3D Rendering of Geo-Spatial Data in XML/GML Using Python

By

Aliyu Salisu

Department of Mathematics, Faculty of Science, Ahmadu Bello University, Zaria

June, 2012

3D Rendering of Geo-Spatial Data in XML/GML Using Python

By

Aliyu Salisu

B.Sc. Computer Science (A.B.U, Zaria)

M.Sc./SCIEN/02414/2009-10

A Thesis submitted to the School of Post Graduate Studies, Ahmadu Bello University, Zaria in partial fulfillment of the requirement for the Award of Masters Degree in Computer Science.

Department of Mathematics, Faculty of Science

Ahmadu Bello University, Zaria Nigeria

June, 2012

DECLARATION

I hereby declare that the work in this thesis titled "3D Rendering of Geo-Spatial Data in XML/GML using Python" was performed by me in the Department of Mathematics under the supervision of Prof. S.B Junaidu and Dr. S.E. Abdullahi. The information derived from the literature has been duly acknowledged in the text and list of references provided. No part of this work has been presented for another degree or diploma at any university.

ALIYU, Salisu		
Name of student	Signature	Date

CERTIFICATION

This thesis entitled "3D Rendering of Geo-Spatial Data in XML/GML using Python" meets the regulation governing the award of the degree of M.Sc. Computer Science of the Ahmadu Bello University Zaria, and is approved for its contribution to knowledge and literary presentation.

Professor S.B. Junaidu	Date
Major Supervisor	
Dr. S.E Abdullahi	Date
Minor Supervisor	
Dr. B. Sani	Date
H.O.D Mathematics	
Professor A. A. Joshua	Date
Dean, Postgraduate School	

DEDICATION

This thesis is dedicated to my beloved parents in the name of Muhammad Raji Aliyu and Kharietu Aliyu and my entire family members for their support.

ACKNOWLEDGEMENT

First of all I will like to acknowledge the effort of my humble supervisor (major) Professor S.B. Junaidu for sacrificing his efforts and time to see me through the completion of this thesis and my minor supervisor Dr. S.E. Abdullahi for his contributions.

I also acknowledge the effort of the entire lecturers of the department of mathematics for their academic and intellectual support towards the success of this thesis.

A special acknowledgment goes to my parents, my sister Dr. Hadiza and my wife for their financial and courageous support throughout my entire study.

I cannot forget mentioning some of my colleagues who contributed in one way or another towards the successful completion of this thesis. These people include Mal. Muhammad Abdullahi, Mal. Abdullahi Buhari, Mal. Mustapha Aminu Bagiwa, Mal. Salihu Idi Dishing, Mal. Abdulrazak (RSK), Mal. Muhammad Abubakar Baba'abu and my entire class members.

Finally, a special acknowledgment also goes to the University Management for their financial support (tuition fee) towards the successful completion of this thesis.

ABSTRACT

Geographic information (also called geo-information, spatial information or geospatial information) plays an increasingly important role in our society. Location-based services, applications for urban planning, disaster management systems rely on up-to-date geospatial data. Geo-spatial data as a major component of every Geographic Information System has been facing many challenges in its use including: portability, maintainability, interoperability, accessibility and unavailability of digital dataset. This work proposes a framework for the 3D rendering of Geo-spatial data in XML/GML which are MarkUp Languages that encodes Geo-spatial information in the Web. With our framework, we were able to extract Geo-spatial information from the satellite imagery of an area covered by the Ahmadu Bello University, Zaria, then store this information in a spatially enabled database interfaced with our Python engine which now renders this geo-spatial information in GML. The basic idea is to render this geographic information in a unique environment (the Web) that will make this data portable, accessible, maintainable and interoperable. The approach reveals interesting results as it was discovered that the framework with a little extension can be adopted to serve geographic data in other XML-base technologies capable of holding geographic information like CityGML, X3D and KML.

Table of Contents

DECLARATION iii
CERTIFICATION iv
DEDICATION v
ACKNOWLEDGEMENT vi
ABSTRACT vii
Table of Contents viii
LIST OF FIGURES x
LIST OF TABLES xi
LIST OF APPENDICES xiii
ABBREVIATIONS, DEFINITIONS, GLOSSARIES AND SYMBOLS xiv
CHAPTER ONE 2
GENERAL INTRODUCTION 2
1.1 Background of study 2
1.2 Research Motivation and Goals 6
1.3 Research Questions 7
1.4 Research Objectives 8
1.5 Methodology 8
1.6 Contributions to Knowledge 9
CHAPTER TWO 10
LITERATURE REVIEW 10
2.0 Geo-Spatial Data 10
2.1 Geographic Coordinate System 11
2.3 Geo-Spatial Data Types (SDT) 12
2.4 Geo-Spatial Data File Format 14
2.5 Geo-Spatial Data Collection 15
2.6 A Major Problem Associated With Geo-Spatial Data Usage 16
2.7 The Web as a Major Platform to Enhance Geo-Spatial Data Interoperability 16
2.7.1 Geography Markup Language (GML) 17
2.7.2 Mechanisms of GML for Data Interoperability 18
2.7.3 Generating GML Data 23
2.7.4 Visualizing GML Data 24

2.8 Review of 3D GIS Implementation with the Web's Markup Language 26
2.9 Relationship between our proposed framework and the prototype reviews..... 30
CHAPTER THREE 31
SYSTEM DEVELOPMENT..... 31
3.0 System Requirements..... 31
3.1 Geographic Data in GML..... 36
3.0.1 GML Features..... **Error! Bookmark not defined.**
3.0.1 Encoding Geographic Information with GML ..... **Error! Bookmark not defined.**
3.0.2 Structures of GML Documents..... 40
3.2 Proposed Framework..... 36
CHAPTER FOUR 48
SYSTEM IMPLEMENTATION 48
4.0 Data Collection 48
4.1 Data Storage..... 52
4.2 Algorithm used for implementation of the Python Engine 54
4.3 Visualization of the Generated GML 55
CHAPTER FIVE..... 56
SUMMARY AND FUTURE WORK..... 56
5.0 Summary 56
5.2 Future Work 57
REFERENCES..... 59

LIST OF FIGURES

Figure 1: Major classes of projections 12
Figure 2 A Point 12
Figure 3 A Polyline 13
Figure 4: A Polygon 14
Figure 5 Geo-enabling the Web by transporting and sharing different format geospatial data using GML 20
Figure 6: VRML / JAVA web GIS architecture by (Morcrette, 1999) 27
Figure 7: VRML / JAVA 3D web GIS prototype architecture by (Kim et al, 1998) 28
Figure 8: Google Earth 31
Figure 9 ArcGIS 32
Figure 10 PostgreSQL Database 33
Figure 11: A FRAMEWORK FOR RENDERING 3D GEOSPATIAL DATA IN GML 37
Figure 12: 2D Digitized Image of ABU, Zaria 49
Figure 13: 2D to 3D Conversion of Digitized Image of ABU, Zaria 52
Figure 14: Attribute Table showing information of the Digitized Image of ABU, Zaria. 50
Figure 15: PostgreSQL Database with Stored information from the Digitized Image of ABU, Zaria 53
Figure 16:3D Visualization of the generated GML Data. 55

LIST OF TABLES

Table 1: GML Document Structure..... **Error! Bookmark not defined.**

Table 2: Java conversion code ..... **Error! Bookmark not defined.**
Table 3: Python conversion code ..... **Error! Bookmark not defined.**
Table 4: Java code to print hello world 34
Table 5: Python code to print hello world 34
Table 6: Database connection with Python's Psycopg2 module..... 35
Table 7: cursor creation to enable database manipulation 36

LIST OF PLATE

PLATE 1: Satellite Imagery of Ahmadu Bello University, Zaria from Google Earth 38

LIST OF APPENDICES

Appendix 1……..55

ABBREVIATIONS, DEFINITIONS, GLOSSARIES AND SYMBOLS

CML-Chemical MarkUp Language

DEM-Digital Elevation Model

EAI-External Authoring Interface

GML-Geography MarkUp Language

GIS-Geographic Information System

CAD-Computer Aided Design

GDF-Geographic Data File

SDTS-Spatial Data Transfer Standard

KML-Keyhole MarkUp Language

GPS-Global Positioning System

SDT-Spatial Data Type

SDC-Smart Data Compression

TIN-Triangular Irregular Network

HTML-Hypertext Markup Language

JDBC- Java Database Connectivity

MusicML-Music MarkUp Language

MathML-Math Markup Language

ODBC-Open Database Connectivity

OGC-Open Geospatial Consortium

VB-Visual Basic

QGIS-Quantum GIS

RDF- Resource Development Framework

SVG-Scalable Vector Graphics

SQL-Standard Query Language

VRML-Virtual Reality MarkUp Language

WFS-Web Feature Services

X3D-Extensible 3D

XML- Extensible MarkUp Language

CHAPTER ONE

GENERAL INTRODUCTION

This chapter discusses the introductory part of the thesis which includes background of the study, research motivations and goals, the research questions for which the thesis attempts to answer, the methodology that is used to answer those questions and finally the summary of the thesis contribution to knowledge.

1.1 Background of study

The World Wide Web has found its application in all spheres of life and studies. Geographic information management and analysis is not an exception to this. Geography is primarily concerned with location-based phenomena in both manmade and natural realms, why things are located where they are, for example, why people inhabit in flood-prone areas, how they are related to historical and present activities, and how to make better plans for the future based on experience gained in the past. (Alan, 2003)

The essence of geography resides in location related problem solving and decision making. Let us take a look at a few examples.

In e-government, location intelligence is used to plan growth, improve public services and share information with citizens. Governments can employ geo-technology to support public works decisions such as highway and sewer planning. In municipalities, citizens now have a centralized portal where criminal intelligence information is shared by law enforcement at the local, state and federal levels-giving law enforcement agencies the insight needed to solve and prevent crimes.

Opening a new store or branch location can cost millions of naira, with payback often calculated in years. Companies can now analyze market demographics, competition and consumer buying

habits across alternative geographies in order to predict events well into the future. This is especially important in times of economic uncertainty, when many companies are deciding whether or not to close or relocate stores or branch locations.

In facilities management-locating underground pipes and cables, planning facility maintenance, telecommunication network services, energy use tracking and planning. In environmental and natural resources management-suitable study for agricultural cropping, management of forests, agricultural lands, water resources , environmental impact analysis, disaster management and waste facility site location. In short, geography helps governments, business, organizations and individuals make better decisions.

Geographic Information has become a key issue in location related problem solving and effective decision making. The field of Geographic Information System (GIS) evolved since the 1960's with the basic aim of improving the efficiency and effectiveness in decision making. A GIS is a system using computers and software to collect, retrieves, store, manipulate, transform, analyze and display data that describe the physical and logical properties of the geographic world. (Hilary M. H., 1994)

The problem of geospatial data portability, maintainability and interoperability and information access has been a major problem in the GIS industry. According to the Open Geo-Spatial Consortium (OGC), the following issues must be solved and promoted.

- **Open data policy:** GIS data and information should be accessible by any user, freely or at inexpensive costs and without restriction.
- **Standardization**: Standards for data format and structure should be developed to enable transfer and exchange of geospatial data.

- **Data/Information sharing**: In order to save cost and time for digitization, data sharing should be promoted. In order to foster operational use of geospatial data, information and experience should be shared among users.
- **Multi-disciplinary approach**: Because GIS is a multi-disciplinary science, scientists, engineers, technicians and administrators of different fields of study should cooperate with each other to achieve the common goals.
- **Interoperable procedures**: GIS should be interwoven with other procedures such as CAD, computer graphics, image processing, DEM etc.

With the rapid development in GIS (Geographic Information System) and its applications, data sharing and acquisition is still a big problem for the development of GIS applications. Not that data are not available, there is a huge amount of geographical data stored in different places and in different formats, but data reuse for new applications and data sharing are daunting tasks because of the heterogeneity of existing systems in terms of data modeling concepts, data encoding techniques and storage structures, etc. (Devogele *et al*, 1998).

Currently, several commercial desktop GIS software systems dominate the geographical information (GI) industry, such as ESRI ArcInfo and Arc View, Smallworld GIS, Intergraph, GeoMedia, MapInfo professional, Clark Lab Idrisi, etc. It is unlikely that all GIS applications will use the same software (Tarnoff, 1998). Different vendors have their own proprietary software designs, data models and database storage structures. Thus, geographic information system applications cannot communicate without data conversion. In order to exchange information and share computational geo-data resources among heterogeneous systems, conversion tools have to be developed to transform data from one format into another.

Furthermore, these diverse desktop GIS database structures make remote data exchange and sharing more difficult because of limited accessibility and required data conversion.

The development of the World Wide Web creates a unique environment for sharing geospatial data. Users can use the World Wide Web to download data for viewing, analysis or manipulation. Many of commercial Internet GIS programs, such as ESRI's MapObject IMS and ArcIMS, AutoDesk's MapGuide, Intergraph's Geomedia WebMap, MapInfo's MapXtreme, GE SmallWorld's Internet Application Server and ER Mapper's Image Web Server, are developed to offer better tools for data sharing over the Web. But like the desktop GIS software these Internet GIS programs also have the problems of proprietary software designs, data models and data storage structures. Sharing of data, facilitated by the advances in network technologies, is hampered by the incompatibility of the variety of data models and formats used at different sites (Choicki, 1999).

The interoperability of data from heterogeneous sources is extremely important in the context of geographical applications, because there exist large amounts of spatial data of different geographical formats and there are increased demands for re-use of these existing spatial data.

How to realize the goal of data interoperability? There are two approaches to data interoperability—database integration and standardization (Devogele *et al*, 1998). The second approach to interoperability i.e. standardization is our major concern for this thesis. The definition of standard data modeling and manipulation of features provide a reference point which facilitates data exchange among heterogeneous systems.

In the past, several useful standards have been developed to facilitate data exchange. Among them, Geographic Data File (GDF) and the Spatial Data Transfer Standard (SDTS) are widely used and accepted. Currently both GDF and SDTS were not so widely used as originally

anticipated. The creation of a new standard data exchange format--Geography Markup Language (GML)--represents another important step taken by the geospatial community towards data interoperability. GML is developed as Data Exchange Standards Interface by Open GIS Consortium (OGC) to achieve data interoperability and reduce costly geographic data conversions between different systems.

With the development of the World Wide Web Consortium's XML (Extensible Markup Language), the creation of Geography Markup Language Implementation Specification by OGC represents a significant step in the development of interoperable architectures for the use of spatial information between different applications. GML holds promise to support mapping from a wide variety of sources and enable sharing of geospatial data for on-line information exchanges.

1.2 Research Motivation and Goals

Virtually all the information required for planning and national development is spatial in nature. That is they are referenced to a given geographic location on the earth. Geospatial Information is needed for social, political, economic and physical development and for environmental and natural resources management. For easy accessibility, such information must be produced in coordinated and consistent manner and distributed harmoniously to all who need them. Geospatially encoded information in GML format provides the platform for handling spatially referenced information.

Nigeria's National Geo-spatial Data Infrastructure which is responsible for the collection of Policies and standards for data acquisition, integration and sharing is faced with problems of its full implementation due to non-existence of digital geospatial datasets in the country (i.e. most of

the fundamental and thematic datasets in the country are still in analogue form). (Igbokwe, J. I. and Ono, M.N. (2005)). We propose a simple and fast growing technology for geospatial data representation and rendering in digital format (i.e. GML).

