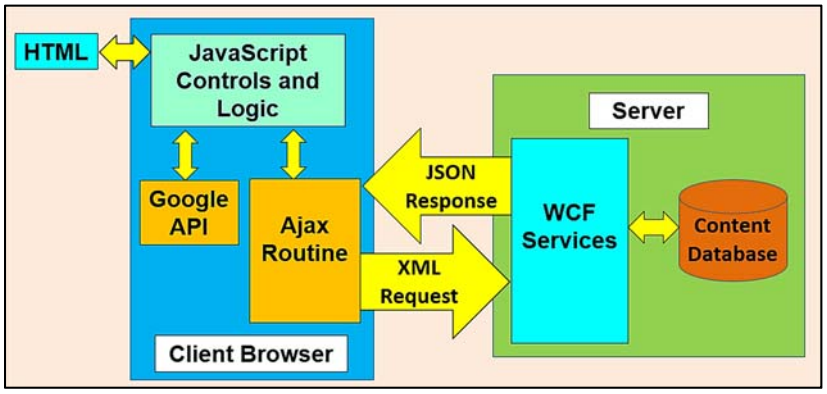


**Figure 3.** Functionality diagram of the application.

# CHAPTER 4: DATABASE DESIGN

Database was implemented in Microsoft SQL Server 2012. Applications described in this study used five tables (interval, qry_FIU_Base, tx_Facilities, tx_Site, tx_WasteType). Tables and their attributes are listed below.

The table, interval (Table 4.1) gives the information on time interval and on the user interface.

**Table 4.1.** Attributes of the interval table.

| Attribute name | Data type | Description |
|---|---|---|
| **interval_id** | int | Interval ID |
| **first_year** | int | First year |
| **last_year** | int | Last year |
| **description** | varchar(50) | Description of the time interval |

The table, qry_FIU_Base (Table 4.2) provides the information on the quantity of waste transferred from reporting site to facility.

**Table 4.2.** Attributes of the qry_FIU_Base table

| Attribute name | Data type | Description |
|---|---|---|
| **dataset_id** | int | Table ID |
| **reporting_site_key** | int | Key of the reporting site |
| **reporting_site_code** | nvarchar(255) | Code of the reporting site |
| **reporting_site_name** | nvarchar(255) | Name of the reporting site |
| **waste_type_key** | int | Key of the waste type |

| | | |
|---|---|---|
| **waste_type_code** | nvarchar(255) | Code of the waste type |
| **waste_type_name** | nvarchar(255) | Name of the waste type |
| **disposition_site_key** | int | Key of the disposition site |
| **disposition_site_code** | nvarchar(255) | Code of the disposition site |
| **disposition_site_name** | nvarchar(255) | Name of the disposition site |
| **disposition_facility_key** | int | Key of the disposition facility |
| **disposition_facility_name** | nvarchar(255) | Name of the disposition facility |
| **disp_fy12_quantity** | float | Amount of waste transferred from reporting site to facility in year 2012 |
| **disp_fy13_quantity** | float | Amount of waste transferred from reporting site to facility in year 2013 |
| **disp_fy14_quantity** | float | Amount of waste transferred from reporting site to facility in year 2014 |
| **disp_fy15_quantity** | float | Amount of waste transferred from reporting site to facility in year 2015 |
| **disp_fy16_quantity** | float | Amount of waste transferred from reporting site to facility in year 2016 |
| **disp_fy17_quantity** | float | Amount of waste transferred from reporting site to facility in year 2017 |
| **disp_fy18-22_quantity** | float | Amount of waste transferred from reporting site to facility for year 2018-2022 |

| | | |
|---|---|---|
| **disp_fy23-27_quantity** | float | Amount of waste transferred from reporting site to facility for year 2023-2027 |
| **disp_fy33-37_quantity** | float | Amount of waste transferred from reporting site to facility for year 2033-2037 |
| **disp_fy38-42_quantity** | float | Amount of waste transferred from reporting site to facility for year 2038-2042 |
| **disp_fy43-50+_quantity** | float | Amount of waste transferred from reporting site to facility for year 2043-2050+ |

The table, tx_Facilities (table 4.3) provides information about the facility (the location where the waste is sent).

**Table 4.3.** Attributes of the tx_Facilities table

| Attribute name | Data type | Description |
|---|---|---|
| **site_id** | int | ID of the site |
| **site_cd** | nvarchar(255) | Code of the site |
| **site_shortname** | nvarchar(255) | Short name of the site |
| **site_name** | nvarchar(255) | Name of the site |
| **group** | nvarchar(255) | Site's group |

| | | |
|---|---|---|
| **state** | nvarchar(255) | State associated with the reporting site |
| **comm** | nvarchar(255) | |
| **lat** | float | Latitude of the site |
| **lng** | float | Longitude of the site |

The table, tx_Site (Table 4.4) gives information about the reporting site (from where waste is taken to different facilities).

**Table 4.4.** Attributes of the tx_Site table

| Attribute name | Data type | Description |
|---|---|---|
| **site_id** | int | ID of the site |
| **site_cd** | nvarchar(255) | Code of the site |
| **site_shortname** | nvarchar(255) | Short name of the site |
| **site_name** | nvarchar(255) | Name of the site |
| **group** | nvarchar(255) | Site's group |
| **state** | nvarchar(255) | State associated with the reporting site |
| **comm** | nvarchar(255) | |
| **lat** | float | Latitude of the site |
| **lng** | float | Longitude of the site |

The table, tx_WasteType (Table 4.5) gives information on the type of waste transferred from a reporting site to facility.

**Table 4.5.** Attributes of the tx_WasteType table

| Attribute name | Data type | Description |
|---|---|---|
| **wtype_id** | int | Waste type ID |
| **wtype_cd** | nvarchar(255) | Code of the waste type |
| **wtype_nm** | nvarchar(255) | Name of the waste type |

# CHAPTER 5: IMPLEMENTATION

There are four main steps to the implementation of this study:

1) Creating web project
2) Creating RESTful web services
3) Configuration setup
4) JavaScript logic on the client side

## 5.1 Creating Web Project

The web project was created using Microsoft Visual Studio 2012, with database backend in Microsoft SQL Server 2012. A new web project was created using ASP.NET Empty Web Application template (Figure 4). This template creates an empty project with a Web user interface, and contains only Web.config configuration file.

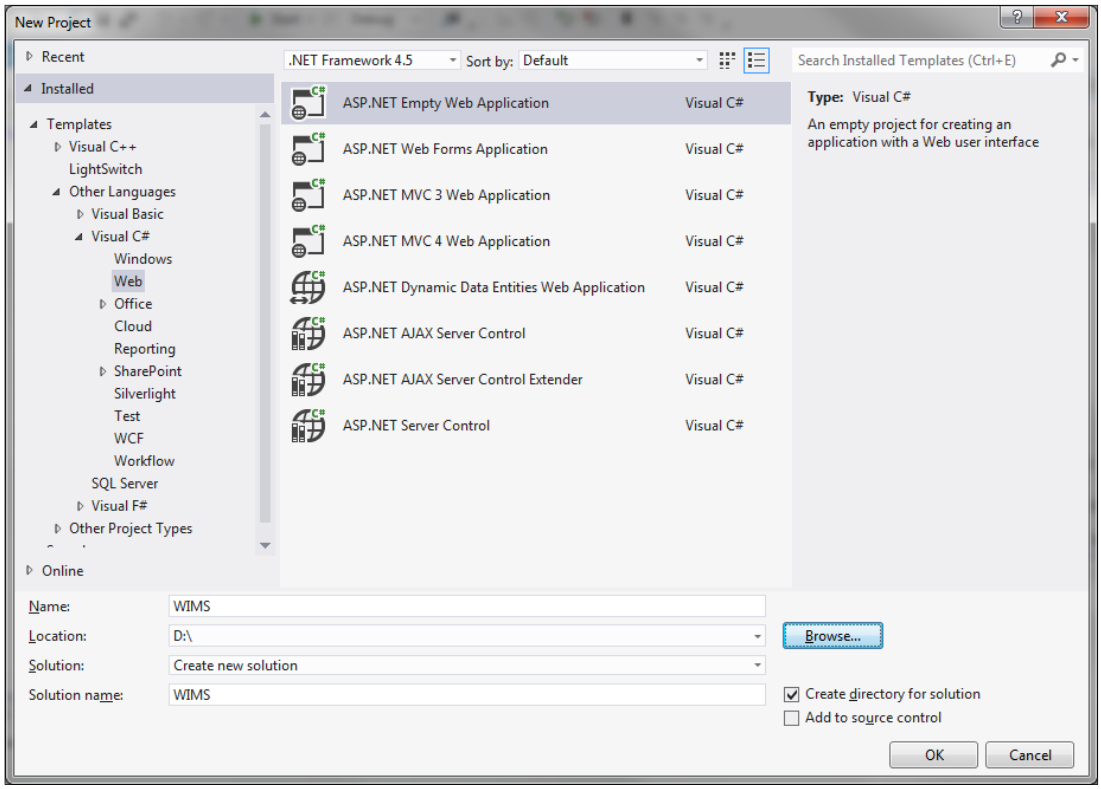


**Figure 4.** Initial step in creating empty web application.

Then web form (.aspx file) was added to the project. Project was right-clicked, and Add->Web Form was chosen.

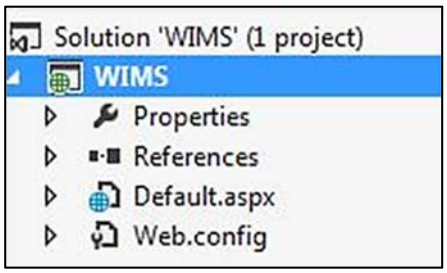


**Figure 5.** Initial structure of the new web project.

The auto generated code in the Visual Studio empty web form was erased and the following HTML elements were added to the web form file:

```
<div class="clearfix" style="height: 100%; width: 100%; overflow: hidden;">
        <div style="position: relative; height: 150px; width: 100%;
background-color: #EBFBE3; border-bottom: 1px solid #000000;">
            <table>
                <tr>
                    <td>
                        <table>
                            <tr>
                                <th>Waste Type</th>
                                <th>Year to Year</th>
                            </tr>
                            <tr>
                                <td>
                                    <select id="waste"></select>
                                </td>
                                <td>
                                    <select id="fromYear"></select>
                                    <select id="toYear"></select>
                                </td>
                            </tr>
                            <tr>
                                <th>To Facilities</th>
                                <th>From Facilities</th>
                            </tr>
                            <tr>
                                <td>
                                    <select id="site1">
                                        <option value="-1">All</option>
                                    </select></td>
                                <td>
                                    <select id="site2">
                                        <option value="-1">All</option>
```

```
                                </select></td>
                            </tr>
                            <tr>
                                <td colspan="2">
                                    <input type="button" value="Set Pos"
id="SetPos" />
                                </td>
                            </tr>
                        </table>
                    </td>
                    <td>
                        <table>
                            <tr><td><input type="radio" name="fiuType"
value="0" checked="checked" />Total Quantites</td></tr>
                            <tr><td><input type="radio" name="fiuType"
value="1" />Base Quantites</td></tr>
                            <tr><td><input type="radio" name="fiuType"
value="2" />ARRA Quantites</td></tr>
                        </table>
                        <div id="log" style="height: 140px; width: 600px;
font-size: 8px; position: relative; top: 3px; left: 25px; overflow: auto;
display:none">
                            log:
                        </div>
                    </td>
                </tr>
            </table>
        </div>
        <div id="map_canvas" style="width: 100%; height: 700px"></div>
    </div>
```

This code represented the user interface of the web application. HTML is a markup language for displaying information in web browsers [35]. Table elements were used to align, select and input elements. There were two  elements inside main  element. The aforementioned tag defines a section in an HTML document, and it is supported by all major web browsers. First,  has selections (for waste type, time interval, facilities) that one can choose, and second, div was the map with identifier (id) *map_canvas*. Later, the mentioned elements were referenced from JavaScript. Next, code described in Figure 6 was added to the head of the web form.