Secondly, most common web feature services are limited to the rendering 2-D dataset (e.g.Geoserver) which is a major drawback in the application of geo-spatial data in the field of urban and regional planning, 3D visual representation of a geographic space captures a clear picture of the environment of study.

This thesis proposes a framework for the rendering of 3D geo-spatial data in GML from a sample satellite imagery of a land cover by the Ahmadu Bello University, Zaria. The framework is not limited to GML, but it can further be extended to serve geographic data in other XML based technologies capable of holding geographic information like: CityGML, KML, Collada and X3D which are XML data structures that can hold geographic information.

1.3 Research Questions

The study is set up to answer the following questions:

1. What does it take to generate digital Geo-spatial dataset?

   As data acquisition or data input of geo-spatial data in digital format forms the most expensive (about 80% of the total GIS project cost) and time consuming part of any GIS project, data sources for data acquisition should be carefully selected for specific purposes. The following data sources are widely used: Analog maps, aerial photographs, satellite images, Ground survey with GPS and finally from reports and publications. Satellite imagery of an area covered by A.B.U was collected through Google Earth and used for the research work.

2. How to generate 3D data from the 2D data obtained from the satellite imagery?
3. Having realized the benefit of the GML data structure and modeling, how then do we render the 3D-Geospatial data in XML/GML format?

   We propose a framework that is capable of serving 3D geospatial data in XML/GML format through our Python engine.
4. How do we test this generated GML data structure/model (i.e. visualizing its contents)?

1.4 Research Objectives

The main objectives of this research include:

1. Collection of geographical data in analogue format and its subsequent transformation and storage in digital format.
2. Identifying the mechanism provided by the GML data format in solving the major problems associated with Geo-Spatial data.
3. Defining a framework to render this digital geospatial data in an XML/GML format.
4. Implementation of the framework structure.

1.5 Methodology

The following are the proposed steps needed for the realization of this research work.

1. Review literatures to understand why it is necessary to render geospatial data in XML/GML format.
2. Conduct extensive literature review on different rendering frameworks of geospatial data in XML related formats.

3. Primary data acquisition:

   The steps involved in this process include:

   - Locate an area of study (A.B.U, Zaria) through satellite imagery capture.
   - Spatial referencing to extract registered points for the above features captured in the satellite imagery which results to coordinates (digitizing).
   - Extraction and naming of features (e.g. roads, buildings etc).

4. Storage of the data acquired from step one above into a spatially enabled database i.e. PostgreSQL configured with PostGIS.
5. Develop and implement a Python Engine to render the stored data in XML/GML format.

1.6 Contributions to Knowledge

i. Developed a digital dataset for a segment of ABU, Zaria.

   - To our knowledge, no digital dataset was available as a data representation of geographic features in ABU.

ii. Developed a framework for encoding digital datasets into GML format.

   - To leverage the benefits of GML data structure/ modeling of geographic information.

ii. Developed a Python Engine that defines an algorithm for transforming stored geographical digital information into 3D GML data.

CHAPTER TWO

LITERATURE REVIEW

This chapter discusses Geo-Spatial data, Spatial Data Types (SDTs), Geographic coordinate systems, Geo-spatial data file format, Geo-spatial data collection strategies, major problems associated with Geo-spatial data usage, the web as a major platform to enhance Geo-spatial data interoperability, Geography Markup Language and the mechanisms it provides for data interoperability. The chapter also takes a look at tools for generating GML and there drawbacks then possible visualization strategies of GML data and finally review of related work.

2.0 Geo-Spatial Data

Spatial data are data with an associated spatial location with respect to a given coordinate system. Geographic data are spatial data with respect to the Earth's surface (Nigel, 2001), such as a map of cities in the United States or a satellite imagery covering the State of California. In this thesis, we focus on the modeling and management of geographic data. To be consistent with the literature, we also use the notion spatial data to refer to geographic data, and spatial object to refer to geographic object. There are two ways to describe geographic data: field-based models and object-based models (Shashi, 1997). Field-based models treat geographic phenomena as a continuous surface, while object-based models describe them as discrete objects such as cities and highways. For example, both models can describe land use data. On a field-based map, every point on a land use map is classified into a category such as farm land or residential area. On an object-based map, it is possible that some points on the map do not belong to any land use object. In this thesis, we primarily focus on an object-based spatial data model.

A geographic object on the Earth's surface is represented by two components: spatial attributes and descriptive attributes. Spatial attributes describe the geometric shape and location of an object, while descriptive attributes describe the non-spatial properties.

2.1 Geographic Coordinate System

A coordinate system is a system for specifying the location of a point in an n-dimensional space via an n-tuple of numbers (Phillippe, R. et al (2002)). We use 2-dimensional geographic coordinate systems to specify the location of geographic objects on the surface of the Earth, which is a 3-dimensional sphere. To do this, we introduce the concept of a map projection.

A map projection is a representation of the surface of the Earth on a 2-dimensional plane. It describes the space of the 2-dimensional coordinate system to position objects on the Earth. Figure 1.0 illustrates three widely adopted projection plane surfaces: azimuthal (or flattenable), conic, and cylindrical. No projection can be true with respect to all desirable properties such as distance, direction, scale, and area (Map Projection, 2005). Therefore, an appropriate projection should be chosen for a specific application of geographic data. For instance, the Universal Transverse Mercator (UTM) projection is a cylindrical projection that projects the sphere onto a cylinder tangent to a central meridian (Map Projection, 2005). It is often used to portray areas with larger north-south than east-west extent. Distortions of scale, distance, direction and area increase away from the central meridian. The two dimensions of the UTM Coordinate system are called Eastings/Northings.

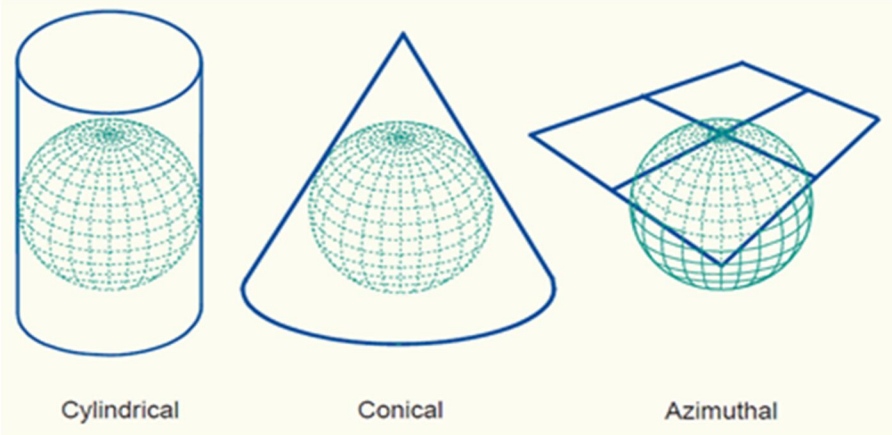


Figure 1.0: Major classes of projections

Eastings are relative positions from the central meridian, while northings are relative positions from the equator.

2.3 Geo-Spatial Data Types (SDT)

Spatial data types are special data types to model geometry and to represent the spatial attributes of spatial data in database systems (Markus, S. (1999)). Spatial data types provide an abstraction for modeling the geometric structure of geographic objects on the Earth. The most commonly use spatial data types include: point, polyline and polygon.

A point is to represent an object whose shape is not considered essential at some scale. A point is a spatial object represented only by its location on the Earth. A point pt at location (x, y) is denoted as pt[x, y].

pt(x,y)

Figure 2.0 A Point

A polyline is used to represent a linear object in space. Highways, rivers, and shorelines are ideal objects to be modeled as polylines. A polyline is a finite sequence of line segments, such that each segment endpoint is shared by exactly two segments, except for the two endpoints that belong to only one segment.

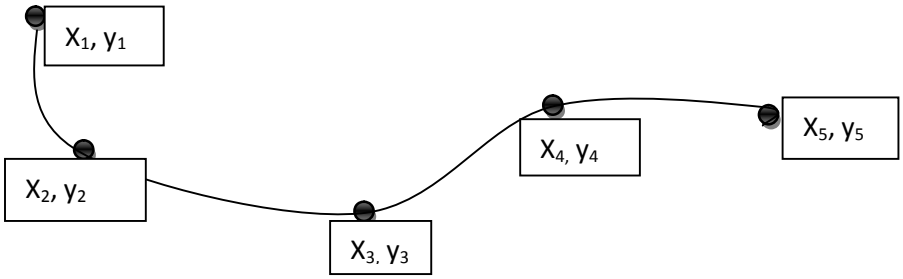


Figure 3.0 A Polyline (could be use to represent a road, path or river)

A polyline pl is constructed from a list of points $< pt_1, pt_2. \ . \ . \ pt_n >$ in a way such that each pair (pt_i, pt_{i+1}) in pl $(i < n)$ represents one edge of the polyline. To add the location property, all points of pl are represented as:

$$pl\ [< x_{pt1}, y_{pt1} >, < x_{pt2}, y_{pt2} >, \ . \ . \ . \ , < x_{ptn}, y_{ptn} >]$$

A polygon is used to represent a region on the Earth. A polygon is a region bounded by a closed polyline, called its boundary that does not intersect itself.

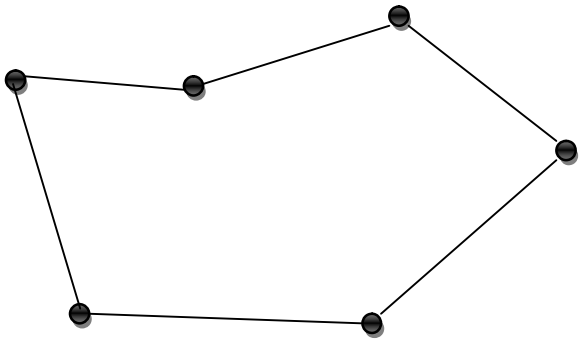


Figure 4.0: A Polygon (could be used to represent the floor cover of a building, river e.t.c.)

Assume a polygon pg is constructed from a polyline as described above. Then pg is represented as:

$$pg\ [< x_{pt1}, y_{pt1} >, < x_{pt2}, y_{pt2} > \ldots < x_{ptn}, y_{ptn} >, < x_{pt1}, y_{pt1} >].$$

By Point, Polyline, and Polygon we mean the spatial data types (SDT) as defined above (i.e., all possible Point, Polyline and Polygon values). SDT is the union of these types.

2.4 Geo-Spatial Data File Format

Raster and Vector Data file format are two methods of representing geographic data in digital computers.

Raster Data: records spatial information in a regular grid or matrix organized as a set of rows and columns. Each cell within this grid contains a number representing a particular geographic feature such as soil type, elevation, land use and slope. Raster data are commonly but not exclusively used to store information about geographic features that vary continuously over a surface, such as elevation, ground water depths. Image data is a form of raster data in which each

cell or pixel stores a value recorded by an optical or electronic device. Examples of raster spatial data format include: JPEG, EGN, MrSID e.t.c.

Vector Data: records spatial information as x, y coordinates in rectangular (planar) coordinate system. Point features are recorded as single x, y locations. Line features including the outlines of polygons are recorded as an ordered series of x, y coordinates. The vector representation of an object attempts to show the object as much as possible in its exact shape. Example of vector spatial data file format include: Smart Data compression (SDC), layer file (.lry), shapefile (shp), triangular Irregular Network (TIN), Geography MarkUp Language (GML).

2.5 Geo-Spatial Data Collection

Geo-spatial data collection in digital format is very expensive and time consuming, so data collection sources should be carefully selected for specific purposes.

The following data sources are widely used:

- Analog Maps: Topographic maps with contours and other terrain features and thematic maps with respect to defined object classes are digitized by digitizers manually or by scanner semi-automatically. Problems of analog maps are lack of availability, out of date, inconsistency in map production time, inaccuracy e.t.c.
- Aerial Photogragraphs: Analytical or digital photogrammetry is rather expensive but the best method for updating.
- Satellite images: satellite images or data are available for land use classification, digital elevation model (DEM), updating highway network e.t.c.
- Ground Survey with GPS: The Global Positioning system gives a very accurate but too expensive method of geo-spatial data collection when covering a wide area.
- Reports and publications: social economic data are manually listed in the reports of statistics and census with respect to administrative units.

2.6 A Major Problem Associated With Geo-Spatial Data Usage

Spatial datasets are produced and distributed in a variety of formats: vector, raster e.t.c. often datasets are designed for certain computer system (e.g. UNIX, DOS/Windows, and Macintosh) or software programs (GRASS, Idrisi and ArcInfo). In the likely event that an important dataset is available, but in the wrong format the prospective user of such data must be aware of the issues, method and tools for converting spatial data to a format which is compatible with the current project at hand.

To solve this major concern in the GIS world the issue of geo-spatial data inter-operability then arises. Also in an attempt to achieve that, standards were made for data exchange.

2.7 The Web as a Major Platform to Enhance Geo-Spatial Data Interoperability

The development of the World Wide Web creates a unique environment for sharing geospatial data. MarkUp Languages are languages of the web. They include languages like HTML, XHTML, XML e.t.c.

XML stands for Extensible MarkUp Language. XML is a set of rules for defining semantic tags that break a document into parts and identifying the different parts of the document. It is a meta-markup language that defines syntax in which other field-specific markup languages such as: MusicML, MathML, CML, GML e.t.c. can be written. Also, XML markup describes a document structure and meaning. It does not describe the formatting of the elements on the page. Formatting can be added to a document with a stylesheet. The document itself only contains tags that say what is in the document not what the document looks like. Elliotte, R.H.(2001)

2.7.1 Geography Markup Language (GML)

The creation of a new standard data exchange format--Geography Markup Language (GML)--represents another important step taken by the geospatial community towards data interoperability. The GML is *an XML grammar for the modeling, transport, and storage of geographic information including both the spatial and non-spatial properties of geographic features* (OGC, 2003) It is developed as Data Exchange Standards Interface by Open GIS Consortium (OGC) to achieve data interoperability and reduce costly geographic data conversions between different systems.

With the development of the World Wide Web Consortium's XML (Extensible Markup Language), the creation of Geography Markup Language Implementation Specification by OGC represents a significant step in the development of interoperable architectures for the use of spatial information between different applications. GML holds promise to support mapping from a wide variety of sources and enable sharing of geospatial data for on-line information exchanges.

Unlike current proprietary commercial Internet GIS programs, the OpenGIS GML specification is a public open standard for coding and sharing spatial data. GML is a good alternative to expensive, proprietary web-based mapping solution:

(1) GML is an open source standard. Users can use it for free. But for other commercial Internet GIS programs, users have to buy. For example ESRI ArcIMS Internet software is so expensive that many users cannot afford it. But they need provide online spatial data services. GML is a good alternative for these users. With GML they can provide the online spatial data services without buying these proprietary software.

(2) GML data are stored in text format, which is a universal format. Thus it is easy to integrate GML data into other data across a variety of platforms and devices.

(3) As a standard data exchange format GML reduces the costly conversion processes among different format databases.

(4) Although GML specifications take the standardization approach to data interoperability as do GDF and SDTS, it goes further and supports interoperable solutions that geo-enable the Web. While GDF and SDTS are useful for off-line data exchange, GML is capable of facilitating real-time data sharing and exchange on the Web because it uses XML grammar which is widely supported on the Web. GML can enable an accessible Geo-Web (Lake, 2002; Peng and Tsou, 2003; Shekhar *et al*, 2001).

2.7.2 Mechanisms of GML for Data Interoperability

The mechanisms of GML for data interoperability are given as following:

1. GML provides a common schema framework for encoding geo-spatial features. GML uses the W3C XML Schema Definition Language to define and constrain the contents of its XML documents.

   GML schemas reconcile the need for standardization with the need for diversity by providing a standard means of extending the GML format. The direct consequence of applying schemas with GML is that it becomes possible for organizations to define formats to suit their needs and exchange geographic information without the need to involve software developers to create translators for that format. This has impacts on both the cost and risk of exchanging data (Curtis, 2003).

2. While GML builds on XML Schema, it provides a more constrained model. GML is based on a common abstract model of geography (OGC Abstract Specification), which describes the world in terms of features. A geographic feature is an abstraction of a real

world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth (OGC Abstract Specification, 2001) A feature has both simple properties and geometry properties. Simple properties refer to the usual name, type and value description. Geometry properties are composed of points, curve (linestring) and surface (polygon). By looking at feature schemas and properties, one can readily compare features and integrate data.

3. GML is based on an XML standard. XML is a universal format for structured documents and data on the Web. XML is easy to transform. Using XSLT or almost any other programming language (VB, VBScript, Java, C++ and JavaScript), users can transform XML from one form to another. By adhering to an open, non-proprietary standard, GML documents can be manipulated, transformed and presented in the same flexible way as XML contents.

4. GML provides XLink and XPointer mechanisms as does XML. The linking mechanism of HTML (one web page linking to another), is one of the key foundations of the Web. GML goes further by providing a mechanism for linking multiple distributed resources into a complex association. As HTML is important to the Internet as a linked collection of web pages, GML can enable the development of a Geo-Web as a linked collection of geo-spatial features. Through XLink and Xpointer, different features and feature collections, which may be located remotely, can be associated together at the feature level (Peng, 2003). XLink and XPointer hold great promise for building complex and distributed geographic data sets (Lake, 1999). They make it possible to access and seamlessly integrate data from different departments, cities, states and countries.

5. GML provides a means to transport geospatial data over the Web. With the help of Web Feature Server (WFS), spatial databases with different formats can transparently communicate with each other by being converted to GML-format data on the fly.

Figure 5.0 illustrates that GML geo-enables the Web by transporting and re -using geospatial data with different formats using GML. As XML is an important Internet data transport technology, GML makes it possible for real-time data access and transport in the Internet environment at feature level. When the GML-marked geospatial data are transported, all the markup elements that describe every spatial and non-spatial features, geometry and spatial reference systems of the data are also transported to the recipient (Peng, 2003).

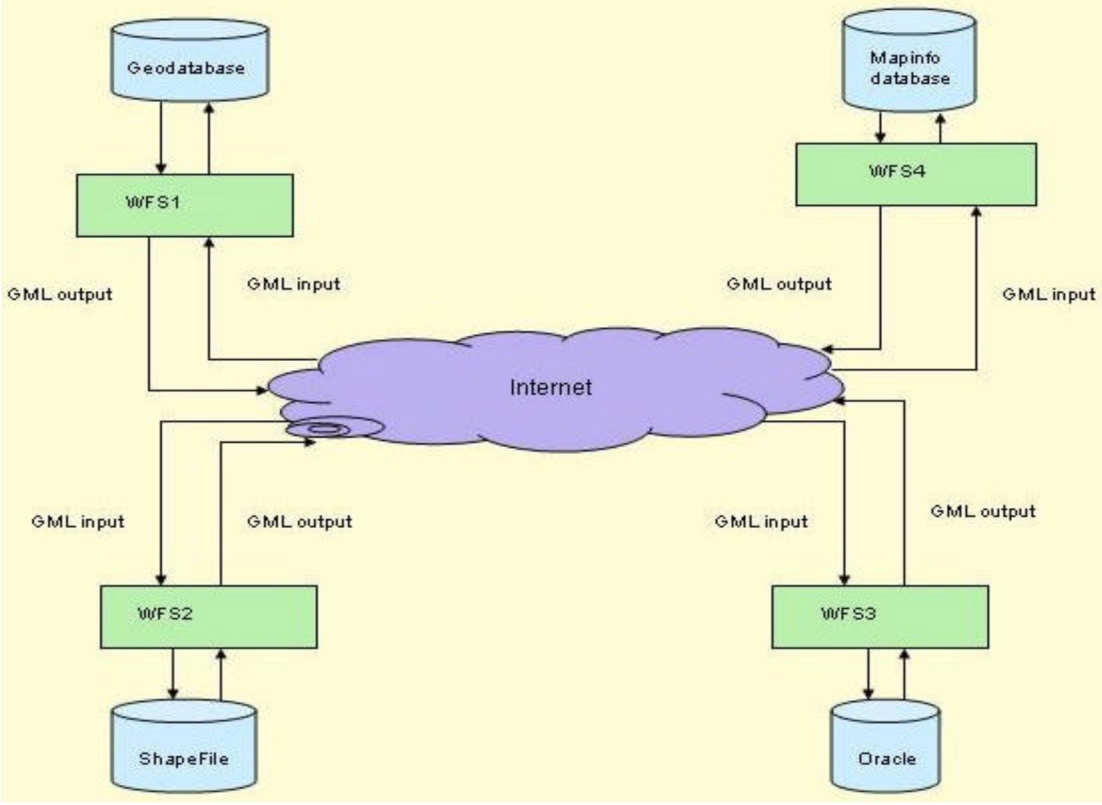


Figure 5.0 Geo-enabling the Web by transporting and sharing different format geospatial data using GML (Peng, 2003)

6. GML data are stored in plain text. Text is vendor-neutral, so information stored in GML is not locked into a proprietary binary format. Since GML is text-based, it can readily integrate geospatial data with a wide variety of non-spatial data types including text, business transactions, graphics, audio, voice and more. This capability would greatly enhance the value and accessibility of geospatial information. For example, users can easily insert a map in a financial report, or vice versa (Peng and Tsou, 2003). In addition, as a text format, GML can be easily transmitted across a variety of platforms over the Internet. Thus GML enables disparate systems to share information easily. GML mechanisms allow users to build a large, global map stored and processed in a scalable and redundant distributed architecture. GML makes it possible that all spatial data in the whole world can be integrated into one map (Misund and Johnsen, 2003). The inherent transformability and accessibility of GML opens a new domain for the geo-community (Lake, 1999).

2.7.3 Geographic Data in GML

The introduction of XML has enabled easy creation of specialized markup languages and as a result some geographically oriented languages were created to cope up with the above requirements, but none was successful as an acceptable standard for the geo-information field. Finally, the OpenGIS Consortium (OGC) proposed a standard called Geographic MarkUp Language (GML), for representing geographic data and its meaning in an XML–based form. It specifies structures for encoding data and their associations and provides a means for transporting and storing data in the internet environment (R. Lake, 2001).

GML 1.0 was approved by OGC in May 2000 and it is based on XML 1.0 specifications. It consists of three forms of GML profiles, which offer different levels of flexibility in data structuring. GML 1.0 makes use of Document Type Definition (DTD) and Resource Description

Framework (RDF) for encoding geographic features in three profiles. (A. Cuthbert, R.Lake *et al*., 2000).

The Profile 1 is for those who wish to use pure DTD based solution and are not prepared to develop application specific DTD's or wish data to be returned against a fixed set DTD's. This profile requires the use of GML features and GML geometry DTD's.

The Profile 2 is for those who wish to use pure DTD based solution but are prepared to develop their own application specific DTD's or are prepared to accept data encoded with a referenced DTD. This profile requires the user to create an application specific feature DTD that uses the GML geometry DTD.

The profile 3 for those who are prepared to make use of RDF (Resource Description Framework) and RDF schema. The users of this profile will typically require stronger control over the geo-spatial typing framework (e.g. they must be able to relate the type name to an actual schema definition). This profile requires the user to create an application specific RDF schema definition that uses the GML RDF schema definition. RDF Schema, used in profile 3 is an alternative basis for DTD, which is used in profile 1 and 2.

GML 2.0 was developed and passed in March 2001 by OGC as the successor to the GML 1.0. It provides a single encoding method (XML schema) and a single approach for creation of GML schemas. XML Schemas adoption in GML 2.0 avoids the awkward combination of DTD and RDF in GML 1.0 and it incorporates support for type inheritance, distributed schema integration and namespaces. The use of XML schema in GML provides validation mechanism defined in XML 1.0 specification (Cox et *al*., 2001)

GML 2.0 was concerned with what the OGC calls simple features: features whose geometric properties are restricted to 'simple' geometries for which coordinates are defined in two dimensions. These simple features represent such things as rivers, roads and colleges. Each

features types would have named and typed properties. The college building feature type might have a property called name whose value must be of the type 'string'. The features required to represent a city might have geometry-valued properties as well as simple properties. The college feature type might have both a point property called location and a polygon property called campus. So, GML version 3 addresses the following needs that were not addressed or adequately met by the previous version:

- Represent geospatial phenomena in addition to simple 2D linear features, including features with complex, non-linear, 3D geometry, features with 2D topology, features with temporal properties, dynamic features, coverages and observations.
- Provide more explicit support for properties of features and other objects whose value is complex.
- Represent spatial and temporal reference systems, units of measure and standards information.
- Use reference system, units and standard information in the representation of geospatial phenomena, observations and values.

The expansion of GML to meet these needs is reflected in the base schemas for GML version3 that are over eight times as large as the base schemas for GML version 2. (S. Cox, *et al*., 2005).

2.7.4 Generating GML Data

Some tools that can be used to generate GML from other data format include:

-OpenJump: is an open source Geographic Information System (GIS) written in Java Programming Language. It is developed and maintained by a group of volunteers from around the globe. The current version can read and write shapefiles and simple GML files. It has limited

support for display of images. OpenJump has limits in reading very large data files and has limited support for cartographic projections.

-GeoServer: is an open source software server written in Java that allows users to share and edit geo-spatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards. Geoserver is limited to 2D geospatial data rendering. Geoserver does not come built-in with support for GML; it must be installed through an extension.

-Quantum GIS (QGIS): It is a powerful and user friendly open source geographic information system (GIS) written in C++ that runs on Linux, Unix, Mac OSX, Windows and android. QGIS supports vector, raster and database formats.

2.7.5 Visualizing GML Data

In addition, GML can deliver vector data over the Internet by styling the data into Scalable Vector Graphics (SVG) /X3D format. Most current Internet GIS programs deliver spatial data through transmission of raster images such as GIF and JPEG formats over the World Wide Web. There are several advantages for delivering SVG/X3D vector GIS data over the Web compared with raster GIS data:

(1) Compatibility: SVG/X3D also uses text-based XML format, which is compatible with other formats. It can seamlessly integrated with current Web technologies, such as HTML, JavaScript, JSP, ASP, JPEG, GIF, etc.

(2) Graphic quality: SVG/X3D format graphics are scalable and resolution-independent. This kind of data can be scaled without loss of quality across different platforms and devices. But coarse raster images are low quality because of a low resolution. Especially when users zoom in too many times, images will become blurred and pixilated. However, a raster image with high resolution usually has a larger file size since it needs

to store information as finer pixels. The speed of delivering such large files over the Web becomes slow, so it is not practical to use high-resolution images for Internet GIS. The need for delivering high quality vector graphic maps over the Internet is becoming pressing as data availability and global sharing increases (Bertolotto and Egenhofer, 2001).

(3) SVG/X3D vector data can be accessed in a more interactive and dynamic way; some dynamic functions can be integrated into SVG/X3D documents so that the SVG graphics are animated on the Web.

For example, a SVG graphic can interact with users by mouse over if a mouseover() function is added in the SVG/X3D document. By combining SVG/X3D with other web technologies like HTML, JavaScript, JSP or ASP, GML-based database can provide users an extremely rich interactive graphic interface.

Although current Internet GIS programs can let users share spatial data online, they have the aforementioned proprietary data model and database structure problems. Thirdly, GML-based databases can let users exchange data at feature level, while current commercial Internet GIS programs cannot. For example, from a big GML-based database, users can just query and download one feature such as a specific road, while from other alternatives users have to download the whole data set. Sharing and exchanging data at feature level in real time are especially important for emergency services; they can greatly reduce the time spending on data acquiring processes. Fourthly, by styling GML data into SVG/X3D, GML-based databases can provide users a more sophisticated interactive graphic interface and deliver higher quality graphic maps over the Web than most other online alternatives. Fifthly, GML is more flexible than other alternatives. It only defines a basic geographical feature schema and geometry

schema, which are convenient for users to use. Based on these schemas users can define their own specific schemas for their spatial data documents. It has been widely recognized that GML will play an important role as a future Web data exchange standard (Clemens, 2002; Lake, 1999; Meneghello, 2001; Murray and Chow, 2002; Smith *et al*, 2002).

Other software tools that can be used to visualize GML data include: MapBuilder GML Viewer, Carbon Tools GAIA, and gvSIG 1.1, Openlayers: a project of the open source geospatial foundation, uDig (user-friendly Desktop Internet GIS (uDig) and GlobalMapper.

2.8 Review of 3D GIS Implementation with the Web's Markup Language

1. 3D GIS Prototype using VRML by Morcrette, 1998

In the USA (Morcrette, 1999) was one of the first researchers to transform spatial information in a database to a 3D scene in VRML using Java. VRML, the Virtual Reality Modeling Language is text based file format or syntax to describe 3D worlds, and the predecessor of X3D based on XML.

A distinction is made between two prototype architectures. In the first prototype architecture, two tier (figure 6.0, left image); there is a direct connection between the VRML file and the database. A Standard Query Language (SQL) scripting node within the VRML realizes the connection; this is a special node that is used to execute normal SQL commands. In the second prototype architecture, three tier (figure 6.0, right image), there is a web server between the 3D VRML visualization and the database. In this architecture, the VRML browser makes contact with the web server in the middle layer, which connects to the database, reads out the data, and sends it back to the VRML browser.