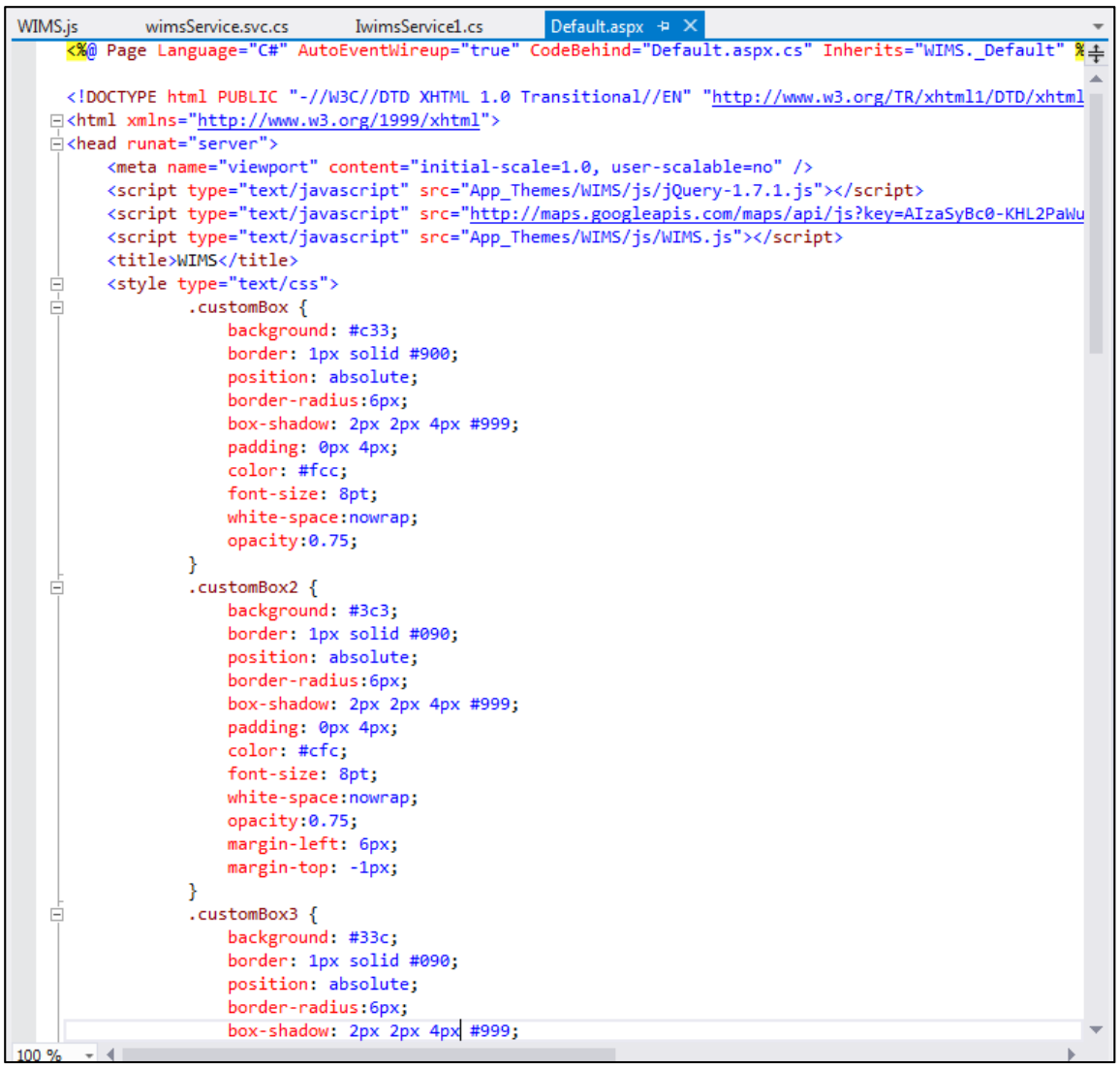


**Figure 6.** Code added to the head of the web form.

The code in Figure 6 adds three JavaScript references and Cascading Style Sheets (CSS) style. The CSS described the look and formatting of HTML elements (e.g.,, background, color, padding, position, etc.) [36]. In the CSS, first JavaScript tag imported jQuery library, then Google Maps JavaScript library, and finally the custom file that was created to implement logic for the web application. The *jQuery* is a popular open-source JavaScript library for HTML manipulation, animation, event handling, and cross-browser

Ajax support. JavaScript library streamlines the communication between JavaScript and HTML document like Document Object Model [37].

A new folder was then added to Visual Studio project. It was created by right clicking on project and selecting Add -> Add ASP.NET Folder -> Theme; it was named WIMS and it contained JavaScript logic and auxiliary files such as images and style sheets. Images were saved to */images* subfolder, and JavaScript code to */js* subfolder.

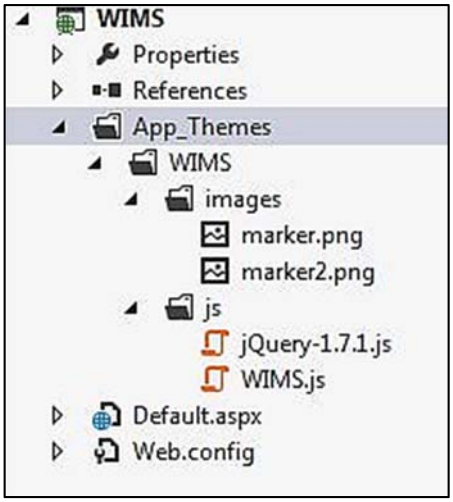


**Figure 7.** Structure of WIMS web project.

Two images (marker.png and marker2.png) were copied into *images* folder, then added jQuery-1.7.1.js and empty file named WIMS.js.

### 5.2 Creating RESTful Web Services

The REST architecture was chosen to implement backend web services. The first step was to add WCF Services project to the Visual Studio solution (Figure 8). Add->New Project was chosen, and C# -> WCF -> WCF Service Application was selected in a new project dialog. The project was named WCF.

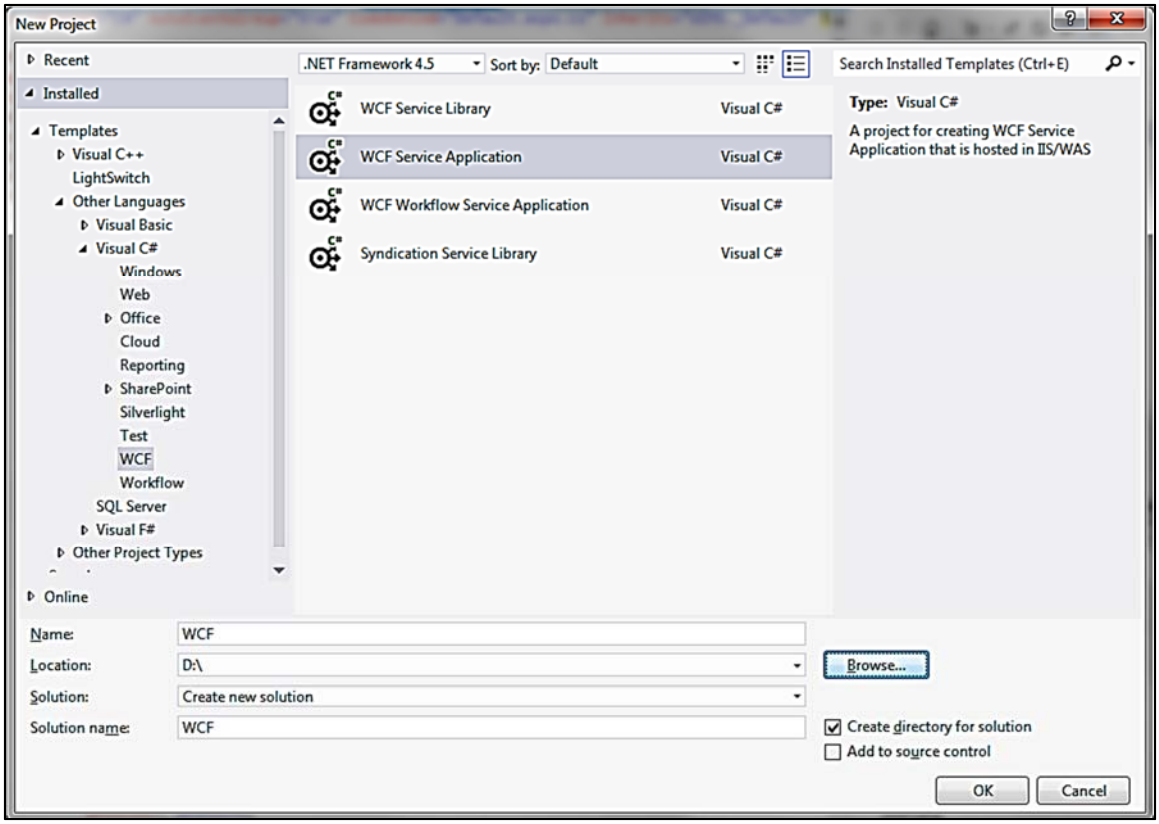


**Figure 8.** Wizard for creation of WCF service.

Visual Studio created three files, IService1.cs, Service1.svc and Web.config. The files were renamed to IwimsService1.cs and wimsService.svc. IwimsService1.cs was the WCF service interface file, and wimsService.svc was the implementation file. IwimsService1.cs file has [ServiceContract] and [DataContract] sections. In service contract section, operations that were called by the service were defined and in data contract section objects that service returned in JSON format were defined. First operation in the web service was *getAllTx_WasteTypes*.


**Figure 9.** Contract of the *getAllTx_WasteTypes* operation.

The mentioned service was used to populate option menu *wasteType* on the client side. This WCF service was invoked using the GET web method, response format was JSON, and body style was wrapped. The *getAllTx_WasteTypes* operation (Figure 9) was called in the web browser using the following URL: *http://localhost:55464/wimsService.svc/json/getAllTx_WasteTypes*. Since this operation was returning complex type (list of *tx_WasteType*), this was defined as a data contract.

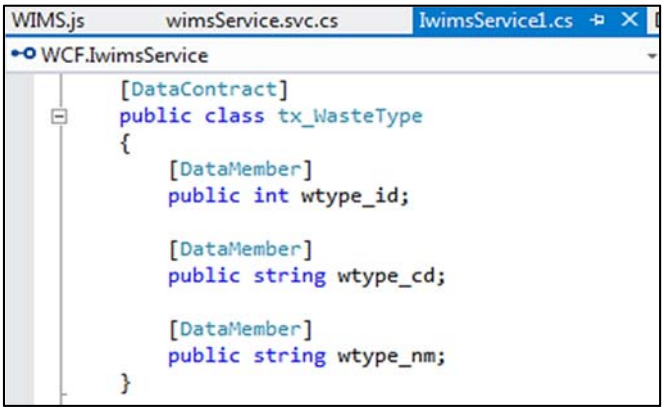


**Figure 10.** The *tx_WasteType* data contract.

The next step was the implementation of this operation in .svc file. First, constructor (the method that get executed on service call) for service was defined in which a connection to database was established.

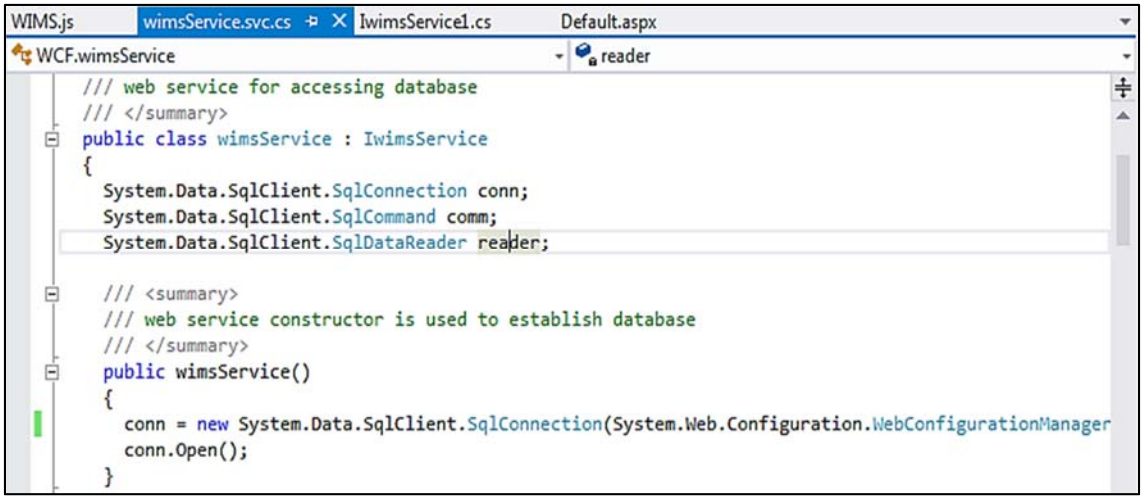


**Figure 11.** The constructor of the web service.

Then the code described in Figure 12 was added to implement the operation.