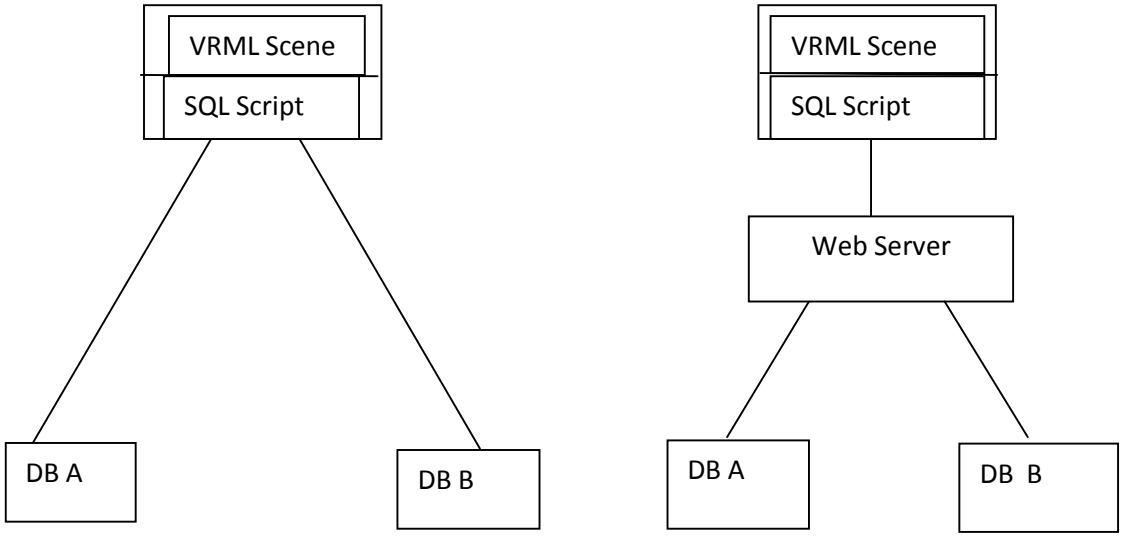


Figure 6.0: VRML / JAVA web GIS architecture by (Morcrette, 1999)

The second architecture is more appropriate for the Internet where firewalls are in existent, while the first can be used on an intranet. The client proposed is a Java applet that runs on the client. The database is accessed using the Java DataBase Connectivity protocol (JDBC) (Sun Microsystems, 2007). JDBC is a standardized protocol to connect to different kind of databases with the same technology using Java; similar to the Open DataBase Connectivity protocol (ODBC). The final prototype architecture is composed of the following three components. The client, a VRML browser (Cosmoplayer), is responsible for displaying (VRML data view) and forwarding input information from the user. A server (unknown brand) is responsible for the data communication, and a Toolset is responsible for the 3D code generation. VRML nodes can be updated individually, for example, the colour can be changed, or a texture can be added.

2. 3D GIS prototype based on VRML and Java by Kim et al, 1998

About the same time in China (Kim et al, 1998), propose a 3D web GIS prototype architecture, based on Java and VRML with more possibilities. 3D GIS functionalities as visualizing, getting

non-spatial attributes, 3D feature indexing, 3D analysis such has selection, buffer and near operation, and metric analysis such as distance and statistics, are implemented in Java, while the 3D visualization is done in VRML. Again, the system is implemented in a Java applet.

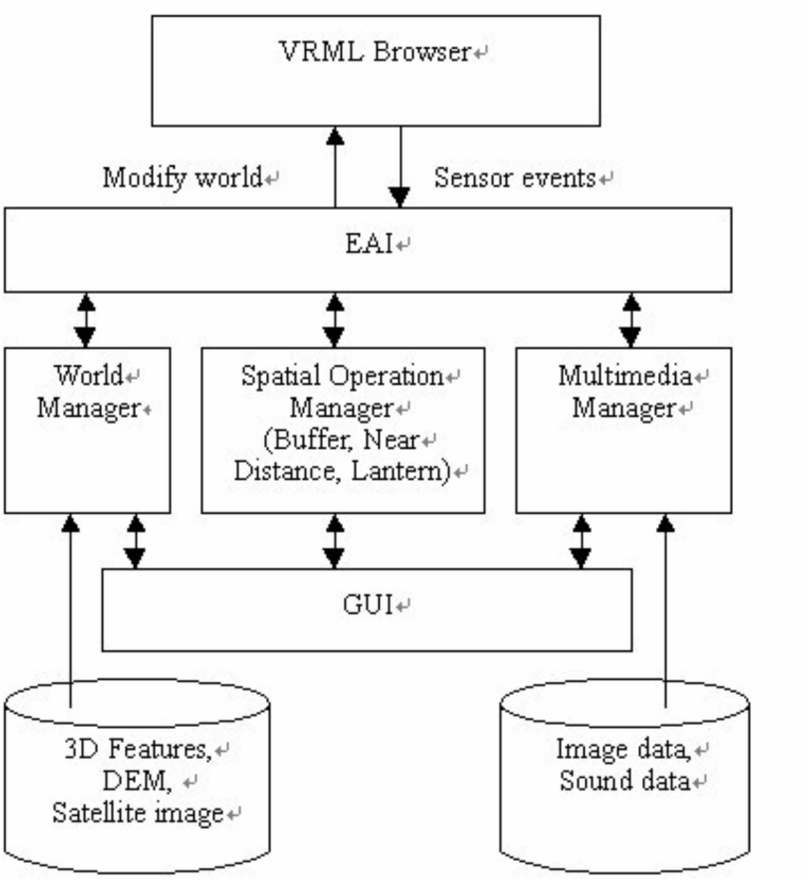


Figure 7.0: VRML / JAVA 3D web GIS prototype architecture by (Kim et al, 1998)

The EAI allows the Java applet to interact with the 3D visualization in VRML. The EAI is also used in Morcrette's project. The advantage of the EAI is that VRML can be dynamically built, and updated based on data received by the Java applet. The other way around, data of the Java applet can be updated from within the 3D visualization in VRML.

The architecture (figure 4.2) of this prototype is layered. In the middle there is the EAI running in the Java applet. Via the World Manager, it can request 3D features, DEMs, and satellite

images from the database (brand not mentioned). The 3D features are made up of four components, which are a geographic primitive as a node, chain, polygon or DEM, visualization information as shape or colour, non-spatial attributes as street name or owner, and multimedia related information for example a website with a monument database.

It is an advantage that the 3D visualization can be dynamically updated with data from the database, and data in the database can be updated from the 3D visualization. Kim et al created the Spatial Operations to do 3D analysis in Java. The results are visualized in the VRML world by letting the EAI update the individual nodes.

Analyses that can be performed in 3D by the spatial operations manager are feature identification, 3D buffer, 3D near for example find the buildings within 100 meters of a school and Lantern (see article by Kim et al. for more detailed information on Lantern operation).

3. 3D GIS prototype based on X3D, Java and Oracle XSQL servlet by de Vries et al, 2003

At OTB – TUDelft, the Netherlands, (de Vries et al, 2003), build two prototypes for 3D visualization. The second prototype uses X3D .The second prototype by (de Vries et al, 2003) is based on an Oracle XSQL servlet, X3D and Java. A servlet is a Java program that runs on a server. The servlet processes requests on the server, which are made on the client. The server requests geometrical information by means of a XSQL query from the Oracle database. By executing an XSQL query, the data in the database comes available as XML. XSQL is the combination of XML (Extensible Markup Language) and SQL (Structured Query Language) to provide a language and database independent means for storing SQL queries, clauses and query results (Dooling, 2002). The XML is transformed to X3D with a stylesheet transformation (XSLT). Attribute information is requested from the database with another XSQL query. XSQL

uses JDBC to make a database connection. The XML stream from the database can be transformed to X3D for a 3D view, to Standard Vector Graphics (SVG) for a 2D view (although this is not done in their prototype), and to HTML to show the attribute information in a table. Queries can be filled in a form and are picked up by JavaScript and then processed by Java and XSQL. Finally the prototype offers the possibility to enter a bounding box from which data is selected. The prototype has a connection between the 3D visualization on the client and the database on the server in two ways.

• Geometrical information is read from the database and transformed to X3D.

• When an object is clicked, its attribute information is displayed as a HTML table.

The part of the data that is visualized is dependent on the query and bounding box filled in by the user on the form. In this way, loading times for large datasets can be decreased by only requesting a certain area.

2.9 Relationship between our proposed framework and the prototype reviews.

The approach taken to encode geographic information in MarkUp Languages in the above review is similar to our proposed framework especially that of Morcette (1999). Although, all the reviewed approaches tend to encode geographic information in a graphic MarkUp language (i.e. VRML and X3D) whereas our approach encodes geographic information in Geography MarkUp Language (GML) and other graphics MarkUp Languages as well. Most of the implementation in our review is based on Java while our framework explores the power of Python.

CHAPTER THREE

SYSTEM DEVELOPMENT

In this chapter we developed a framework for the 3D rendering of Geo-spatial data in GML; In the process, the following issues were discussed: the system requirement for the four basic stages of our framework which include: data collection, data storage facility, python as a programming language used in the implementation of our GML generating engine and the visualization of our generated data, the encoding of geographic data in GML.

3.0 System Requirements

-Google Earth: is a free software that allows you to zoom in to see highly detailed aerial photos or satellite images of any place on the planet earth. Once you install Google Earth, you will be able to lauch it. On the left-hand side of the screen, you will be able to search layers and places. Use search to look for specific address, a city name or a country and Google earth will fly you there. Google earth also allows you to download this images/data. “Streaming” provides an indication of how much data has been downloaded-once it reaches 100%, which is the best resolution you will see in Google Earth.

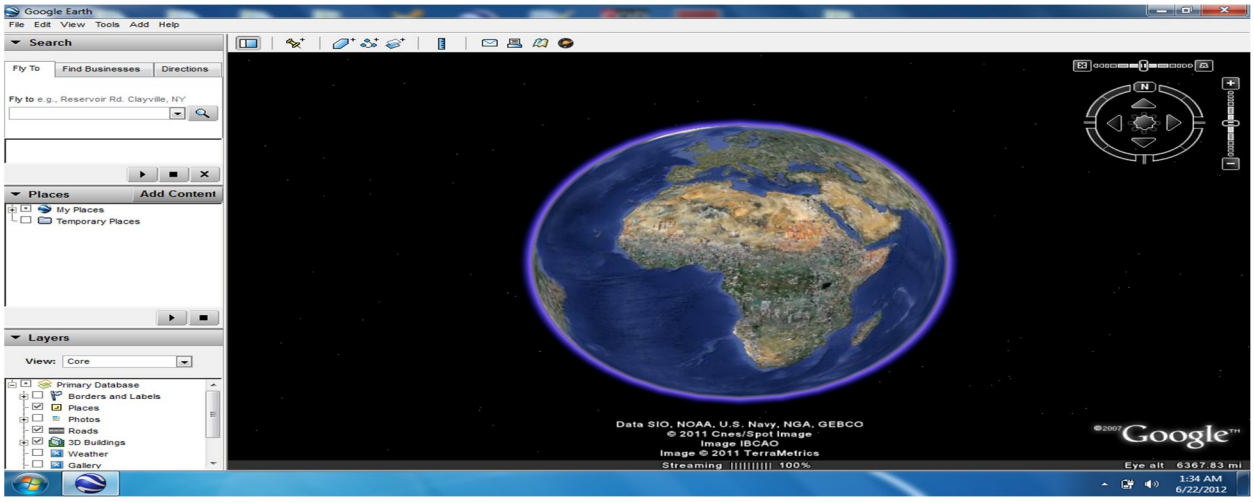


Figure 8.0: Google Earth

-ArcGIS Desktop: includes a suite of integrated applications that allow you to perform GIS tasks from simple to advanced, including mapping, geographic analysis, data editing and compilation, data management, visualization and geoprocessing. This tools was used to digitize the satellite imagery collected from google earth. Digitizing is the process of converting features on the paper map into their equivalent digital format. The x,y coordinates of these features are automatically recorded and stored as spatial data which can be viewed in the attribute table of the software. The attribute table is edited to add some non-spatial information like describing each features and labeling the feature with feature id.

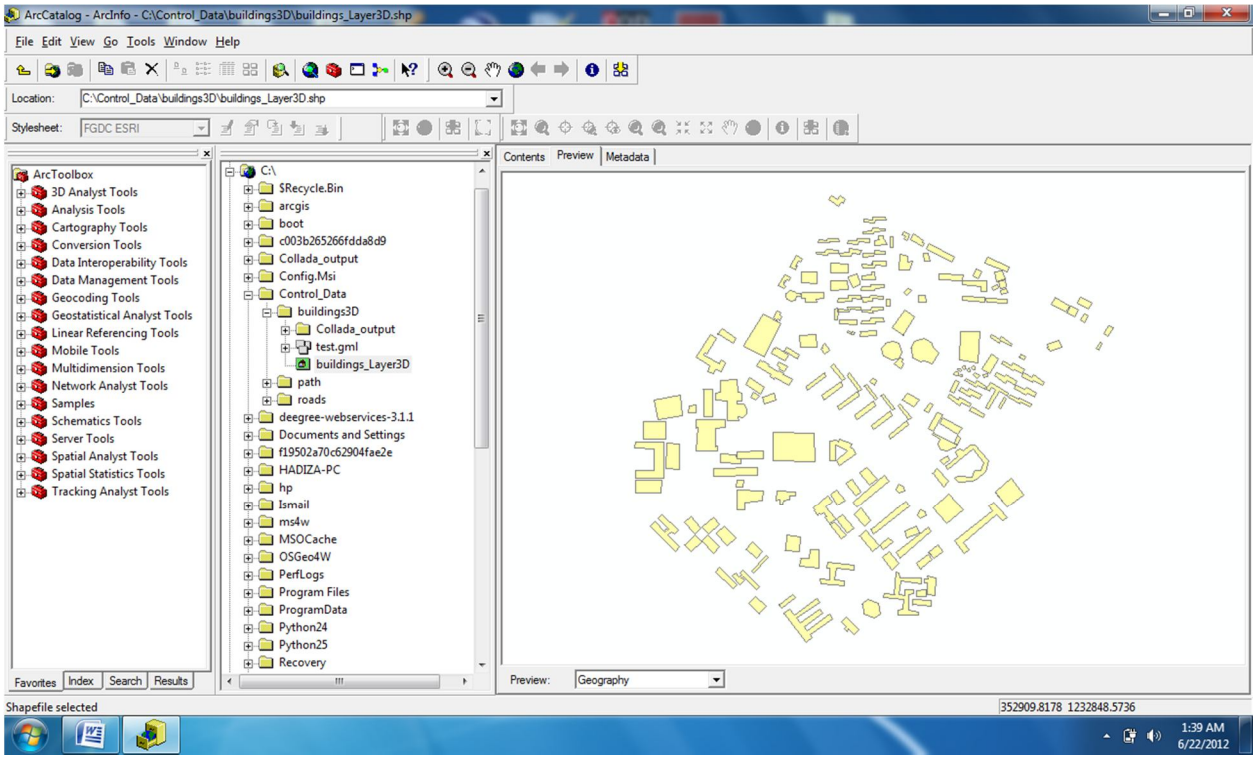


Figure 9.0 ArcGIS

-PostgreSQL: often simply termed postgres, is an object-relational database management system (ORDBMS) available for many platforms. Several versions of this database exist but the

8.4 version was used as it was more compatible to the system in use. PostgreSQL is able to store spatial data with the aid of a PostGIS extension.

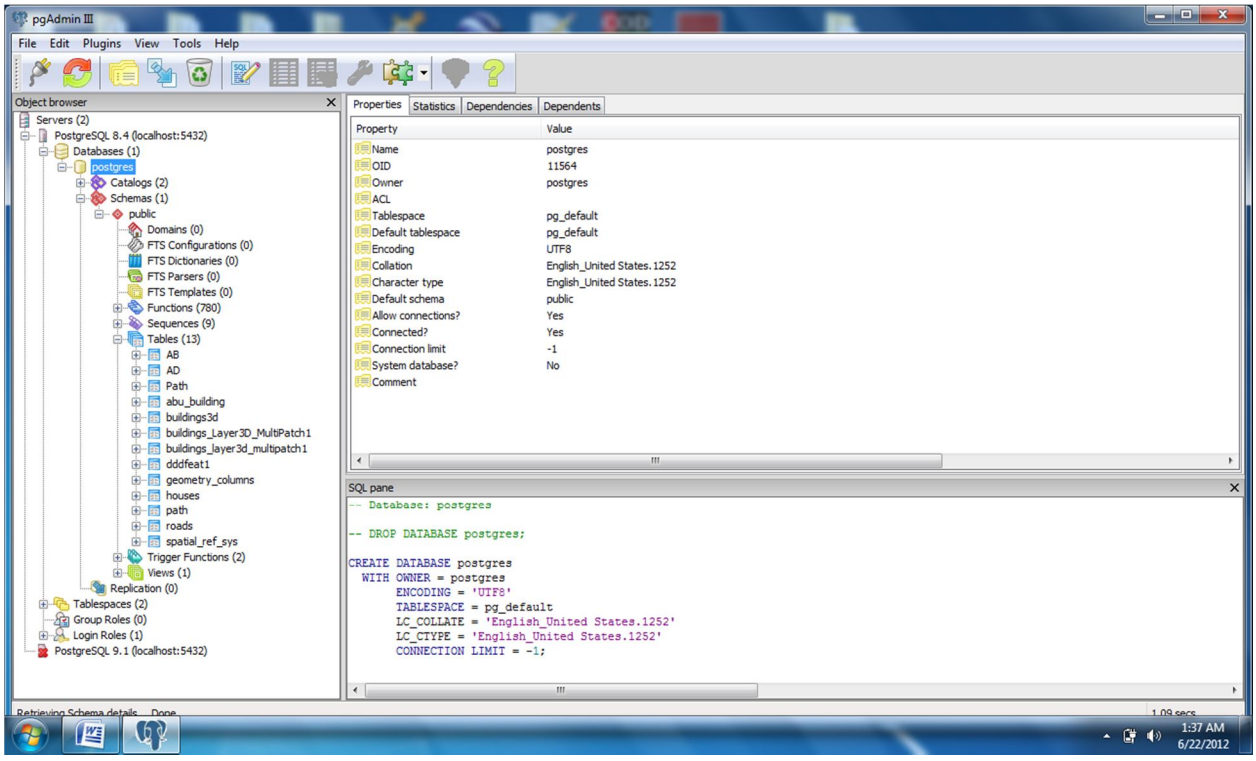


Figure 10.0 PostgreSQL Database

-PostGIS: PostGIS is an Open source, freely available and fairly OGC complaint spatial database extender for the PostgreSQL database Management System. PostGIS is very similar in functionality to SQL Server 2008 spatial support, Oracle Spatial and DB2 spatial extender. PostGIS 1.5.0 which is the latest stable version as at the time of this research was used for our implementation processes.

-Python: Python is an object oriented, high-level, prototyping language. Python is a scripting language which means it is interpreted every time it is run. Advantages of Python over

other (scripting) languages are that it is extensible and embeddable in applications. Python is free, open source, easy to learn and well readable (Van Rossum, 1999).

Python is a Programming language, just as Java, Visual Basic and C++ are. However, there are some major differences with these languages. Python is dynamically typed. This means the type of a variable never has to be declared. In Python, if a name is assigned to an object of one type it may be later assigned to an object of another type. Python container objects can hold objects of any type including numbers and lists. When an object is retrieved from a container, it always remembers its type unlike in Java (Ferg, 2007). Python is more concise and less verbose than other programming languages. It means it uses less words and characters.

```
// Java code to print hello world
Public class HelloWorld
{
        Public static void main (string [] args)
        {
                System.out.println ("hello world");
        }
}
```

Table 2: Java code to print hello world

```
#python code to print hello world
Print "hello world"
```

Table 3: Python code to print hello world

Python code can be 2-10 times shorter than similar C++ or Java code. Python has a number of smart operations to inspect string. For example:

“Hello”[6] would result in “o”. “Hello”[3:6] would result in “llo”, “Hello”[3:] would result in “Hel” and “Hello[:3] would result in “llo”.

Drawbacks of python are it can run slower than similar C++ or Java code. The use of indentation for loops and if statements can be confusing, especially when they have to be closed. Python can connect to a database by using a python module that can connect to a database.

For the PostgreSQL database, python’s popular module that could be used includes:

- Psycopg/ Psycopg2
- PyGreSQL
- pyPySQL

Access to the database is available through connection objects. For our PostgreSQL connection we used the Psycopg2 database adapter module. It main advantages are that it supports the full python DB API and it is thread safe (i.e. threads can share the connections).

Basic Module Usage:

```
#creating a connection
        conn = psycopg.connect( host=          dbname=          user=          )
```

Table 4: Database connection with Python’s Psycopg2 module

then, you’ll use the connection to create a cursor

```
# open a cursor to perform database operations.
        cur = conn.cursor ( )
```

Table 5: cursor creation to enable database manipulation

the cursor objects represent a database cursor, which is used to manage the context of fetch operations. Cursors created from the same connection are not isolated.

Cursors methods include:

cur. execute (operation, parameters), cur. fetch ( ), cur.fetchall ( ), cur. close ( ) e.t.c.

Python Tools for XML:

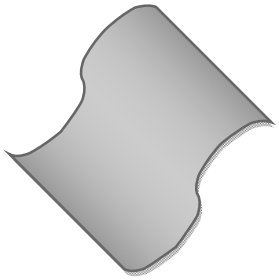
Three major packages provide Python tools for working with XML.

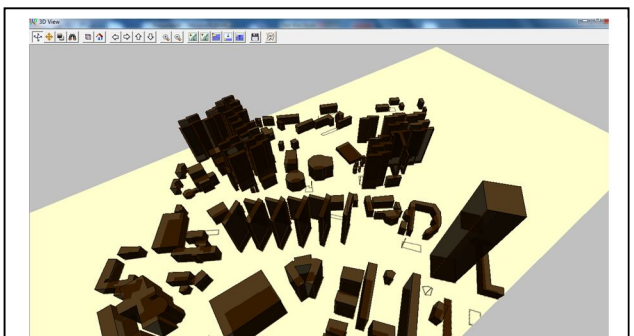
- The Python Standard library.
- PyXML, produced by Python XML Special Interest Group
- 4Suite, provided by fourThought, Inc.

The Python standard library provides a minimal but useful set of interfaces to work with XML, including an implementation of the lightweight Simple API for XML (SAX) and a basic implementation of a core Document Object Model (DOM). A light-weight implementation of the Document Object Model Interface (xml.dom.minidom) module was used in this research work.

3.1 Proposed Framework

The logical architecture of the system we have developed is depicted in fig below. Let's discuss it from a data flow perspective.





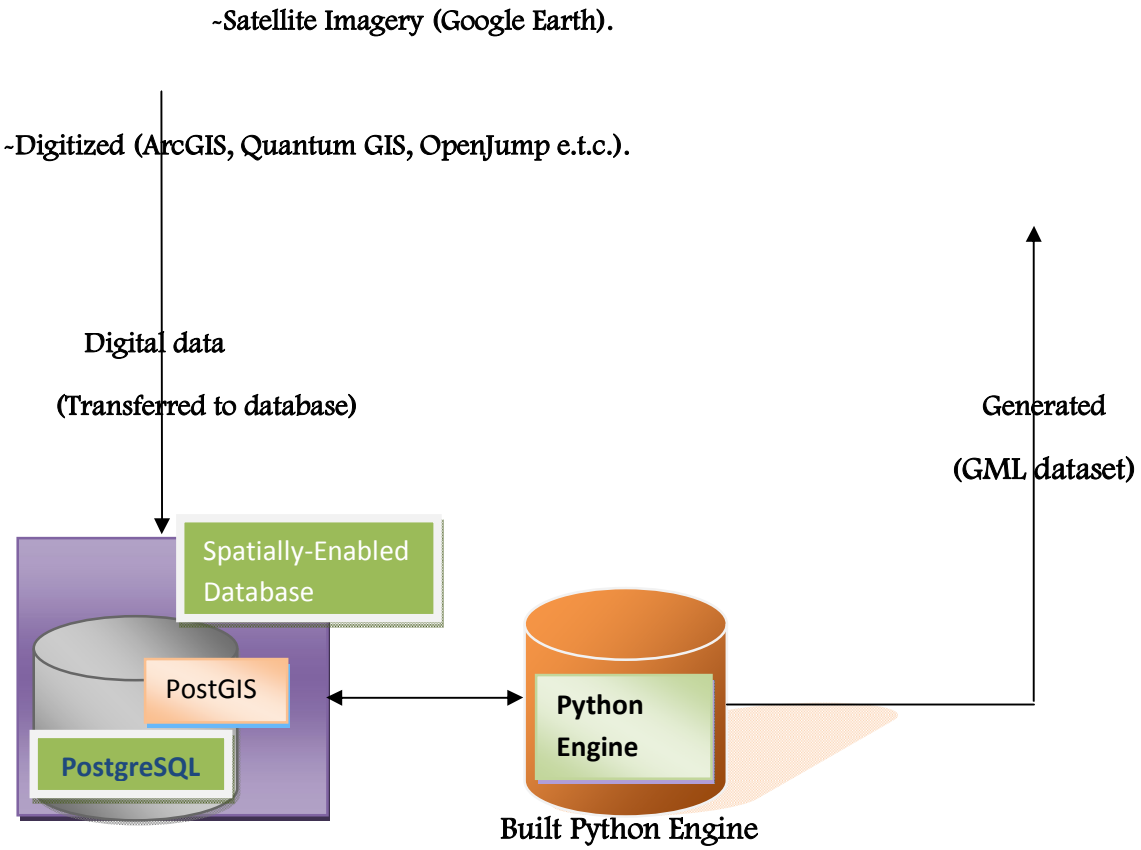


Figure 11.0: **A FRAMEWORK FOR RENDERING 3D GEOSPATIAL DATA IN GML**

The captured (satellite imagery) is represented by (1) in the figure above. Spatial (Geo-referencing) is done on it to extract registered points (coordinates) of features captured in the image in a process known as digitization. These points (coordinates) are now stored in a spatially enabled database as depicted in label 2 of the figure above , then an interface is provided between the built python engine labeled 3 in the figure and the spatially enabled database. The Python engine then reads the extracted coordinates from the database which is in Well-Known-Binary format (WKB) and translates it into its Well Known Text equivalent and finally renders it in Geography markup Language (GML). The rendered data is now imported into a visualize (GlobalMapper) for it visualization.

3.1.1 Study Area

A land cover by the Ahmadu Bello University (A.B.U), Zaria was taken as the area of study. Ahmadu Bello University (ABU) is the largest university in Nigeria and second largest in Africa, second only to Cairo University, Egypt. It is situated in Zaria-Nigeria with a geographical coordinate 11^{o} 09' 14.41" N, 7^{o} 39' 26.75" E. It was founded on October 4, 1962 as the University of Northern Nigeria. The study area was selected because of its proximity and physical visual interpretations for proper site survey. Performing a site survey is not fundamental to creating maps but can provide us with very useful contextual information. The more familiar you are with the detail of the area you are mapping, the less likely you are to make errors. Performing a local site survey will probably reveal important features you may want to eventually add to or use to enhance your map. You should treat the site survey as a way of collating all the information you think you will require to prepare an accurate 3D map of the area. The image below is a satellite imagery of the study area.



PLATE 1: Satellite Imagery of Ahmadu Bello University, Zaria from Google Earth

3.1.2 Encoding Geographic Information in GML

GML models the world, being based on a common model of geography called the OpenGIS abstract specification. According to the specification, two form of document are used to encode geographic information. They are: GML schema document and GML instance document.

GML schema documents: are XML/Schemas that define XML types and XML elements to encode GML objects with identity, elements to encode GML properties of those objects and XML attributes qualifying those properties. A GML object is an XML element of a type derived directly or indirectly from AbstractGMLType. From this derivation, a GML object may have a gml:id attribute. An element is a GML property if and only if it is a child element of a GML object. No GML object may appear as the immediate child of a GML object. Consequently, no element may be both a GML object and a GML property.

GML instance document: GML uses an explicit syntax to instantiate the Feature Model defined in the OGC abstract specification in an XML instance document. A Feature is encoded as an XML element with the name of the feature-type. A property of a feature is also encoded as an XML element whose name is the name of the property.

The result is a layered XML document in which XML elements corresponding to features, objects or values occur interleaved with XML elements corresponding to the properties that relate them.

Using the base XML schema described in the OGC specification it is possible to encode a wide variety of geospatial information such as; features with or without geometry, geometry, collection of features and associations between features (i.e. topologies).

> "A feature is an abstraction of a real world phenomena; it is a geographic feature if it is associated with a location relative to the earth."

Therefore the real world can be digitally represented by feature. The state of a feature is defined by a set of properties, where each property has a name, type and value triplet. The type is the class and properties are associations or attributes of the feature class in object modeling terminology.

A feature collection is a collection of features that can itself be regarded as a feature. Consequently, a feature collection has a feature type and thus may have properties of its own, in addition to the features it contains. The definition of feature types depends on the domain of application and the number of properties a feature may have is determined by its feature type.

The Feature schema, feature.xsd, which is one of the base schemas, provides a framework for the creation of GML features and feature collections. All GML feature collection must be derived by extension or restriction from gml:AbstractFeatureCollectionType which is defined as follows:

```
<complexType  name=”AbstractFeatureCollectionType” abstract=”true”>
    <complexContent>
        <extension base=”gml:BoundedFeatureType”>
            <sequence>
                <element  ref=”gml:featureMember” minOccurs=”0” maxOccurs=”unbounded”/>
                <element  ref=”gml:featureMembers” minOccurs=”0”/>
            </sequence>
        </extension>
</complexContent>
</complexType>
```

In an instance document the element might appear as:

```
<gml:FeatureCollection>
        <gml:featureMember>
        </gml:featureMember>

        <gml:featureMember>
        </gml:featureMember>
                -
                -
</gml:FeatureCollection>
```

Note: The property **gml:featureMember** encloses or references a single feature instance.

The gml:AbstractFeatureCollectionType is based on gml:BoundedFeatureType. This means that the gml:boundedBy property is mandatory in all GML feature collections. This is to ensure that spatial searching is supported without imposing the need to inspect every member of a collection. So, the value of this property defines the bounds of an envelop that encloses the entire feature which is a bounding box described by two point objects, the lower left coordinate and the upper right coordinate. Below is a sample document instance describing the gml:boundeBy property.

```
<gml:boundedBy>
  <gml:Box srsName="epsg:4326">
<gml:coord><gml:X>352431.9676587594</gml:X><gml:Y>1232681.331809967</gml:Y></gml:coord>
<gml:coord><gml:X>353443.3527047579</gml:X><gml:Y>1233626.285570063</gml:Y></gml:coord>
  </gml:Box>
 </gml:boundedBy>
```

Other properties that could form a child element of the gml: featureMember include: gml:description, gml: multiPolygon, gml:polgonMember, gml:Polygon gml:LinearRing and gml:coordinates.

-gml:description; the value of this property is a text description of the object. Below is a sample code from an instance document.

```
<gml:description> public Library</gml:description>
```

-gml:multiPolygon; a multipolygon is defined by one or more polygons, referenced through polygonMember element.

-gml:polygonMember; this property element either refrences a polygon via the Xlink-attribute or contains the polygon element.