**Figure 12.** Implementation of the *getAllTx_WasteTypes* operation.

First line of the code initialized a new list of waste type. The second line created new *SqlCommand* object and assigned it to variable, comm. It contained SQL query to retrieve a list of *tx_WasteType* objects from database. Third line executed command and

returned *SqlDataReader* object which was assigned to reader variable. The next step involved creation of new empty list of *tx_WasteType* object, execution of SQL command and assignment of result to a reader object. In while loop, results were fetched while there were any, and for each result fetched, new object of type *tx_WasteType* was created, assigned values from database to it, and added it to the list. When the loop is done, the filled list was returned.

The next operation in the web service was *getAllIntervals* (Figure 13). It was used to populate option menus of time intervals on the client side.


**Figure 13.** Contract of the *getAllIntervals* operation.

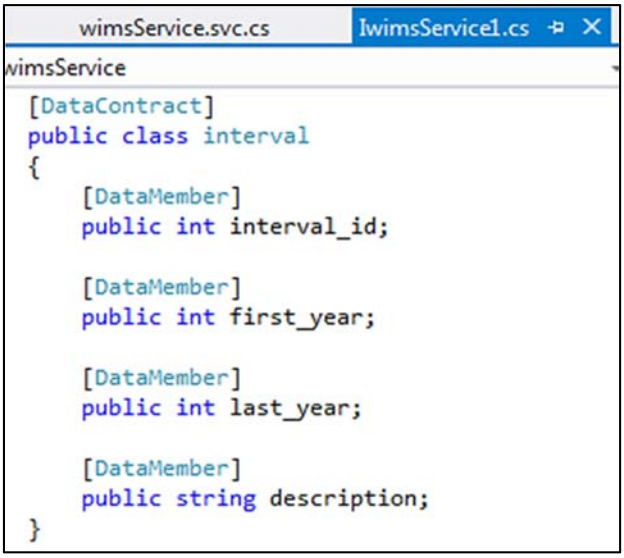


**Figure 14.** The interval data contract.

This operation was similar to the *getAllTx_WasteTypes*. The operation in Figure 15 returned a list of intervals obtained from *"interval"* table using SQL query *"SELECT * from interval"*.

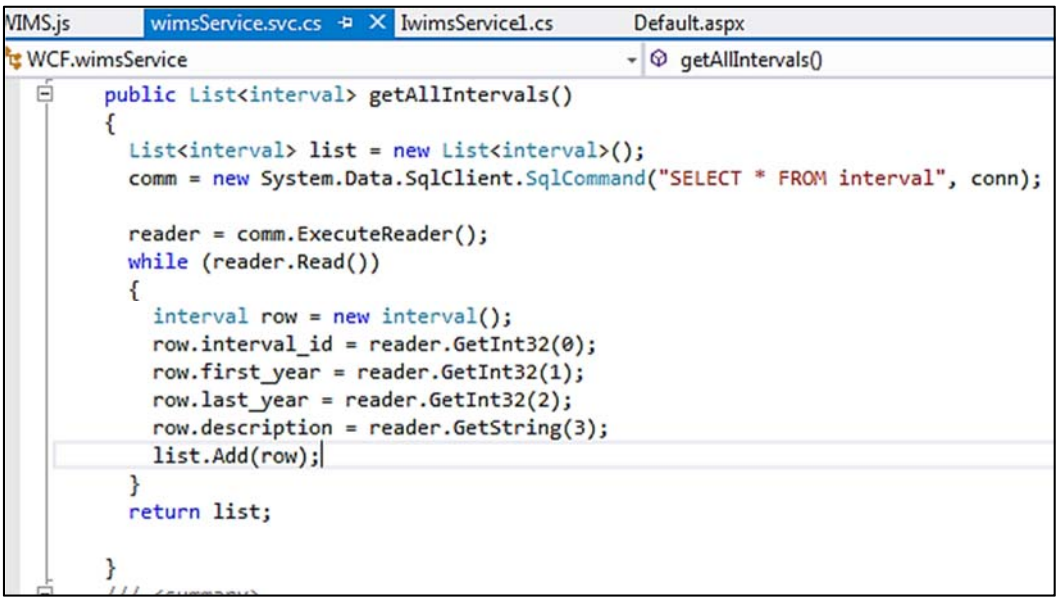


**Figure 15.** Implementation of the *getAllIntervals* operation.

Then, the *getAllTx_Sites* operation used to populate option menus of sites on the client side was defined (Figure 16).


**Figure 16.** Contract of the *getAlltx_Sites* operation.

The operation *getAlltx_Sites* returned list of all sites ordered by site name (Figure 17). The SQL command *"SELECT * FROM tx_Site ORDER BY site_name"* was executed to get the list from *tx_Site* of the backend database.

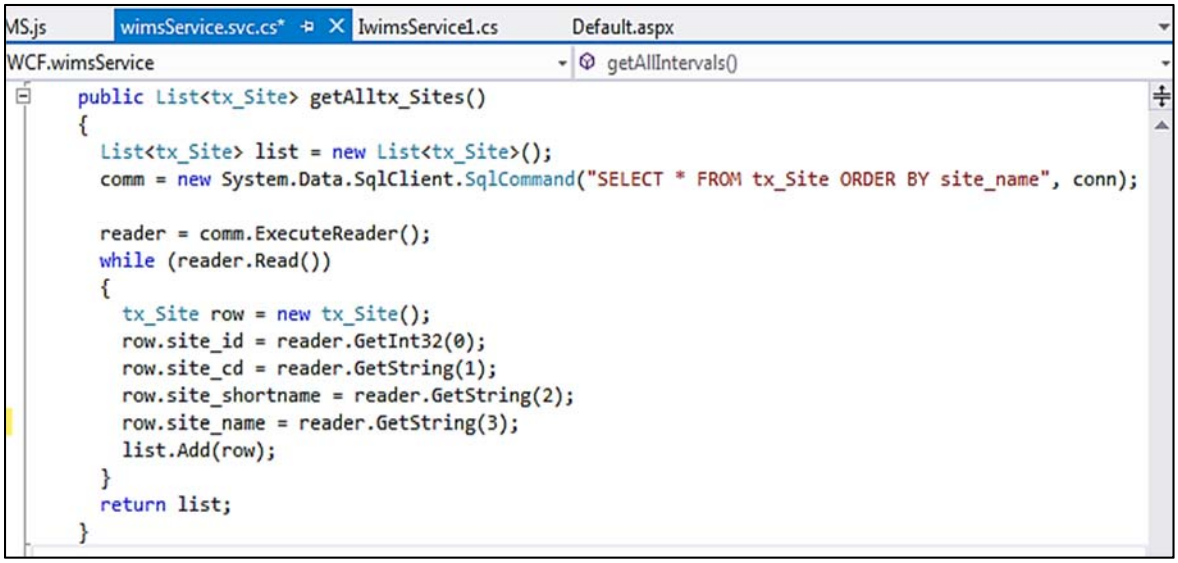


**Figure 17.** Implementation of the getAlltx_Sites operation.

Next, *getAllTx_Facilities* (Figure 18 and Figure 19) operation was added to the web service implementation.

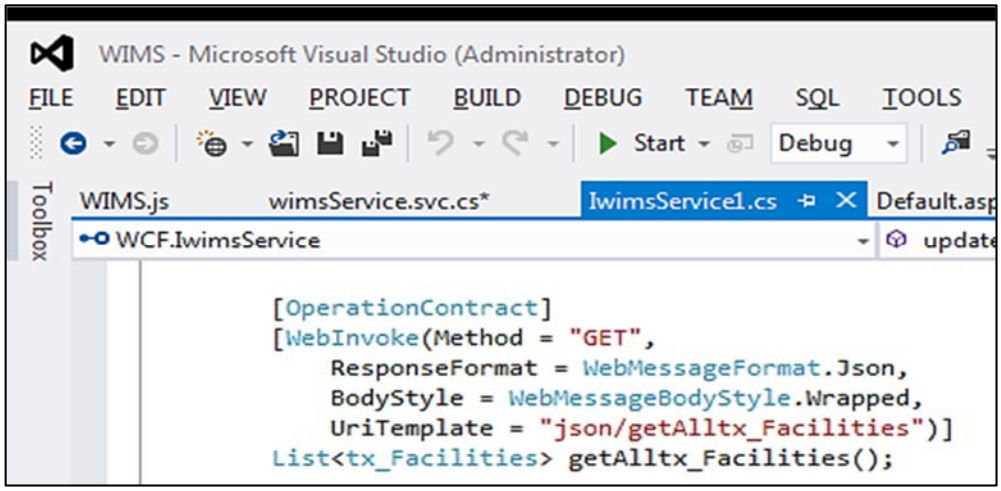


**Figure 18.** Contract of the *getAllTx_Facilities* operation.

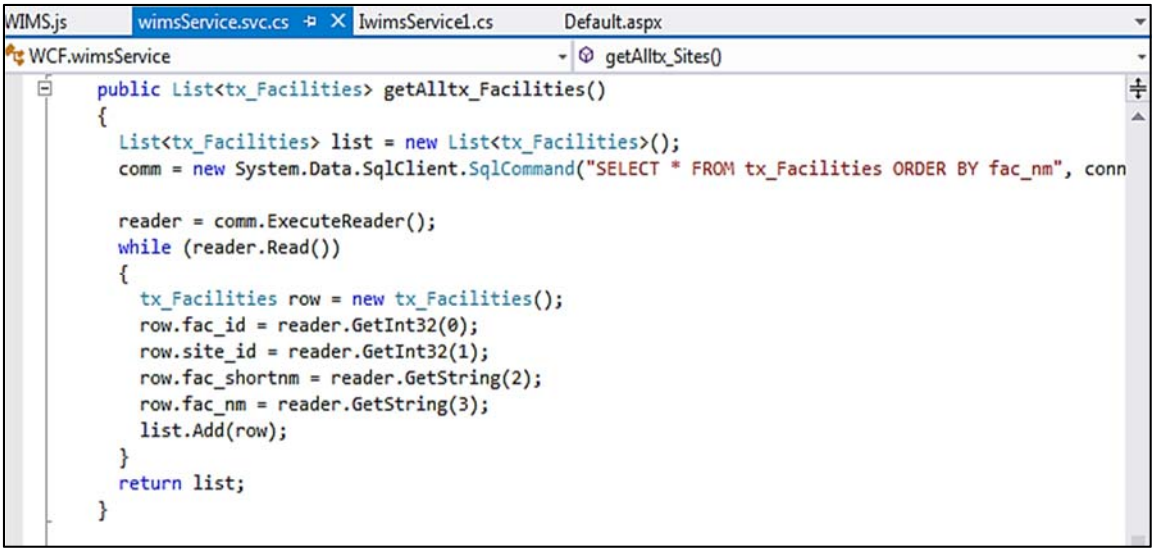


**Figure 19.** Implementation of the *getAlltx_Facilities* operation.

Then, the SQL query *"SELECT * FROM tx_Facilities ORDER BY fac_nm"* was constructed to obtain list of facilities from backend database. The last implemented operation was *getFiu*. It was used for drawing disposals on the map. The operation *getFiu* returned the list of waste disposals according to given criteria defined by operation parameters (Figure 20). Parameters were waste type, reporting site, disposition site, starting year, ending year, and FIU type.