- **gml:LinearRing;** this property element encapsulates a ring to represent properties in feature or geometry collections.

- **gml:coordinates;** this property element is intended to be used to record an array of tuples or coordinates describing the geometry of the feature.

Below is a sample GML code describing the above properties.

```
<gml:featureMember fid="1">
  <gml:description> public Library</gml:description>
  <gml:MultiPolygon>
    <gml:polygonMember>
      <gml:Polygon>
        <gml:outerBoundaryIs>
          <gml:LinearRing>
            <gml:coordinates>352877.02163887,1233618.83923531,5
352872.32998848,1233609.44478035,5  352867.693426132,1233612.15611458,5
352861.67354393,1233602.76770592,5 352841.814386368, 1233616.9818058,5
352848.495328903,1233625.69463921,5 352861.076768875,1233617.56668663,5
352866.429475784,1233626.28557014,5  352877.02163887,1233618.83923531,5
            </gml:coordinates>
          </gml:LinearRing>
        </gml:outerBoundaryIs>
      </gml:Polygon>
    </gml:polygonMember>
  </gml:MultiPolygon>
</gml:featureMember>
```

Also, it is important to note that every GML document should starts with the standard XML header in which the character encoding and XML version are mentioned then the root element of the GML document is appeared with all namespaces definitions used in the GML document and schema location.

3.1.3 3D Geospatial Data Storage Structure

Spatial databases are databases designed to store geospatial data. Vector geometries of objects on a map, spatial reference systems of spatial objects are examples of geospatial data stored in any spatial database (e.g. PostGIS). Coordinates for geometries may be 2D(X,Y), 3D(X,Y,Z), 4D(X,Y,Z,M) with an m value that is part of a linear referencing system or 2D with an m value (X,Y,M). 3D geometries are designed by a "Z" after the geometry type and geometries with a linear referencing system have an "M" after the geometry type.

PostgreSQL with a PostGIS extension was used in this research and it stores geospatial geometries in hexadecimal strings.

The first byte indicates the byte order for the data:

- `00` : big endian
- `01` : little endian

The second part is an integer for the geometry type, as described below:

Geometry types, and WKB integer codes				
Type	**2D**	**Z**	**M**	**ZM**
Geometry	0000	1000	2000	3000
Point	0001	1001	2001	3001
LineString	0002	1002	2002	3002
Polygon	0003	1003	2003	3003
MultiPoint	0004	1004	2004	3004
MultiLineString	0005	1005	2005	3005
MultiPolygon	0006	1006	2006	3006
GeometryCollection	0007	1007	2007	3007
CircularString	0008	1008	2008	3008
CompoundCurve	0009	1009	2009	3009
CurvePolygon	0010	1010	2010	3010
MultiCurve	0011	1011	2011	3011
MultiSurface	0012	1012	2012	3012
Curve	0013	1013	2013	3013
Surface	0014	1014	2014	3014
PolyhedralSurface	0015	1015	2015	3015
TIN	0016	1016	2016	3016
Triangle	0017	1017	2017	3017

Each data type has a unique data structure, such as the number of points or linear rings, followed by coordinates in double-precision floating-point format (double).

For example, the geometry POINT (2.0, 4.0) is represented as: 000000000140000000000000004010000000000000, where:

- 1-byte integer 00 or 0: big endian
- 4-byte integer 00000001 or 1: POINT (2D)
- 8-byte float 4000000000000000 or 2.0: *x*-coordinate

- 8-byte float 4010000000000000 or 4.0: *y*-coordinate

Though, these data are stored in hexadecimal format-Well Known Binary (WKB), with our transformation algorithm we were able to convert them to their denary representation-Well Known Text (WKT).

3.1.4 Transformation Algorithm

Below is the pseudocode used for the implementation of our Python Engine that is in charge of reading these stored geospatial data from our configured PostgreSQL+ PostGIS database, convert its data from its hexadecimal representation-Well Known Binary(WKB) to its denary representation-Well Known Text(WKT) then finally transforming the data into its GML equivalent.

*1. **Import** psycopg2 //Library to manage Database Access*
*2. **Import** sys //System Library*
*3. **Import** pprint //Formatted Printing Library*
*4. **Import** re //Regular Expression Library*
*5. **Import** string // Library to manage string functions*
*6. **From** xml.dom.minidom **import** Document //Library to manage xml Library*
*7. **Declare the** main ( ) **Function** //main function*
*a) **Declare** variable conn_string //conn_string holds database connection parameters*
*b) **Print** "Connecting to database\n->%s"*
*c) **try:** //Exception Handling*
*i) **Declare** a Variable **conn***
*ii) **Create** a new database session //a new connection instance is created*
*iii) **Create** a cursor object //to execute database commands*
*iv) **Execute** SQL Command with cursor object*
*v) **Declare** Variable records*
*vi) **Store** Database Operation Output in records*
*v) **Create** a document instance **doc***
*vii) **Declare** a node holder **gml***
*ix) **Create** element **gml: FeatureCollection***
*x) **Append gml: FeatureCollection** element to **gml** node holder*
*xi) **Set gml** attribute to "**xmlns: xsi**"*
*xii) **Set gml** attribute value to "**http://www.w3.org/2001/xml....**"*

xiii) ***Append gml*** *as root of document instance created above*
xiv) ***Declare*** *variable* ***e***
xv) ***Set e*** *to length of record list*
xvi) ***Repeat e*** *times*
1a) ***Declare*** *variable* ***record_to_list***
2a) ***Convert*** *each record content to a* ***list*** *data type*
3a) ***Declare*** *variable* ***i***
4a) ***Set i*** *to length of* ***record_to_list***
5a) ***for*** *(loopcounter=i)* ***do***
1b) ***fid=record_to_list[0]***
2b) ***description=record_to_list[1]***
3b) ***srid_multipolygon=record_to_list[2]***
4b) ***Declare*** *variable* ***coordinate_1***
5b) ***Set Coordinate_1*** *to* ***srid_multipolygon***
6b) ***coordinate_2*** *= formatted* ***coordinate_1****//replace string with* ***“”***
7b) ***coordinate_3*** *= formatted* ***coordinate_2****//replace all “(“with* ***““***
8b) ***cordinatesRegex*** *=* ***re.compile****(r"\b\d+\.?\d*\b”)*
9b) ***cordinates****=****cordinatesRegex.findall*** *(****coordinate_3****)*
10b) ***function*** *format(c)*
1c) ***Declare*** *empty list* ***result***
2c) ***Declare*** *variable* ***b***
3c) ***Set b*** *to 1*
4c) ***for (****loopcounter****<=****length(c)****)*** ***do***
Append c[i] *to* ***result***
if (b%3==0) do{
result.append *(" ")*
else:
result.append *(",")*
b*=****b****+1*
Return result
End do
End do
11b) ***End*** *function* ***format***
12b) ***Declare*** *variable* ***cord***
13b) ***cord****=****format*** *(****cordinates****) // call function format*
14b) ***Declare*** *variable* ***cord_1***
15b) ***cord_1****=string.* ***join*** *(****cord****, "")//store string output in single line*
16b) ***Declare*** *a node holder* ***featureid***
17b) ***Create*** *element* ***gml: featureMember***
18b) ***Append gml: featureMember*** *to* ***featureid***
19b) ***Set featureid*** *attribute to “****fid****” and value as* ***str****(fid)*
20b) ***Append featureid*** *to the* ***gml*** *node*
21b) ***Declare*** *a node holder* ***describe***
22b) ***Create*** *an element* ***gml: description***
23b) ***Append gml: description*** *to* ***describe***
24b) ***Append describe*** *to the* ***featureid*** *node*
25b) ***Declare*** *a node holder* ***dtext***

26b) ***Create*** *a document text node description*
27b) ***Append*** *the* ***dtext*** *node to* ***describe*** *node*
28b) ***Declare*** *a node holder* ***Multipolygon***
29b) ***Create*** *a document element* ***gml: Multipolygon***
30b) ***Append*** *the* ***Multipolygon*** *node to* ***featureid*** *node*
31b) ***Declare*** *a node holder* ***Poly_member***
32b) ***Create*** *an element* ***gml: polygonMember***
33b) ***Append*** *the* ***Poly_member*** *to* ***Multipolygon***
34b) ***Declare*** *a node holder* ***poly***
35b) ***Create*** *an element* ***gml: Polygon***
36b) ***Append*** *the* ***poly*** *node to* ***Poly_member*** *node*
37b) ***Declare*** *a node holder* ***boundary***
38b) ***Create*** *an element* ***gml: outerBoundaryIs***
39b) ***Append*** *the* ***boundary*** *to* ***poly*** *node*
40b) ***Declare*** *a node holder* ***linear_ring***
41b) ***Create*** *an element* ***gml: linearRing***
42b) ***Append linear_ring*** *to* ***boundary*** *node*
43b) ***Declare*** *a node holder* ***coord***
44b) ***Create*** *element* ***gml: coordinate***
45b) ***Append*** *the* ***coord*** *node to* ***linear_ring***
46b) ***Declare*** *a text holder* ***ptext***
47b) ***Append coord_1*** *to* ***ptext***

6a) ***Declare*** *variable* ***f***
7a) Set ***f*** *to* ***doc***
8a) ***Open*** *output file* ***wor.gml***
9a) ***Declare*** *variable* ***a***
10a) ***Assign*** *variable* ***a*** *to* ***wor.gml***
11a) ***Write f*** *into* ***a***

d) ***Except:***
1b) ***Exit*** *sys with* ***failure*** *report*

8. ***if*** *(main())* ***exit***

3.2 Handling other XML base Data Structures

Although, our framework is limited to rendering of geospatial data in GML data structure, with a little extension in our Python Engine algorithm we should be able to render this geographic information in other XML-base data structure capable of holding geospatial data. Examples of XML-base dialect capable of holding geospatial data include: CityGML-which is an extension of GML to describe different levels of details of geographic features, X3D (Extensible 3D)-which is an XML graphic description language in 3D e.t.c.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

In this chapter, we discuss the result of implementation of our framework; the processes involved starting from the data collection stage, data storage and the algorithm used by the python engine to read points (coordinates) for each feature from the database and subsequent rendering in XML/GML format then finally the visualization of the generated GML data.

4.0 Data Collection

A satellite imagery capturing features of the Ahmadu Bello University, Zaria was gotten from Google Earth. An area of 697,272.356 sq m was digitized, depicting major features like; Major Hostels for students (Amina Hall, Suleiman Hall & Ribadu Hall), the universities senate building, departments in faculty of science & engineering, centers like the Iya Abubakar Computer center, KIL & public Library, the University Health care services(Sick Bay), convocation square e.t.c.

The starting point for your 3D map is to first and foremost generate the 2D representation of the area of study through a process known as digitization. The result of digitizing our satellite imagery using ArcGIS is a 2D footprint representation of features we want in our map. The figure below shows the building footprints of an area covered by the Ahmadu Bello University, Zaria.

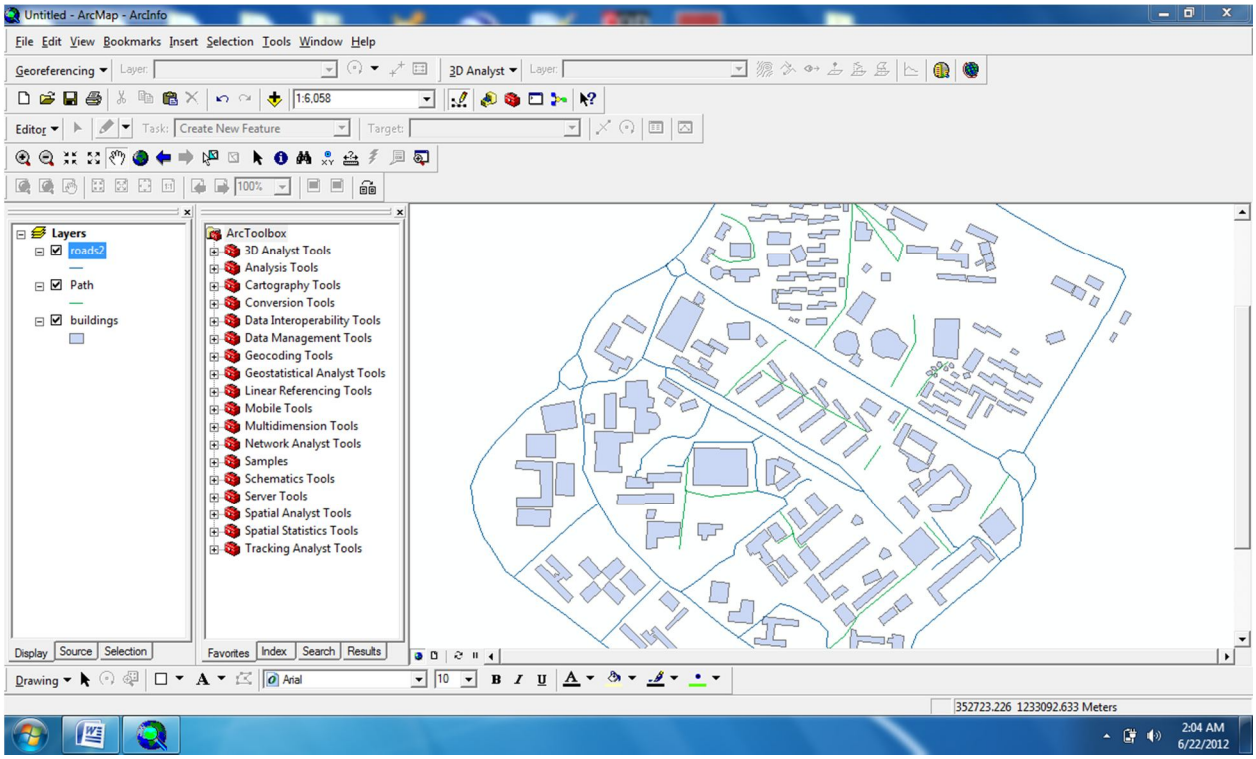


Figure 12.0: 2D Digitized Image of ABU, Zaria

Next, we determined the height of the buildings and add that to the 2D data attributes. Building heights can be derived from: lidar data, estimated height-by measuring and counting a repetitive building material such as: bricks (multiplying the value by the repetition to get the overall height of each building) or assigning a nominal height per each building level (e.g. 3 meters per level). The third approach was taken for this research. After we have determined our height data, we added a new height field to our 2D footprint attribute table and add the data in an ArcMap edit session. The figure below shows the height in meters for the buildings in the university campus dataset.

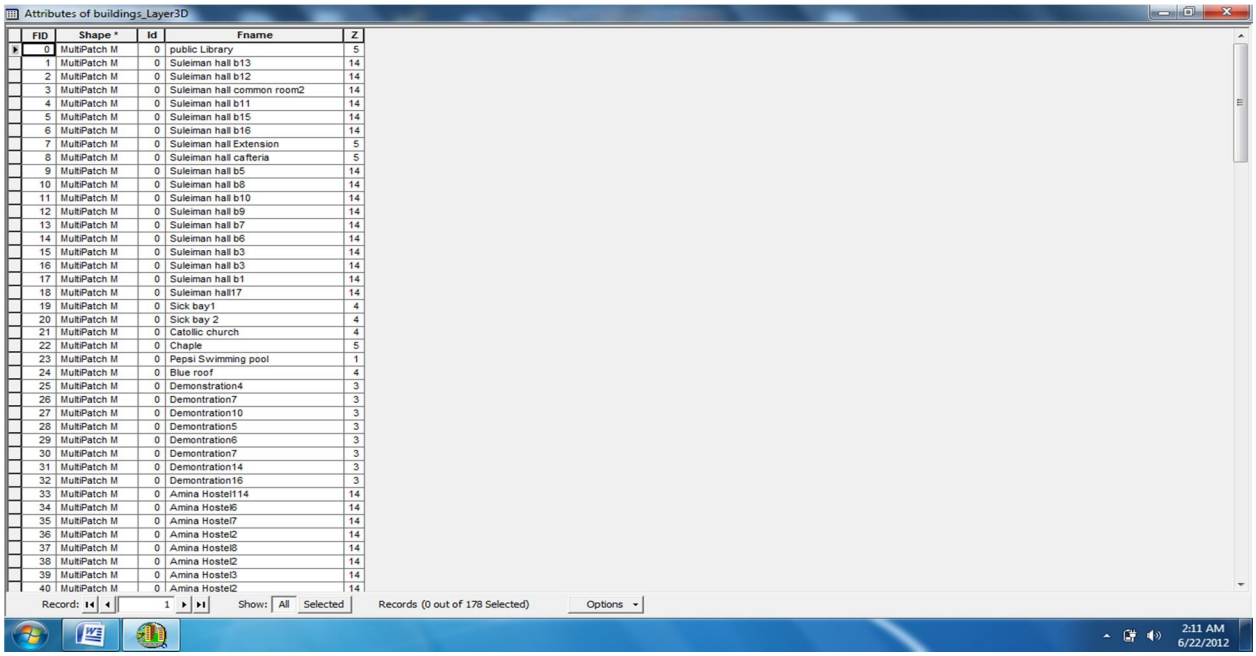


Figure 13.0: Attribute Table showing information of the Digitized Image of ABU, Zaria.

Then, using ArcScene a component of the ArcGIS tool which is similar to ArcMap in many ways but with a crucial difference in that you can visualize features in 3D by providing height information from feature geometry, feature attributes, layer properties or a defined 3D surface (such as a digital elevation model or DEM). This is done using the height data added as an attribute.

In ArcScene, add your building footprint feature class to the table of contents. Now we can use our height information to extrude the building footprints. To do this:

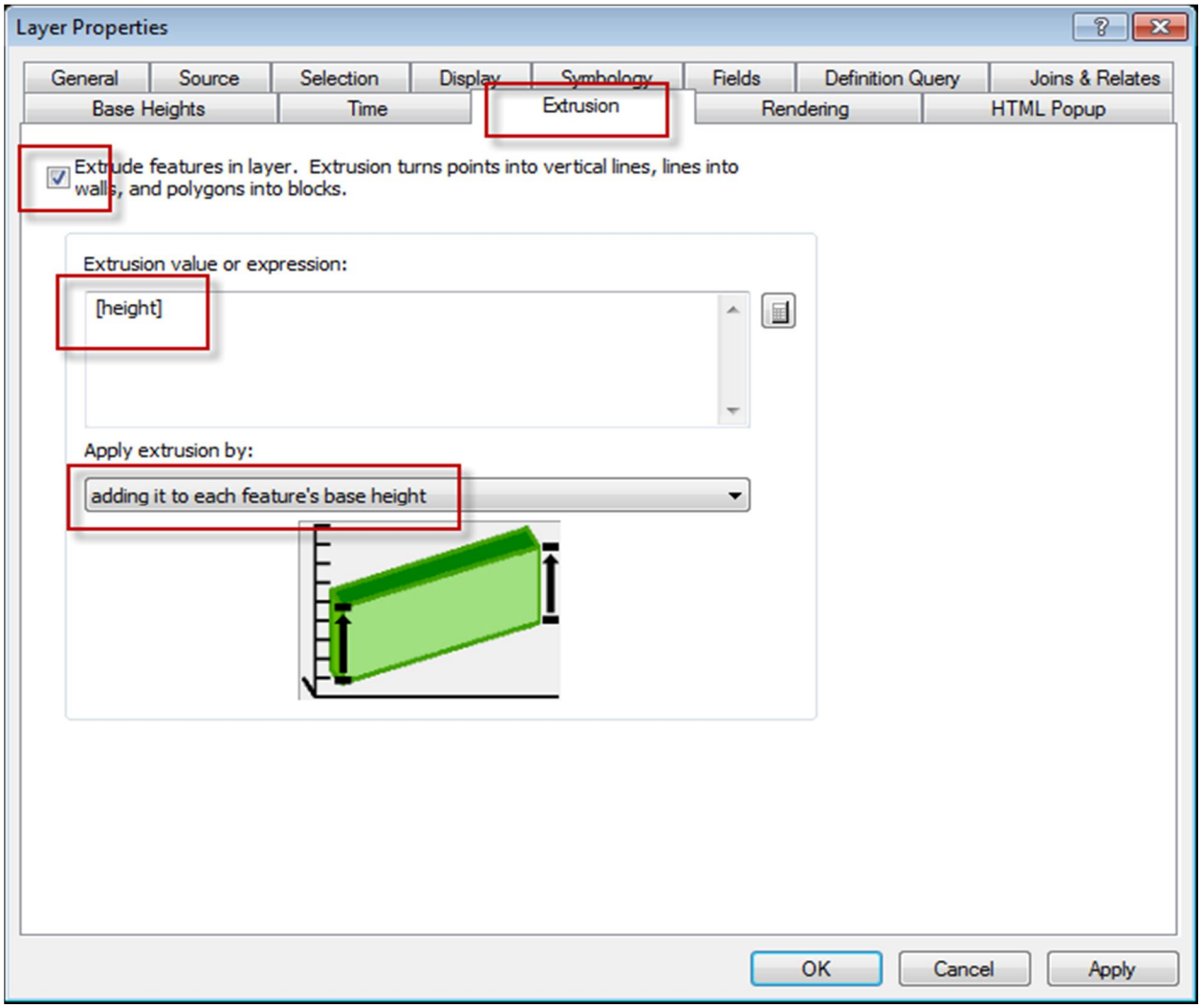
-Right-click the building feature class in the table of contents.

-Select Properties.

-In the Layer properties dialog box;

- Click the tab Extrusion

- Check the extrude features in the layer box, then select the field that contains your height attribute as an extrusion value (in the figure below, this is [height].).
- Then toggle the apply extension by drop-down box, select adding it to each feature's base height and click OK.



Our 2D building footprints will now be extruded to the height of each building to create 3D representations for each building as shown in the figure below.

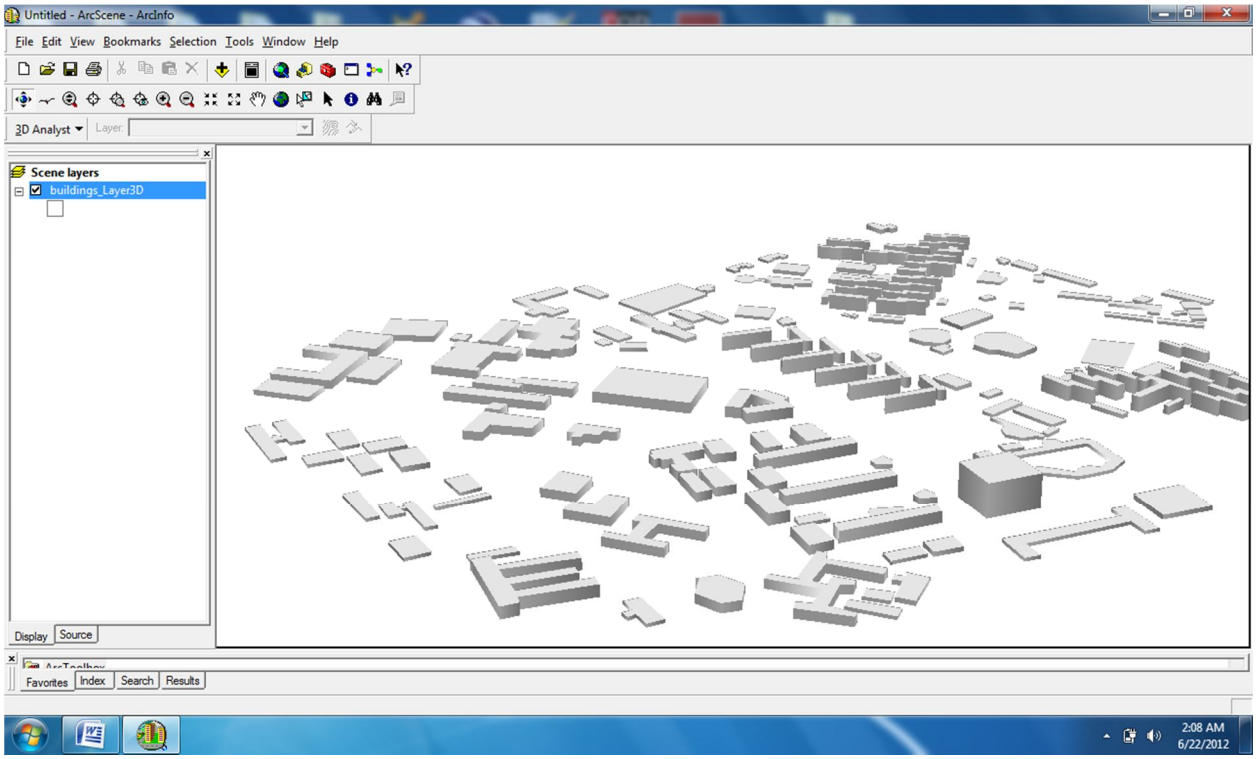


Figure 14.0: 2D to 3D Conversion of Digitized Image of ABU, Zaria

4.1 Data Storage

These data are stored in a format termed as shapefiles. These shapefiles are then imported into the postgreSQL database with a PostGIS extension to enable it store spatial data. A screen capture of the postgreSQL database management system is as shown below.

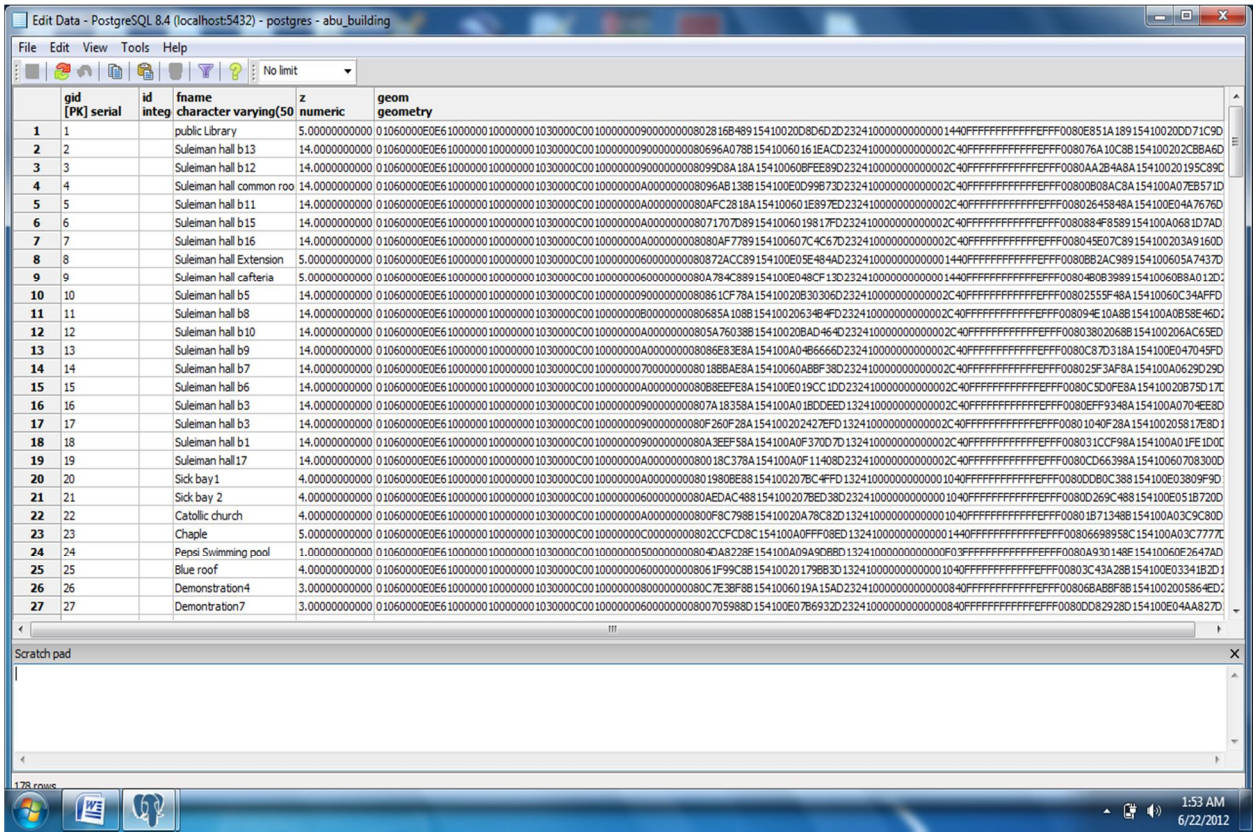


Figure 15: PostgreSQL Database with Stored information from the Digitized Image of ABU, Zaria

4.2 Generated GML data

Below is a fragment our generated GML data from our Python Engine representing an area covered by the Ahmadu Bello University, Zaria.

```
<?xml version="1.0" ?>
<gml:FeatureCollection  xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<gml:boundedBy>
   <gml:Box srsName="epsg:4326">

<gml:coord><gml:X>352431.9676587594</gml:X><gml:Y>1232681.331809967</gml:Y></gml:coord>

<gml:coord><gml:X>353443.3527047579</gml:X><gml:Y>1233626.285570063</gml:Y></gml:coord>
   </gml:Box>
 </gml:boundedBy>
 <gml:featureMember fid="1">
   <gml:description> public Library</gml:description>
   <gml:MultiPolygon>
     <gml:polygonMember>
       <gml:Polygon>
         <gml:outerBoundaryIs>
           <gml:LinearRing>
             <gml:coordinates>352877.02163887,1233618.83923531,5
352872.32998848,1233609.44478035,5  352867.693426132,1233612.15611458,5
352861.67354393,1233602.76770592,5 352841.814386368, 1233616.9818058,5
352848.495328903,1233625.69463921,5 352861.076768875,1233617.56668663,5
352866.429475784,1233626.28557014,5  352877.02163887,1233618.83923531,5
             </gml:coordinates>
           </gml:LinearRing>
         </gml:outerBoundaryIs>
       </gml:Polygon>
     </gml:polygonMember>
   </gml:MultiPolygon>
 </gml:featureMember>
 <gml:featureMember fid="2">
   <gml:description> Suleiman hall b13 </gml:description>
   <gml:MultiPolygon>
     <gml:polygonMember>
       <gml:Polygon>
         <gml:outerBoundaryIs>
           <gml:LinearRing>
             <gml:coordinates> 352961.853918076,1233580.11752892,14 352963.157678604,
1233574.73114204,14  352940.574705124,1233574.16129875,14
352938.557882309,1233568.79001427,14  352915.316907883,1233569.56830025,14
352916.008558273,1233575.61816597,14  352933.939668655, 1233575.5365963,14
352935.295434952,1233581.5834446,14  352961.853918076,1233580.11752892,14
             </gml:coordinates>
           </gml:LinearRing>
         </gml:outerBoundaryIs>
       </gml:Polygon>
     </gml:polygonMember>
   </gml:MultiPolygon>
 </gml:featureMember>
 <gml:featureMember fid="3">
   <gml:description> Suleiman hall b12 </gml:description>
```

4.3 Visualization of the Generated GML

The wealth of geospatial data cannot be fully valued when information implicit in data is difficult to discern. This confronts geographic information scientists with an urgent need for new methods and tools that can intelligently and automatically transform geospatial data into information and, furthermore, synthesize geographic knowledge. In this work we present a tool (GlobalMapper) which is able to read geographic data in GML and graphically visualize it.