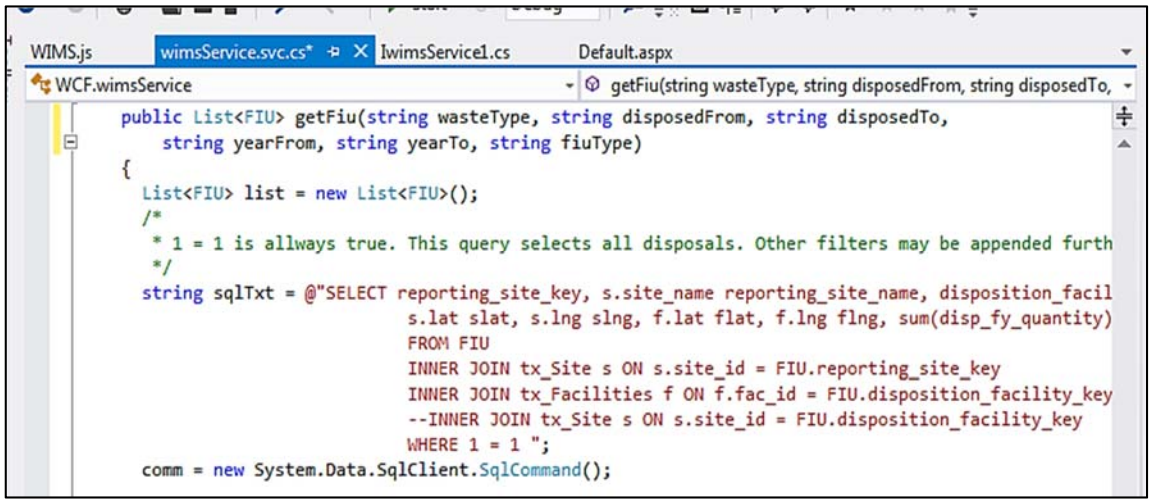


**Figure 20.** A portion of *getFiu* implementation.

First, a basic query string (Figure 20) was created to select all disposals. Then, "WHERE 1=1" was used in the where clause in the query because 1=1 was always true. The next step was to add additional query criteria based on operation's input parameters. If a specific waste type was selected (value 1 means all waste type), the condition *"AND waste_type_key = @waste_type"* was added to the query string. If reporting site was selected, it was added to query criteria, *sqlTxt += " AND reporting_site_key = @disposed_from "*. Other parameters (disposition site, starting year, ending year, FIU type) were also added to the search criteria. At the end of the operation, the filled query string was executed, results were fetched, and list of disposals were returned as a result of the *getFiu* operation.

### 5.3 Configuration Setup

The next step was to set up Web.config file to add a connection to the database (Figure 21). First, <connectionStrings> node was added to configuration.

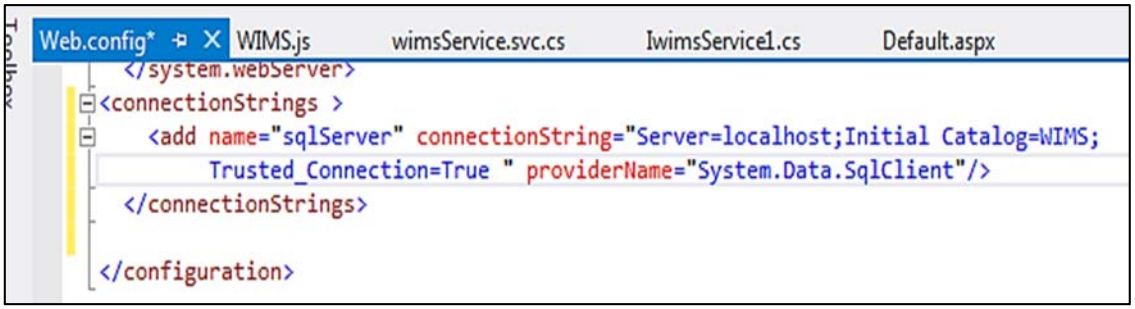


**Figure 21.** Configuration of database connection.

Here, local host connection was added to WIMS database, without username and password, and WCF used Windows credentials for SQL server. When local SQL server database was not used, this connection string should change to contain server address,

username and password. Sections <system.web> and <system.webServer> were left unchanged. WCF configuration was set in <system.serviceModel> node (Figure 22).

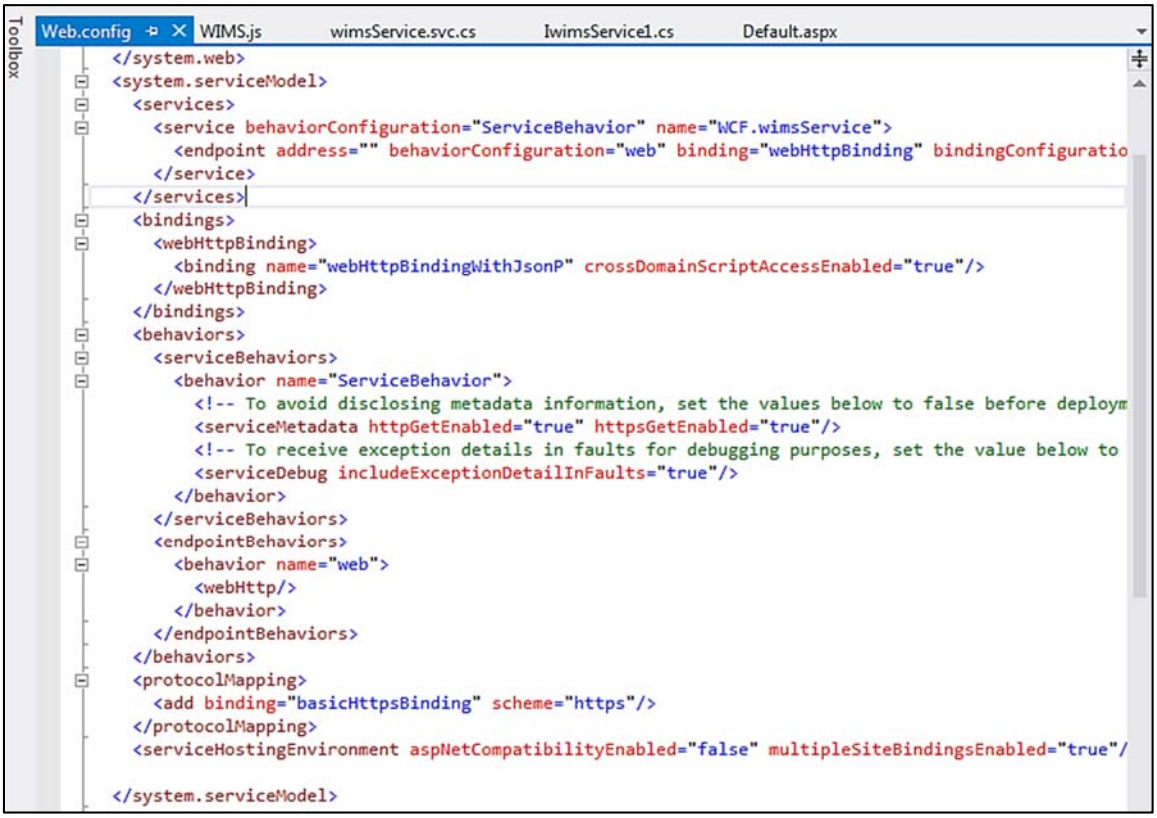


**Figure 22.** WCF configuration.

### 5.4 JavaScript Logic on the Client Side

Most of the application logic was implemented on the client side in JavaScript file found in the WIMS web project in App_Themes/WIMS/js/WIMS.js. On top of the mentioned JavaScript file some variable definitions were added that were mostly parameters for application. Variables map, geocoder and overlay were used for Google Maps API. Variable index was used to loop through colors of polylines. Zoom is a variable for setting up Google Maps zoom of the map, and acceptable values were integers from 1 to 10 [27]. *CenterLat* and *centerLng* are latitude and longitude of map

center, and it was initialized as a location somewhere in Kansas, USA. *ServiceUrl* is the base URL of the WCF service. Users need to change this location to point to Uniform Resource Locator (URL) where they have deployed web service component of application described in this study. *ArrColors* defined array of colors used for polylines, and *precision* determines number of points on each polyline. *BezierDiv* and *bezierPow* are parameters for Bézier algorithm (described in the next chapter). *LocationLblxDist* and *locationLblyDist* are distances in pixels from markers and labels on the map.

The main entry point for client application was *$(document).ready(function ())*. This is jQuery code that gets executed when HTML elements were loaded (Figure 23).

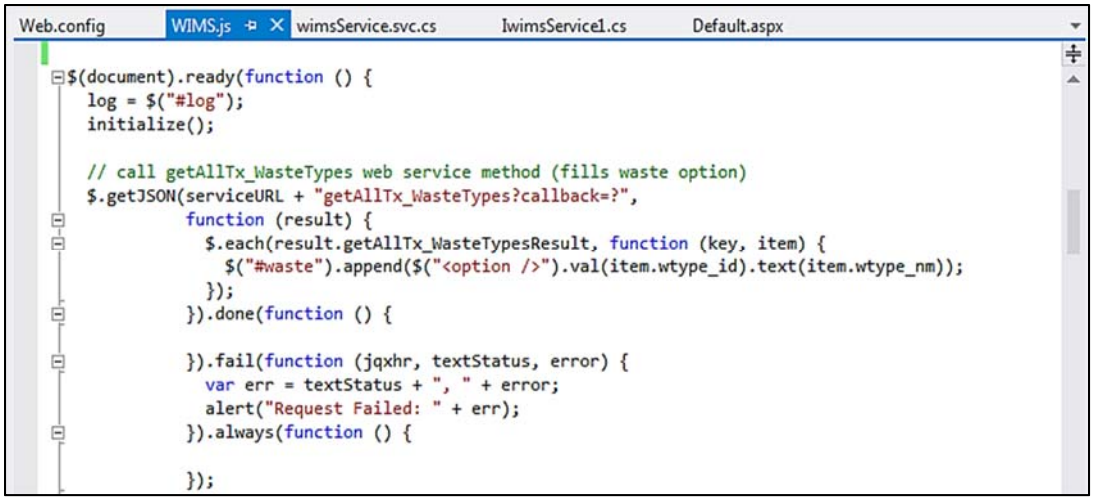


**Figure 23.** Beginning of the main entry point.

First, the function *initialize()* was called that initializes the map using calls to Google Maps JavaScript API file included in the project (Figure 24).

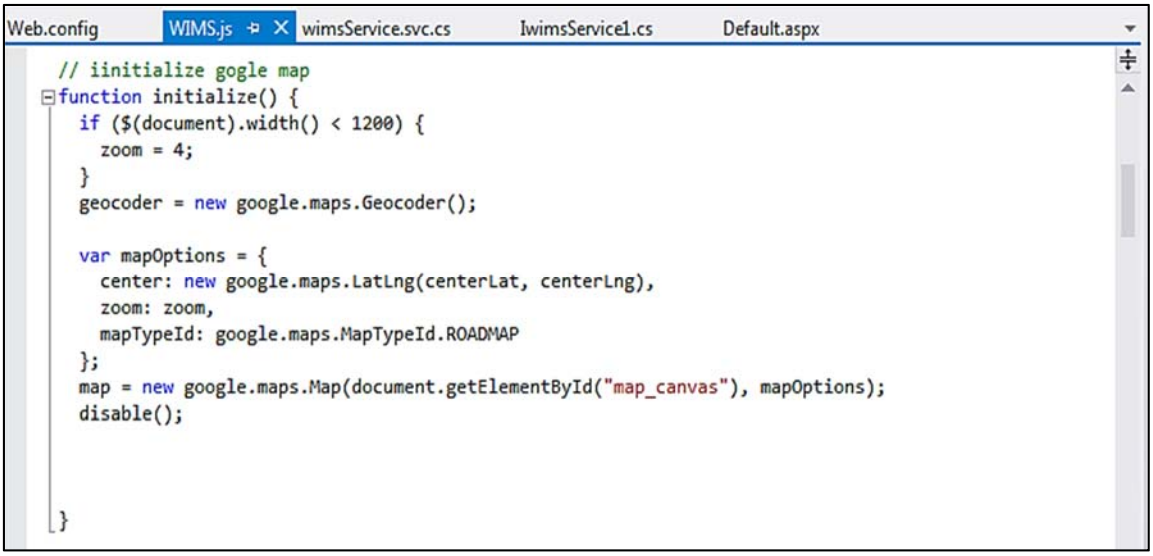


**Figure 24.** The *initialize()* JavaScript function.

Next, WCF methods was called to populate dropdown lists (waste types, time intervals, facilities) on the web page by using $.getJSON AJAX calls to WCF. Afterwards, handler for button were set “Set Pos“ click in $("#SetPos").click(function (). This code was executed when user clicks button. Once *getFiu* method returns results, the code simply displayed locations and polylines on map. The algorithm for polylines is described in the next chapter.

There are number of helper functions in JavaScript file. For example, the function *addLocations* traverses all locations. The function, *getLocationByName* retrieves data location by its name (Figure 25).