The figure below depicts the visualization of the generated GML data.

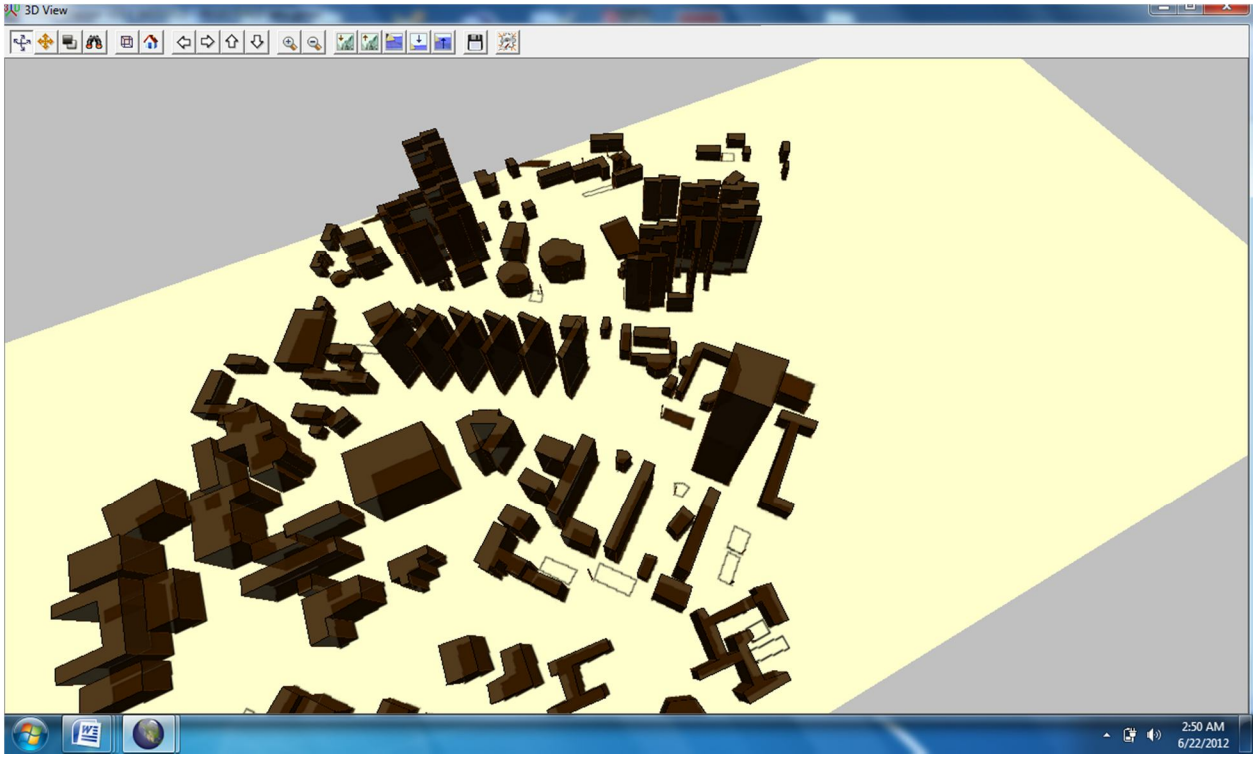


Figure 16: 3D Visualization of the generated GML Data.

CHAPTER FIVE

SUMMARY AND FUTURE WORK

This chapter presents our summary and directions for future work.

5.0 Summary

Since, virtually all information required for planning and national development is spatial in nature, then for easy accessibility such information must be produced in a coordinated and consistent manner and distributed harmoniously to all who need them.

Inventing open data format and going towards interoperability has helped data owners to prepare their data for all users with different needs and different software. Markup languages with their characteristics make it easy to define any kind of data without the need to have any particular software. So spatial data is no exception. The creation of a new standard data-exchange format-Geography Markup Language (GML) represents another important step taken by the geospatial community towards data interoperability. GML is developed as Data Exchange Standards Interface by Open GIS Consortium (OGC) to achieve data interoperability and reduce costly geographic data conversions between different systems. GML holds promise to support mapping from a wide variety of sources and enable sharing of geospatial data for on-line information exchanges.

The Nigeria's National Geo-spatial Data Infrastructure which is responsible for the collection of Policies and standards for data acquisition, integration and sharing is faced with problems of its full implementation due to non-existence of digital geospatial datasets in the country.

According to the research objectives, this research is aimed at generating digital geospatial datasets of an area covered by the Ahmadu Bello University, Zaria, this becomes necessary as we were originally working on visualization but no digital datasets in 3D was available for our utilization and also haven discovered the advantages attached to GML data modeling format for geographic information, a framework was also designed to render this digital dataset in a GML format using Python. With our framework, we were able to extract Geo-spatial information from the satellite imagery of an area covered by the Ahmadu Bello University, Zaria, then store this information in a spatially enabled database interfaced with our Python engine which now renders this geo-spatial information in GML. The basic idea is to render this geographic information in a unique environment (the Web) that will make this data portable, accessible, maintainable and interoperable.

The thesis contribution to knowledge include: the development of digital dataset for an area covered by the Ahmadu Bello University, then the development of a framework for encoding digital datasets into GML format and finally, building a Python engine that defines an algorithm for the transformation of stored digital geo-spatial information into 3D GML format.

5.2 Future Work

- A web service can be designed to interface with the engine to provide access to the database for both querying and updating facility.
- An implementation of an XML/GML based database to further enhances interoperability.

- The python engine could be extended to generate geographic information in XML base graphic representation (i.e. X3D). X3D is the best choice to work with for 3D

visualization, it can be dynamically modified and is directly visualized in the browser with an X3D plugin.

- Since GML as other markup languages does not define presentation methods, these generated GML could be styled to XML related graphics representation like X3D, SVG or VRML.
- The whole idea could be enhance to support real-time geospatial data rendering and visualization.
- Conversion of satellite imagery directly into XML format without any intermediate tool.

REFERENCES

1. A. Cuthbert, R. Lake, S. Cox and R. Martell (2000). Open GIS Consortium. Geography Markup Language (GML) version 1.0, OpenGIS Implementation Specification. http://www.opengis.net/gml/GML.html, May, 2000.

2. Alan Reginald (2003) Geography and History. Cambridge University Press.

3. Bertolotto and Egenhofer (2001). Progressive Transmission of Vector Map Data over the World Wide Web. GeoInformatica. Vol.5, no. 4, PP. 345-373.

4. Choicki, J. (1999). Constraint- based Interoperability of Spatio-temporal Databases. Geoinformatica. Vol. 3, no.3, PP. 211-243.

5. Curtis (2003). XML Schema: Reconciling Diversity with Standardization. http:// www.gmldev.org/GMLdev2002/presentations/

6. Clemens (2002). The Elegance of XML-based Mapping Proceedings of URISA 2002. Annual Conference and Exposition Chicago, Illinois. Oct.2002. PP. 320-324.

7. Devogele, T., Parent, C. and Spaccapietra, S. (1998).On Spatial database integration. International Journal. Geographical Information Science, 1998, vol. 12, no.4, PP.335-352.

8. De Vries et al (2003), Accessing a 3D geo-DBMS using Web technology, In Proceedings of the ISPRS WG 11/S, IV/I and IV/2 joint workshop on spatial, temporal and multi-dimensional data modeling and analysis (PP. 1-8)

9. Dooling (2002), XSQL, sourceForge, (http:// xsql.sourceforge.net)

10. Elliotte, R.H.(2001) Hungry Minds-XML Bible, 2nd Edition, PP. 7

11. Ferg S. (2006), Python and Java: side by side comparision. http://www.ferg.org/projects/python-java-side _by_side.html

12. Geometric aspects of mapping: map projections. http://kartoweb.itc.nl/geometrics/Map Projections/body.htm

13. Hilary M.Hearnshaw and David J. Unwin (1994). Visualization in Geographical Information Systems. John Wiley & Sons, 1994.

14. Igbokwe, J. I and Ono, M.N. (2005) Nigeria's National Geo-spatial Data Infrastructure: Problems and Prospects, From Pharaohs to Geoinformatics, FIG Working Week 2005 and GSDI-8 Cairo, Egypt April 16-21, 2005.

15. Kim K., Lee K, Lee H.G., and Ha Y.L. (1998), Virtual 3D GIS's Functionalities Using Java/VRML Environment, Proceedings of the Earth Observation & Geo-Spatial Web and Internet Workshop' 98, Salzburg, 1998, Austria.

16. Lake, A. (2002). Will GML enable an accessible Geo-Web? GEEO World, July 2002. http://www.geoplace.com/gw/2002/0207/0207gml.asp

17. Markus Schneider Tutorial "Spatial data types: Conceptual foundation for the design and Implementation of spatial database systems and GIS", In the 6th international symposium on advances in spatial databases, springer-verlag, 1999.

18. Map Projection overview. http://www.colorado.edu/geography/gcraft/notes/mapproj /mapproj.html, September, 2005.

19. Meneghello, M. (2001). XML: The New Language of Data Exchange, Cartography, vol. 30, No. 1.

20. Johnsen (2003).The Map Project. http://www.webmapping.org/ GMLIntroduction. Html

21. Morcrette (1999), VRML Generation tools for visualization of database content in three dimensions. Published M.Sc. thesis, Massachusetts institute of technology.

22. Murray and Chow (2002). GML User perspectives. Proceedings of URISA 2002 Annual Conference and Exposition Chicago, Illinois. Oct. 2002. PP. 324-332.

23. Nigel Walford. Geographical Data: Characteristics and Sources. Prentice-Hall, inc., 2001.

24. OGC Abstract Specification (2001). http:// www.opengis.net/gml/01-029/GML2.html

25. OGC (2003) http:// www.opengis.org/techno/documents/02-023r4.doc

26. Phillipe, R. (2002). Spatial Databases with Application to GIS. Morgan Kaufmann Publishers, San Francisco, 2002.

27. Peng and Tsou (2003). Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks, New York. John Wiley & Sons.

28. R. Lake (2001). Galdos Systems Inc. Making Maps for the Web with GML. http//www.geojava.com/company/galdos/articles.

29. Shashi, S. (1997) Data models in geographic information systems. In communications of ACM, PP. 103-111.

30. Simple polygon. http://en-wikipedia.org/wiki/simple_polygon. October, 2006.

31. Shekhar, S., Vatsavai, R. R., Sahay, N., Burk, T. E., and Lime, S. (2001), WMS and GML based interoperable web mapping system. Proceedings of the ninth ACM international symposium on advances in geographic information systems. Atlanta, Georgia, U.S.A 2001. PP. 106-111.

32. Smith, C., Aberline, P. and Bellman, C. (2002) Investigation and Prototyping of Emerging Web Standards for Web GIS, Cartography, vol. 31, No. 1.

33. Sun Microsystems (2007), Java SE Technologies-Database, http://java.sun.com/javase/database/

34. Tarnoff, P. J. (1998). Evolution or Control. Traffic Technology International. PP. 32-35.

35. Van Rossum (1999). Python Workshop, Corporation for National Research Initiatives, Virginia.

APPENDIX ONE

```
import psycopg2
import sys
import pprint
import re
import string
from xml.dom.minidom import Document
def main():
        conn_string = "host='localhost' dbname='postgres' user='postgres'
password='computer'"

        print "Connecting to database\n        ->%s" % (conn_string)
        try:
                conn = psycopg2.connect(conn_string)
                cursor = conn.cursor()
        cursor.execute("SELECT gid,fname,asewkt (force_3d(geom))FROM
abu_building")
        records = cursor.fetchall()

                doc = Document()

                gml = doc.createElement("gml:FeatureCollection")
                gml.setAttribute("xmlns:xsi","http://www.w3.org/2001/XMLSchema-
instance")
                gml.setAttribute("xmlns:gml","http://www.opengis.net/gml")
                doc.appendChild(gml)
```

```
        for e in range(len(records)):
            #change each content to a list to enable manipulation
            record_to_list=list(records[e])
            for i in record_to_list:
            #the first item in the newly created list is always a feature id
                fid=record_to_list[0]
                description=record_to_list[1]
                srid_multipolygon=record_to_list[2]
                        #coordinates=srid_multipolygon[25:-3]
            #cleanup string 3 int each list
coordinate_1=srid_multipolygon.replace("SRID=4326;MULTIPOLYGON","")
                coordinate_2=coordinate_1.replace("(","")
                coordinate_3=coordinate_2.replace(")","")
                #coordinate_4=coordinate_3.replace(" ",",")
                #coordinate_5=coordinate_3.replace(","," ")

                cordinatesRegex = re.compile( r"\b\d+\.?\d*\b" )
                cordinates=cordinatesRegex.findall( coordinate_3 )

                def format(c):
                    result=[]
                    b=1
                    for i in range(len(c)):
                        result.append(c[i])
```

```
                    if b % 3 == 0:
                        result.append("  ")
                    else:
                        result.append(",")
                    b=b+1
                return result
            cord=format(cordinates)
            cord_1=string.join(cord, "")

        featureid = doc.createElement("gml:featureMember")
        featureid.setAttribute("fid",str(fid))
        gml.appendChild(featureid)

        describe = doc.createElement("gml:description")
        featureid.appendChild(describe)

        dtext = doc.createTextNode(description)
        describe.appendChild(dtext)

        Multi_polygon=doc.createElement("gml:MultiPolygon")
        featureid.appendChild(Multi_polygon)

        poly_member=doc.createElement("gml:polygonMember")
        Multi_polygon.appendChild(poly_member)
```

```
                poly=doc.createElement("gml:Polygon")
                poly_member.appendChild(poly)

                boundary=doc.createElement("gml:outerBoundaryIs")
                poly.appendChild(boundary)

                linear_ring=doc.createElement("gml:LinearRing")
                boundary.appendChild(linear_ring)
                coord = doc.createElement("gml:coordinates")
                linear_ring.appendChild(coord)
                ptext = doc.createTextNode(cord_1)
                coord.appendChild(ptext)
            f=doc.toprettyxml(indent="  ")
            a=open("wor.gml","w")
            a.write(f)
            #print doc.toprettyxml(indent="  ")
        except:
            exceptionType, exceptionValue, exceptionTraceback = sys.exc_info()
            sys.exit("Database connection failed!\n ->%s" % (exceptionValue)
if __name__ == "__main__":
        sys.exit (main ())
```