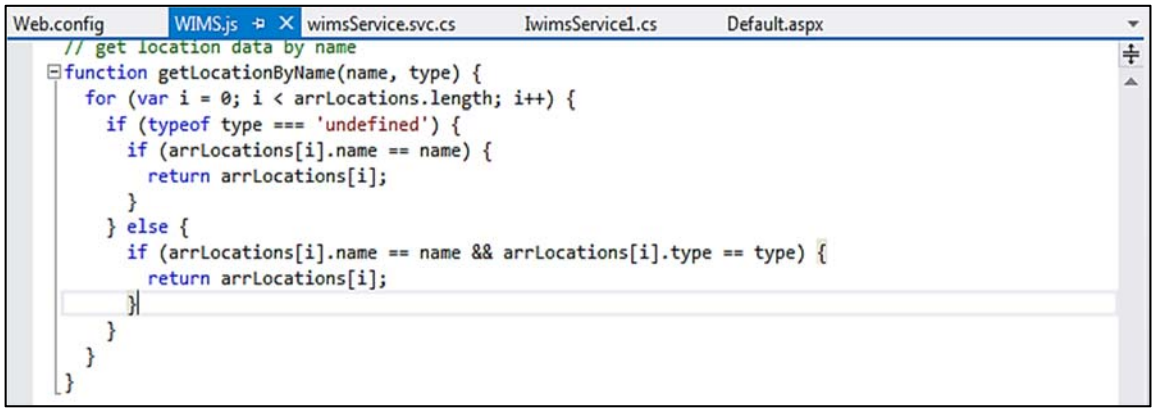


**Figure 25.** The *getLocationByName* function.

The function *displayLocations* displayed all retrieved locations on Google map. This function displayed location markers, names and quantities on the map. *TxtOverlay* object was used to draw text overlay on the map. The helper function *displayDisposals* was used to depict all retrieved disposals on the map; it drew all Bézier curves on the map (Figure 26).

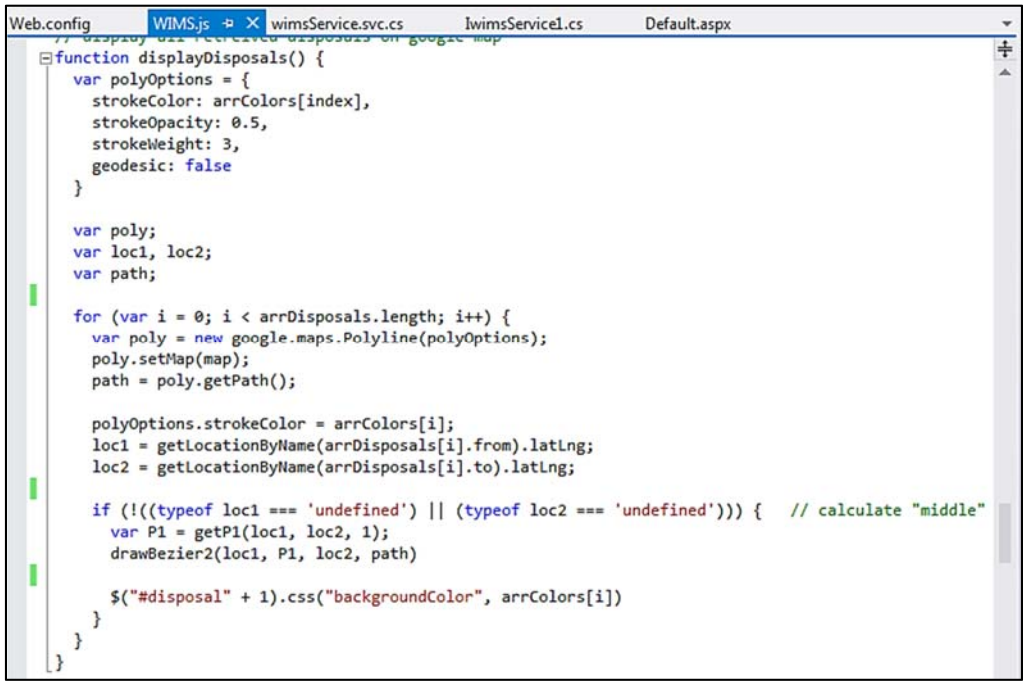


**Figure 26.** The *displayDisposals* function.

# CHAPTER 6: ALGORITHM

## 6.1 Bézier curve

Bézier curve was introduced in the early 1960's by Dr. Pierre Bézier who used this curve formulation for shape design in the Renault Company [38]. His main aim was to simplify curves used in shape design for designers and artists who were not very fond of mathematics [38]. According to Sederberg, "the beauty of the Bézier representation is that a Bézier curve mimics the shape of its control polygon. A Bézier curve passes through its first and last control points, and is tangent to the control polygon at those endpoints. An artist can quickly master the process of designing shapes using Bézier curves by moving the control points……." [39].

A sequence of Bézier curves can be used to create complex shapes, for example, all PostScript fonts are defined using the Bézier curves. PhotoShop use Bézier curves for their "paths", and almost all vector drawing programs like Flash, Illustrator or InkScape use the same type of curves [40]. Bézier curves run from some start point to some end point, and their curvature is influenced by one or more control points [40].

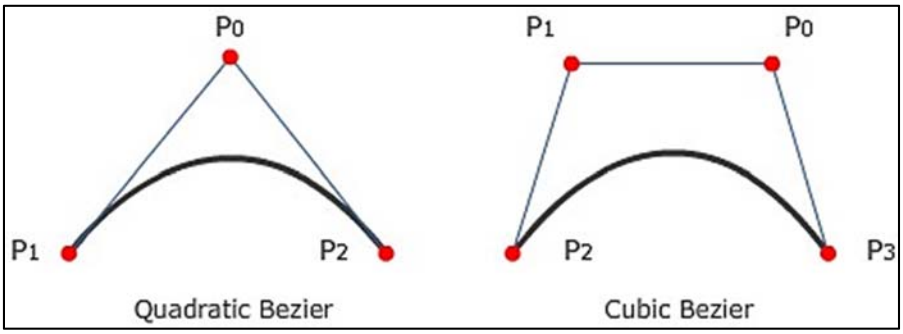


**Figure 27.** Example of quadratic and cubic Bézier curves.

"Bézier curves uses same base function in its dimensions and is a parametric function [40]. The base function for Bézier curves is following:

$$Bézier(n,t) = \sum_{i=0}^{n} \binom{n}{i} . \ (1-t)^{n-i} \, . t^{i}$$

Binomial term: $\binom{n}{i}$ Polynomial term: $(1-t)^{n-i} \, . t^{i}$"

In this study, quadratic Bézier curves were used (Figure 27). According to Li and Xue [41], "a quadratic Bézier curve is the path traced by the function B(t), given points $P_0$, $P_1$, and $P_2$,

$$B(t) = (1-t)[(1-t)P_0 + tP_1] + t\,[(1-t)P_1 + tP_2]\,, t \ \in [0,1]$$

which can be interpreted as the linear interpolant of corresponding points on the linear Bézier curves from $P_0$ to $P_1$ and from $P_1$ to $P_2$ respectively. Rearranging the preceding equation yields:

$$B(t) = \ (1-t)^2 \, P_0 + 2(1-t)tP_1 + \ t^2 P_2 \,, t \ \in [0,1]$$

The derivative of the Bézier curve with respect to t is

$$B'(t) = 2(1-t)(P_1 - \ P_0) + 2t(P_2 - \ P_1)$$

from which it can be concluded that the tangents to the curve at $P_0$ and $P_2$ intersect at $P_1$. As t increases from 0 to 1, the curve departs from $P_0$ in the direction of $P_1$, then bends to arrive at $P_2$ from the direction of $P_1$.

The second derivative of the Bézier curve with respect to t is

$$B''(t) = 2(P_2 - 2P_1 + \ P_0)$$ [41]"

**6.2 Studies Which Used Bézier Curves and Other Related Studies**

Bézier curves are used frequently in several studies to show geo location and similar information. For example, Wong et al. [42] proposed a comprehensive framework

for the geolocalization of Internet hosts based solely on network measurements. They created a system of constraints from network measurements and estimated the region bounded by Bézier curves in which target resides. Past approaches to Internet hosts geolocalization relied only on information where a node may be located. Wong et al.'s approach used the same techniques with the addition of negative information that specified where the specific node cannot be located. In another study, Tremblay et al. [43] used piecewise cubic Bézier curves along recorded tracks to interpolate animal tracking information in a fluid setting. They also provided guidelines to choose the best track algorithm for different types of marine species [43].

Kono et al., [44] provided integrated pathway maps implemented as a scalable map using the Google Maps API. It showed genes, enzymes, and metabolites on a map with search features, and enabled easy exchange of pathway data. Kaspar [45] used cubic Bézier curves for geometric transformations of text characters. Kasper developed a tool for representing curved characters and performing geometric transformations on different letters using matrices [45]. Kingre [46] proposed a new meta-modeling technique of simulation output by using Bézier curves and patches. Bézier curve was used to output modeling of univariate and bivariate outputs, and this new meta-modeling technique was compared to other existing similar techniques.

In another study, Carpatorea [47] developed interactive tool for rapid exploration of diverse traffic scenarios using cubic Bézier curves to depict the roads and vehicle trajectory. Eckert [48] did a study on intelligent support for knitwear design where she used Bézier curves to model the curves of cutting patterns. Liu [49] used Bézier curves to model the trajectories on the Digital Bibliographic Library Browser (DBLP) map to

represent the change of the research topics of an individual researcher over time. Hence, Bézier curves are used in different fields of studies.

### 6.3 Description of the Developed Algorithm

Quadratic Bézier curve can be described as a curve from point $P_1$ to point $P_2$, determined by midpoint $P_0$ as in Figure 27. In the algorithm for this study, the midpoint always has the same distance from $P_1$ and $P_2$, meaning it is centered. Key to the algorithm is the distance from $P_1$ and $P_2$ to $P_0$. The bigger the distance, the bigger the curvature of polyline. This translates to moving point $P_0$ up and down. Points $P_0$ and $P_2$ (waste source and destination) was shown on the Google Maps using geographic coordinate system. A geographic coordinate system is defined as, "a method for describing the position of a geographic location on the earth's surface using spherical measures of latitude and longitude. These are measures of the angles (in degrees) from the center of the earth to a point on the earth's surface when the earth is modeled as a sphere. When using a spheroid (ellipsoid), latitude is measured extending a line perpendicular to the earth's surface to the equatorial plane. Except at the equator or a pole, this line will not intersect the center of the earth" [50]. Google Maps are 2-D maps, and this was used for latitude and longitude to draw waste source and destination points on the map. "The latitude of a point on the Earth's surface is the angle subtended at the Earth's center by the arc along the meridian passing through the point and measured from the equator to the point. The range of latitude angles is from $0^0$ to $90^0$ North and $0^0$ to $90^0$ South" [51]. "The longitude of a point on the Earth's surface is the angle subtended at the Earth's center by the arc along the equator measured East or West of the prime meridian to the meridian passing through

the point. The range of longitude angles to cover on all points on the Earth's surface is thus $0^0$ and $180^0$ East of the meridian and $0^0$ to $180^0$ West of the prime meridian" [51].

In the algorithm for the present study, 2D Cartesian coordinate system was used. "A Cartesian coordinate system is the unique coordinate system in which the set of unit vectors at different points in space are equal. In polar coordinates, the unit vectors at two different points are not equal because they point in different directions" [52]. The sample coordinate system is shown in Figure 28.

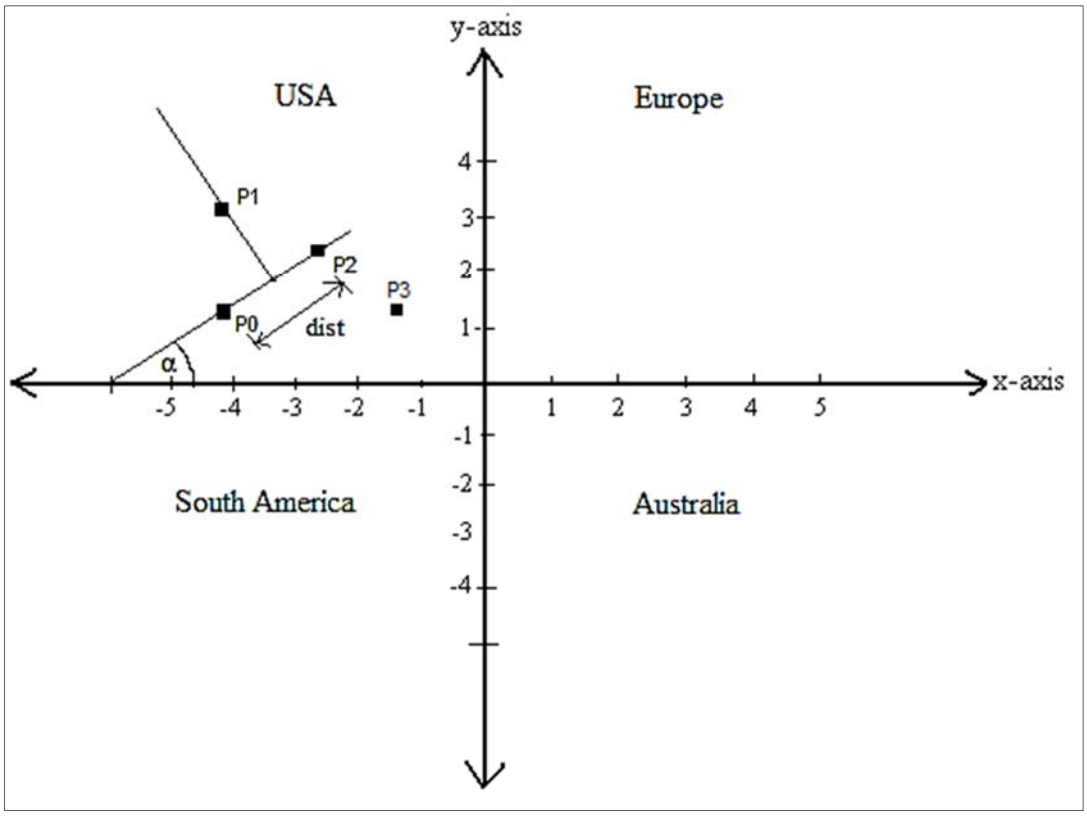


**Figure 28.** 2-D coordinate system.

X-axis shows a longitude, and Y-axis depicts a latitude of a point. "The axes of a two-dimensional Cartesian system divide the plane into four infinite regions, called quadrants, each bounded by two half-axes. These are often numbered from 1st to 4th and

denoted by Roman numerals: I (where the signs of the two coordinates are I (+,+), II (−,+), III (−,−), and IV (+,−)" [53]. If map of Earth is observed, the USA will be always in the II quadrant of 2-D coordinate system. Therefore, all the points (waste sources and destinations) will be drawn in the mentioned quadrant. There will always be negative longitude on X-axis and positive latitude on Y-axis. $P_0$ and $P_2$ are waste source and destination, and $P_1$ is the midpoint used for Bézier curve (Figure 28). The algorithm will first determine the $P_1$ point, and this was done in function *getP1(P0, P2, n)*. Following is the JavaScript function that calculates midpoint's coordinates:

```
function getP1(P0, P2, n) {
  var alpha = Math.atan2(P2.lat() - P0.lat(), P2.lng() - P0.lng());
  var rand = (Math.random() - .5) * randLevel / 100;

  dist = Math.sqrt((P0.lat() - P2.lat()) * (P0.lat() - P2.lat()) + (P0.lng() - P2.lng()) * (P0.lng()

  if (isUpOrDown(P0, P2) > 0) {
    return new google.maps.LatLng(
        (P0.lat() + P2.lat()) / 2 - Math.cos(alpha) * n * Math.pow(dist, bezierPow) / bezierDiv,
        (P0.lng() + P2.lng()) / 2 + Math.sin(alpha) * n * Math.pow(dist, bezierPow) / bezierDiv);

  } else {
    return new google.maps.LatLng(
        (P0.lat() + P2.lat()) / 2 + Math.cos(alpha) * n * Math.pow(dist, bezierPow) / bezierDiv,
        (P0.lng() + P2.lng()) / 2 - Math.sin(alpha) * n * Math.pow(dist, bezierPow) / bezierDiv);

    ;
  }
}
```

To find out middle point, the algorithm will first calculate the distance from $P_0$ to $P_2$ (and store it into *dist* variable) and angle between $P_0$ and $P_2$ (*alpha* variable in the JavaScript code above) (Figure 28). The angle $\alpha$ and distance *dist* are depicted in the JavaScript function above. To calculate distance and angle, standard mathematical formulas were used. "The Euclidean distance between two points of the plane with Cartesian coordinates (x1, y1) and (x2, y2) is

$$dist = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

This is the Cartesian version of Pythagoras' theorem" [53]. The angle is calculated by means of the inverse function of tangent (arctan) of the $P_0$-$P_2$ line's slope. "The slope of a line in the plane containing the x and y axes is generally represented by the letter m, and is defined as the change in the y coordinate divided by the corresponding change in the x coordinate, between two distinct points on the line" [54]. Then the following functions was used to determine the latitude and longitude of midpoint $P_1$:

*P1.lat = (P0.lat() + P2.lat()) / 2 - Math.cos(alpha) * n * Math.pow(dist, bezierPow) / bezierDiv*

*P1. Lng = (P0.lng() + P2.lng()) / 2 + Math.sin(alpha) * n * Math.pow(dist, bezierPow) / bezierDiv);*

This will give the midpoint that will have bigger curvature based on the distance of points. For this purpose exponential function was used *(Math.pow(dist,bezierPow) / bezierDiv)*. Variable *dist* is distance from $P_0$ to $P_2$, and variables *bezierPow* and *bezierDiv* are constant numbers 200 and 2.2, respectively. For these variables, polylines will never intersect. On the contrary, if distance from $P_0$ to $P_2$ were used instead of the shown exponential function, there would be intersections among Bézier curves. Consider figure 28, where there are 3 points, $P_0$, $P_2$, and $P_3$ – $P_0$ is source location, and $P_2$ and $P_3$ are destination location. Here, $P_3$ has greater latitude and longitude or lesser latitude and longitude value than point $P_0$ and distance between $P_0$ and $P_3$ is greater than $P_0$ and $P_2$. In such a scenario there are chances of intersection between the curves (curves from $P_0$ to $P_2$ and curve from $P_0$ to $P_3$). During these situations, the algorithm will check (in function

*isUpOrDown*) whether point $P_2/P_3$ has greater or smaller longitude than point $P_0$. If $P_3$ has smaller longitude than $P_0$, algorithm will change sign and midpoint will displace to opposite side. Point $P_0$ will always be either destination or source for all polylines. Once $P_1$, the "middle" point, is identified, the next step is to determine other points on the Bézier curve, and display that on the map.

The *bezier2* JavaScript function determines coordinates of point on a quadratic Bézier curve (Figure 29). This function will calculate points on curve based on $P_0$, $P_2$, $P_1$ and interval t. Interval t has a value between 0 and 1. The mathematical definition of the quadratic Bézier curve is described above, and here I implemented the formula to form a quadratic Bézier curve.

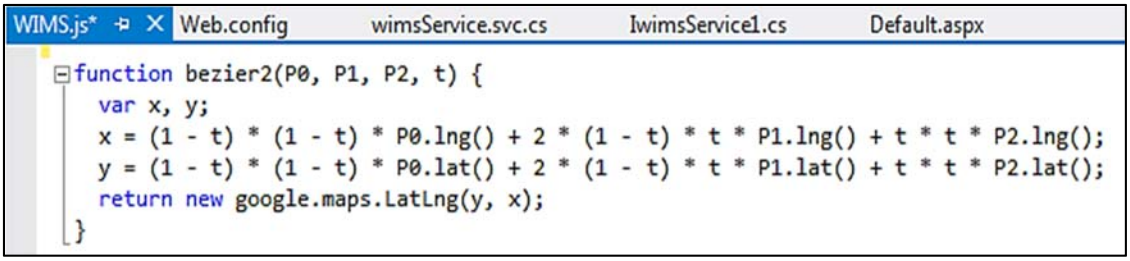


**Figure 29.** The *bezier2* JavaScript function.

Next, Bézier curve was drawn on the map by using *drawBezier2* JavaScript function depicted above (Figure 30). Polylines from Google Maps API were used to draw the required quadratic curve. The *displayDisposals* function in the JavaScript file will draw all Bézier curves on the Google Map. All locations will be enumerated, $P_1$ will be calculated, and *drawBezier2* function will be called to draw polyline. The *displayLocations* function was used to show location markers, names and quantities on the map.

WIMS.js | Web.config | wimsService.svc.cs | IwimsService1.cs | Def

```
function drawBezier2(P0, P1, P2, path) {
  var B, midLatLng;
  for (var t = 0; t < 1 + 1 / precision; t = t + 1 / precision) {
    B = bezier2(P0, P1, P2, t);
    midLatLng = new google.maps.LatLng(B.lat(), B.lng());
    path.push(midLatLng);
  }
}
```

**Figure 30.** The *drawBezier2* function.

Figure 31 and Figure 32 shows client side of application without and with algorithm for comparison.

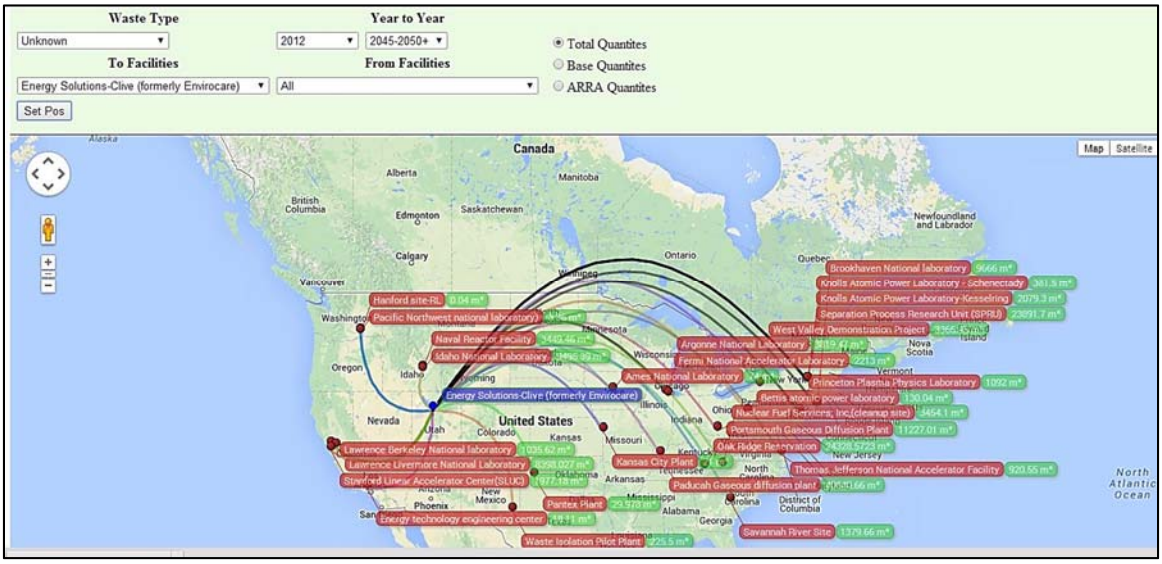


**Figure 31.** Client side of the application without using the algorithm.

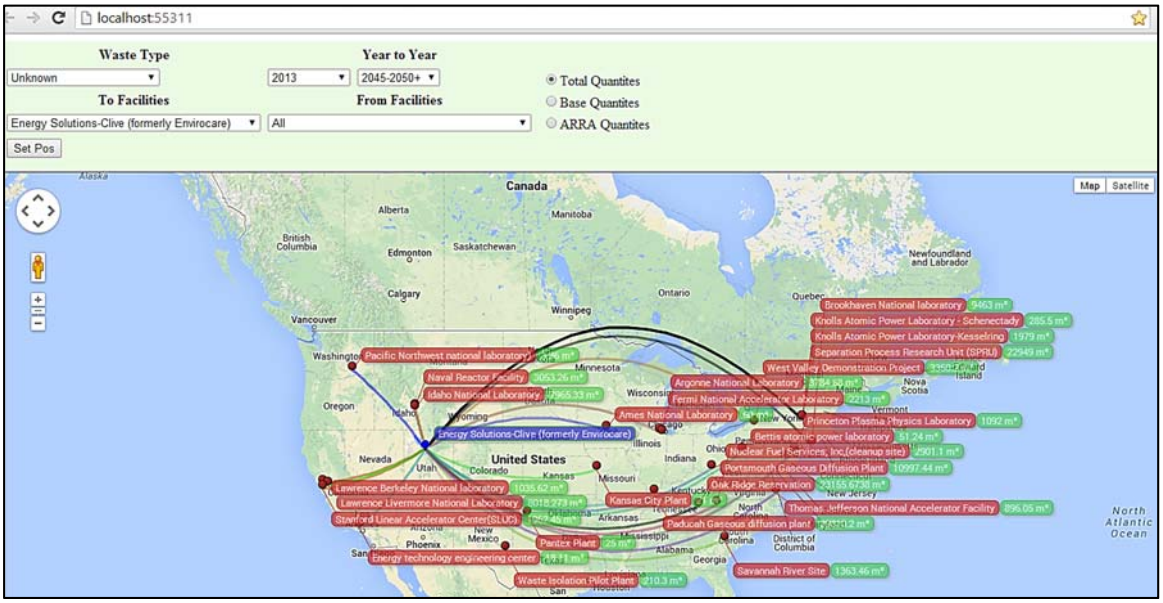


**Figure 32.** Client side of the application with the algorithm.

### 6.4 Comparison of Bézier Curves with other Parametric Curves

In addition to quadratic Bézier curves, other splines such as B-Spline, Hermite polynomials and Catmull-Rom splines could be used for developing curves in the present application [55]. However, the Bézier curve is simpler and the most commonly used curve in computer graphics, and it is easy to implement in many computer programming languages. Another advantage of using Bézier curves is that it is easy to find intersecting curves. Intersecting curves can be found by: Bézier clipping; and recursively (De Casteljua) subdividing curve to create polyline [39].

Compared to Bézier curve, B-spline curve uses blending function to create polynomial curves which passes through multiple points. B-spline curve are parametric, single and smooth. It is difficult to create Bézier curves of same smoothness. However, by combining several bits of Bézier curves smoothness similar to B-spline curve can be achieved. In addition, B-spline curves have multiple control points, even though it do not

interpolate these control points, and programmer has better control and flexibility for adjusting curvatures while designing sharp bends and corners. An important difference between B-spline curve and Bézier curve is that in B-spline curve, degree of the polynomial could be designated independent of the points of control. In B-spline curve, if a vertex is displaced, only a small section of the curve will be changed and remaining of the curve will be unaffected. Hence B-spline curve is said to have better local control than Bézier curve [39, 56].

Another alternative to Bézier curve is the Hermite polynomials. An important advantage of Hermite polynomials compared to Bézier curve is that, in Hermite polynomials, curve is defined based on starting and ending points and velocities than control points. In contrast to Bézier curve, Hermite polynomial curve is not a weighted average. This is because, the sum of Hermite polynomials is always less than 1 [56].

Another alternative to Bézier curve is the Catmull-Rom curves which are commonly used in animation. The biggest advantage of Catmull-Rom curves is that the curves are smooth and interpolate control points, similar to B-spline curve. This gives the programmer greater control on the curve. Again, similar to B-spline curve, Catmull-Rom curves have better local control. Hence, only a small segment of the curve will change if the vertex is moved. These curves have explicit piecewise polynomial representation. This will allow users to manipulate it easily [57].

### 6.5. Application of Developed Algorithm on Other Map Application

The algorithm developed in this study could be used in other map applications where there is: one source and multiple destinations; or multiple sources and one destination; or multiple paths that do not overlap. One such example is displaying airline

paths to multiple destinations from a chosen airport. This algorithm was used to demonstrate its application in airline tracking, using data obtained from open-source flight maps web site http://openflights.org/. Figure 33 shows the flight pathways of airlines from Miami, Florida (MIA) to multiple destinations. The data from http://openflights.org/ was first exported to Google's KLM format and imported into Google Maps. Data on latitudes and longitudes of airports was then downloaded from http://openflights.org/data.html. Then, to test the developed algorithm, a new ASPX page was added and called it Airports.aspx and put sample airport locations to demonstrate that the developed algorithm could work with other data as well (Figure 34).

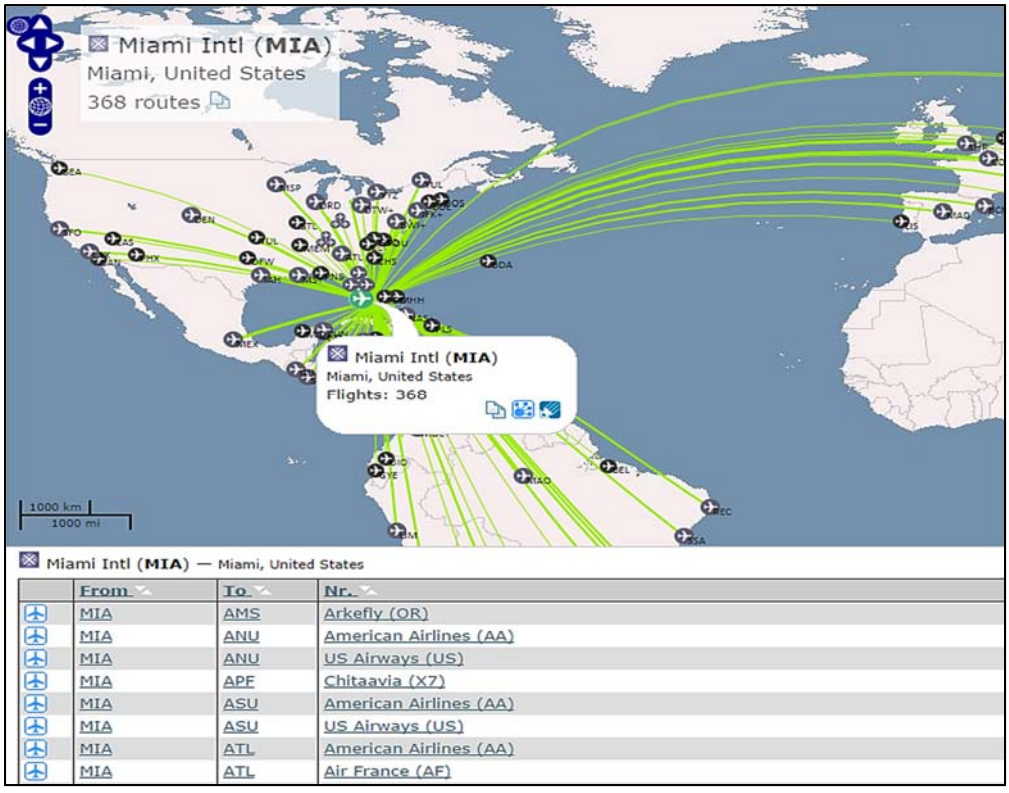


**Figure 33.** Flight map from Miami International Airport (MIA) to all possible destinations on http://openflights.org.

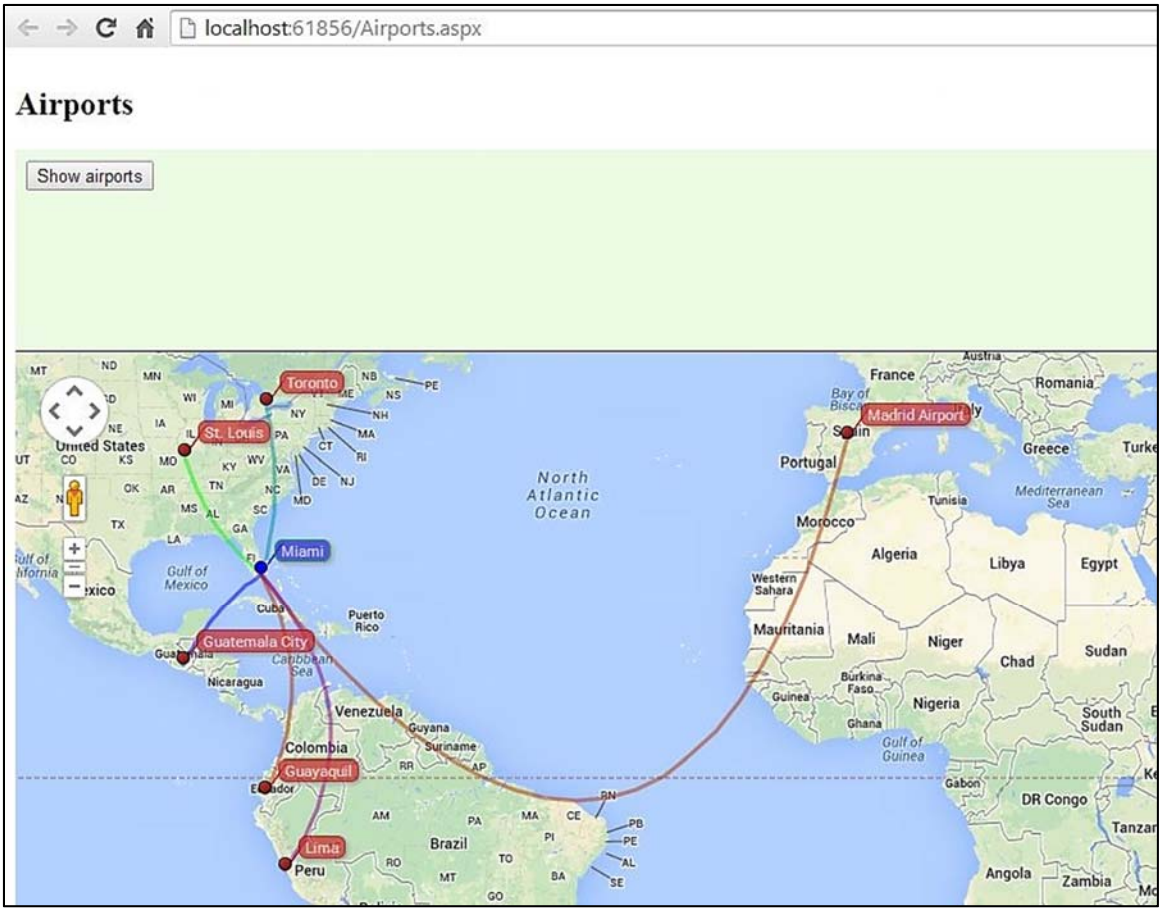


**Figure 34.** Flight map from Miami International Airport to all possible destinations with developed algorithm.

### 6.6. Computational Aspects the Algorithm

In this study, a direct implementation of Bézier curves through Bernstein polynomials is used.

$$Bézier(n,t) = \sum_{i=0}^{n} \binom{n}{i} . \ (1-t)^{n-i} \ . t^{i}$$

Binomial term: $\binom{n}{i}$ Polynomial term: $(1-t)^{n-i} \ . t^{i}$,,

Bernstein expansion, due to its binomial character, has an asymptotic complexity

of O(n^2). More precisely, n(n+1) multiplications and (n+1) additions are required to compute the location of a single point on the Bézier curve, where ‘n’ is the order of the Bezier curve.

Another approach commonly used to compute Bézier curves is De Casteljau's algorithm. In this approach, the original Bézier curve is recursively divided into smaller curves, defining a new set of control points at each division in such a way that the points on the sub-curve coincides exactly with the points on the original curve. The computational complexity of De Casteljau's algorithm is known to be n(n+1) multiplications and n(n+1)/2 additions per point [58]. From the above description, the asymptotic computational complexity (/large N) remains constant while the local computational complexity (small N) slightly increases. Moreover, De Casteljau's algorithm is more complex to implement than the Bernstein expansion. However, De Castaljau's algorithm, in spite of its complexity, is preferred in certain applications because of its favorable numerical stability.

As only a quadratic form of Bézier curves is used in this study, both direct Bernstein implementation and De Casteljau's algorithm are computationally cheap (few number of arithmetic operations per point). Numerical stability is defined as the accuracy of a series of computerized calculations, and an analysis of numerical stability studies the bounds on the errors accumulated in computed values resulting from the finite representation of real numbers in computer memory. As far as Bézier curves is concerned, computational errors may translate to imperfections on the generated curve surfaces, which can be significant in applications such as industrial design and sculpture.

In such applications, De Casteljau's algorithm is preferred due to its numerical accuracy. In the context of this study, we do not need such a precision.

Many curves depart from a single location or arrive at a single location based on the number of nuclear waste streams. The resulting Bézier curve density may become high at the source or destination locations depending on the number of incoming or outgoing curves. This caused a constraint on the zoom-in quality at such points. We needed the program to resolve and unambiguously display paths - even where they come too close - when displays are zoomed into a certain extent. The algorithm in this study handles this in two ways and avoids the implementation complexity of De Casteljau's algorithm:

- The set of incoming paths to or outgoing paths from a location are divided into two subset, as north-wise paths and south-wise paths. Although this is primarily done to avoid intersection, this also serves to decrease point density at otherwise highly dense end-points.
- The computational advantage of using only a second order curve increased curve resolution. This is represented in the code by the variable called *precision*, which controls the number of points on a computed curve. In other words, since the algorithm is computationally cheap, increased the density of points on the curve (precision) can cope with dense regions (multiple ‘from’ and ‘to’ locations)

The algorithm used for this study has several advantages: it uses congestion avoidance to prevent intersections; it is visually more attractive; it is simple to implement, using straightforward Bernstein base functions; it is computationally efficient, few computations per point; and it is asymptotically quadratic, scales by the

square of N. Major disadvantage of the algorithm is that its numerical stability is low. Due to low numerical stability, the smoothness of the curves may degrade, but higher degreee of smoothness is not required in this application.

# CHAPTER 7: CONCLUSION

This study has provided a dynamic Google map for forecasting nuclear waste stream. All of the identified issues of WIMS regarding static image design were solved by developing Web 2.0 client application, WCF web services, and Google Maps JavaScript API. The study has shown that the combination of these technologies can successfully provide dynamic mapping functionality. The system presented in the thesis and the accompanied web application has fulfilled all listed research objectives of the study. Microsoft's documentation is well-organized and comprehensive. So, developing WCF web services was a relatively easy task. The study showed that relational database model (using Microsoft SQL Server) is very effective in storing data and establishing associations among waste disposal data. The most challenging part of the study was the writing of the JavaScript code and the development of algorithm to prevent intersection of polylines in the map. Since using browser's script debuggers and relevant plug-ins can be complicated, debugging JavaScript was challenging. However, this was made easy by using third-party JavaScript libraries which enable AJAX and mapping functionality. The study also showed that Google Maps API was a good choice to depict waste disposal streams on a map. The developed application prototype could enable waste managers to plan their future actions regarding volumes, categories, and issues related to forecasted waste streams.

The most important contribution of this study was the development of an algorithm that successfully eliminated overlapping or intersecting of polylines among waste sources and destinations. This feature greatly enhanced visualization of the map. For this purpose, quadratic Bézier curves connecting waste sources and destinations were

used. Bézier curve was used for this projects because of its sculpting simplicity and quick computation. Though we developed our applications based on Bézier curve, in the future, applications could be derived using other curves like B-Spline, Hermite polynomials and Catmull-Rom splines there by increasing the visibility and dispersion of polylines as well as facilitating better understanding of the waste disposal process. Additionally, exponential function was coded in JavaScript to obtain bigger curvature based on the distance of points.

Future work could include inspection of other Google Maps API functionalities to determine whether they can be used to further enhance visualization of the nuclear waste streams. For example, Google Maps API can display road map views and satellite images, and maps centered on terrain information. Furthermore, there are other free JavaScript-based map libraries such as OpenStreetMap, OpenLayers, and Yahoo! Maps. Future studies should also focus on customizing the applications of this study for smart phones and tablets because more and more people use these devices instead of computers to access the internet.

We also tried to compare the developed algorithm with De Casteljau's algorithm to assesss the efficacy of our algorithm. However, De Casteljau's algorithm has the same asymptotical complexity (they are both O(n^2)). For lower degree of the curves (in our algorithm, N=2), the number of arithmetic operations per point is very low and has about 10 arithmetic operations per point. This made the algorithms very fast and making a comparison at such a significance level is scientifically inappropriate.

Furthur exapnsion of our project includes overcoming the limitations due to extreme curve densities. This could happen when multiple locations are selected by the

user for waste disposal. This could be overcome through advanced methods like tracking the curve density at a given point and increasing the points-per-inch density at locations where curves are very dense. Andersson and Kvernes have discussed this technique to keep track of intersecting curves through second derivatives which measure curvature and estimate forward difference [59].

### 7.1 Implications of the study

Open specifications and open source software could be used to set up a GIS application for managing nuclear waste streams. Several challenges faced in the implementation of web-based applications could be understood and overcome through sequential advancement in existing methods. Our study investigated a broad array of topics concerned with the development of Google Map applications as well as managing the information more efficiently and spreading waste related information to the nuclear waste management professionals more clearly and concisely. The study also provides a dynamic Google map of the nuclear waste stream for waste managers to plan their future actions. Several other fields with one source-multiple destinations or multiple sources-one destination concepts could take advantage of our Bézier curve dependent algorithm. Examples include scheduling and routing airline (Figure 34) and shipping lines, commercial freights, mining services, oil rigs and oil refineries etc. The culmination of these technologies are virtually limitless, given the number of applications.

References


[1] A. Andrews, *Radioactive waste streams: waste classification for disposal*: DIANE Publishing, 2011.

[2] G. Peterson and D. Tonkay, "The Department of Energy's National Disposition Strategy for the Treatment and Disposal of Low Level and Mixed Low Level Waste," 2006.

[3] H. Upadhyay, W. Quintero, P. Shoffner, L. Lagos, and D. Roelant, "Waste Information Management System-2012-12114."

[4] K. Gawande and H. Jenkins-Smith, "Nuclear waste transport and residential property values: estimating the effects of perceived risks," *Journal of Environmental Economics and Management,* vol. 42, pp. 207-233, 2001.

[5] E. L. Gershey, R. C. Klein, E. Party, and A. Wilkerson, "Low-level radioactive waste," 1990.

[6] U. E. P. Agency. (2014, March 5). *Radiation Protection*. Available: http://www.epa.gov/radiation/mixed-waste/about.html

[7] W. J. Weber, R. C. Ewing, C. Catlow, T. D. De La Rubia, L. Hobbs, C. Kinoshita, *et al.*, "Radiation effects in crystalline ceramics for the immobilization of high-level nuclear waste and plutonium," *Journal of Materials Research,* vol. 13, pp. 1434-1484, 1998.

[8] R. Walton, "Radioactive Waste Treatment Technology," presented at the 19th Annual Symposium on Geological Disposal of Nuclear Waste, 1979.

[9] S. Davies, "End of the road for Yucca Mountain," *Engineering & Technology,* vol. 5, pp. 46-48, 2010.

[10] H. Upadhyay, W. Quintero, P. Shoffner, L. Lagos, and D. Roelant, "Waste Information Management System-2011–11303."

[11] M. P. Papazoglou and W.-J. Van Den Heuvel, "Service oriented architectures: approaches, technologies and research issues," *The VLDB journal,* vol. 16, pp. 389-415, 2007.

[12] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, 2003, pp. 3-12.

[13] M. Bichier and K.-J. Lin, "Service-oriented computing," *Computer,* vol. 39, pp. 99-101, 2006.

[14] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. big'web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 805-814.

[15] L. Richardson and S. Ruby, *RESTful web services*: " O'Reilly Media, Inc.", 2008.

[16] C. Pautasso, "RESTful Web service composition with BPEL for REST," *Data & Knowledge Engineering,* vol. 68, pp. 851-866, 2009.

[17] J. Flanders, "More On REST," *MSDN Magazine, Julho de,* 2009.

[18] C. McMurtry, M. Mercuri, N. Watling, and M. Winkler, *Windows Communication Foundation Unleashed (Wcf)(Unleashed)*: Sams, 2007.

[19] A. Mackey, "Windows Communication Foundation," in *Introducing. NET 4.0*, ed: Springer, 2010, pp. 159-173.

[20] J. Smith, *Inside microsoft windows communication foundation*: Microsoft Press Redmond, 2007.

[21] P. Cibraro, K. Claeys, F. Cozzolino, and J. Grabner, *Professional WCF 4: Windows Communication Foundation with. NET 4*: John Wiley & Sons, 2010.

[22] C. Nagel, B. Evjen, J. Glynn, K. Watson, and M. Skinner, *Professional C# 4.0 and. NET 4*: John Wiley & Sons, 2010.

[23] S. Murugesan, "Understanding Web 2.0," *IT professional,* vol. 9, pp. 34-41, 2007.

[24] J. J. Garrett, "Ajax: A new approach to web applications," 2005.

[25] D. Flanagan, *JavaScript: The definitive guide: Activate your web pages*: " O'Reilly Media, Inc.", 2011.

[26] A. Taly, "Sandboxing Untrusted JavaScript," Stanford University, 2013.

[27] S. Hu, "Online Map Service Using Google Maps API and Other JavaScript Libraries: An Open Source Method," in *Online Maps with APIs and WebServices*, ed: Springer, 2012, pp. 265-278.

[28] G. Svennerberg, M. Wade, C. Andres, S. Anglin, M. Beckner, E. Buckingham*, et al.*, *Beginning Google Maps API 3*: Springer, 2010.

[29] G. Inc. (2014, April). *Google Maps JavaScript API v3*. Available: https://developers.google.com/maps/documentation/javascript/tutorial

[30] R. Ramakrishnan, J. Gehrke, and J. Gehrke, *Database management systems* vol. 3: McGraw-Hill New York, 2003.

[31] R. Narang, *Database management systems*: PHI Learning Pvt. Ltd., 2011.

[32] P. Nielsen and U. Parui, *Microsoft SQL server 2008 bible* vol. 607: John Wiley & Sons, 2011.

[33] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study," *Caine,* vol. 2009, pp. 157-162, 2009.

[34] B. Casselman, *Mathematical illustrations: a manual of geometry and PostScript*: Cambridge University Press Cambridge, 2005.

[35] T. A. Powell, *HTML: the complete reference*: McGraw-Hill Professional, 1999.

[36] J. N. Robbins, *Learning web design: A beginner's guide to HTML, CSS, JavaScript, and web graphics*: " O'Reilly Media, Inc.", 2012.

[37] J. C. Experts, *JQuery Cookbook*: " O'Reilly Media, Inc.", 2010.

[38] G. Messina, E. Ingrà, S. Battiato, and G. Di Blasi, "Bezier curves approximation of triangularized surfaces using SVG," in *Electronic Imaging 2006*, 2006, pp. 60610O-60610O-10.

[39] T. Sederberg, *Computer-aided geometric design*: Brigham Young University, 2012.

[40] M. Kamermans. (2014, January 8). *A Primer on Bézier Curves*. Available: http://pomax.github.io/bezierinfo/

[41] X. Li and J. Xue, "Complex Quadratic Bézier Curve on Unit Circle," in *International Conference on Logistics Engineering, Management and Computer Science (LEMCS 2014)*, 2014.

[42] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts," in *NSDI*, 2007.

[43] Y. Tremblay, S. A. Shaffer, S. L. Fowler, C. E. Kuhn, B. I. McDonald, M. J. Weise*, et al.*, "Interpolation of animal tracking data in a fluid environment," *Journal of Experimental Biology,* vol. 209, pp. 128-140, 2006.

[44] N. Kono, K. Arakawa, R. Ogawa, N. Kido, K. Oshita, K. Ikegami, *et al.*, "Pathway projector: web-based zoomable pathway browser using KEGG atlas and Google Maps API," *PLoS One,* vol. 4, p. e7710, 2009.

[45] C. Kaspar. (2009, January 30). *Using Bezier Curves for Geometric Transformations*. Available: http://www.math.iastate.edu/thesisarchive/MSM/KasparCCF09.pdf

[46] H. Kingre, "Bezier Curve For Metamodeling Of Simulation Output," Louisiana State University, 2004.

[47] I. N. Carpatorea, "A graphical traffic scenario editing and evaluation software," 2012.

[48] C. Eckert, "Intelligent support for knitwear design," The Open University, 1997.

[49] H. Liu, *Dynamic concept cartography for social networks*: University of Sydney, 2011.

[50] A. Resources. (2014, February 6). *Geographic Coordinate System*. Available: http://resources.arcgis.com/en/help/main/10.1/index.html#//003r0000002t000000

[51] R. P. Collinson, *Introduction to avionics systems*: Springer, 2011.

[52] M. I. o. Technology. (2014, February 15). *Coordinate Systems*. Available: http://web.mit.edu/8.02t/www/materials/modules/ReviewB.pdf

[53] E. Board, *The concise dictionary of mathematics*: V & S Publishers, 2012.

[54] L. M. Surhone, M. T. Timpledon, and S. F. Marseken, *Slope: Mathematics, Line (geometry), Differential Calculus, Tangent, Curve, Grade (slope), Gradient, Geography, Civil Engineering, Trigonometry, Calculus, Trigonometric Functions*: Betascript Publishing, 2010.

[55] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An introduction to splines for use in computer graphics and geometric modeling*: Morgan Kaufmann, 1987.

[56] S. R. Buss, *3D computer graphics: a mathematical introduction with OpenGL*: Cambridge University Press, 2003.

[57] M. Khan and Y. Ohno, "Compression of Temporal Video Data by Catmull-Rom Spline and Quadratic Bézier Curve Fitting," 2008.

[58] H. N. Phien and N. Dejdumrong, "Efficient algorithms for Bézier curves," *Computer Aided Geometric Design,* vol. 17, pp. 247-250, 2000.

[59] F. Andersson and B. Kvernes, "Bezier and B-spline Technology," *Umea university Sweden,* 2003.