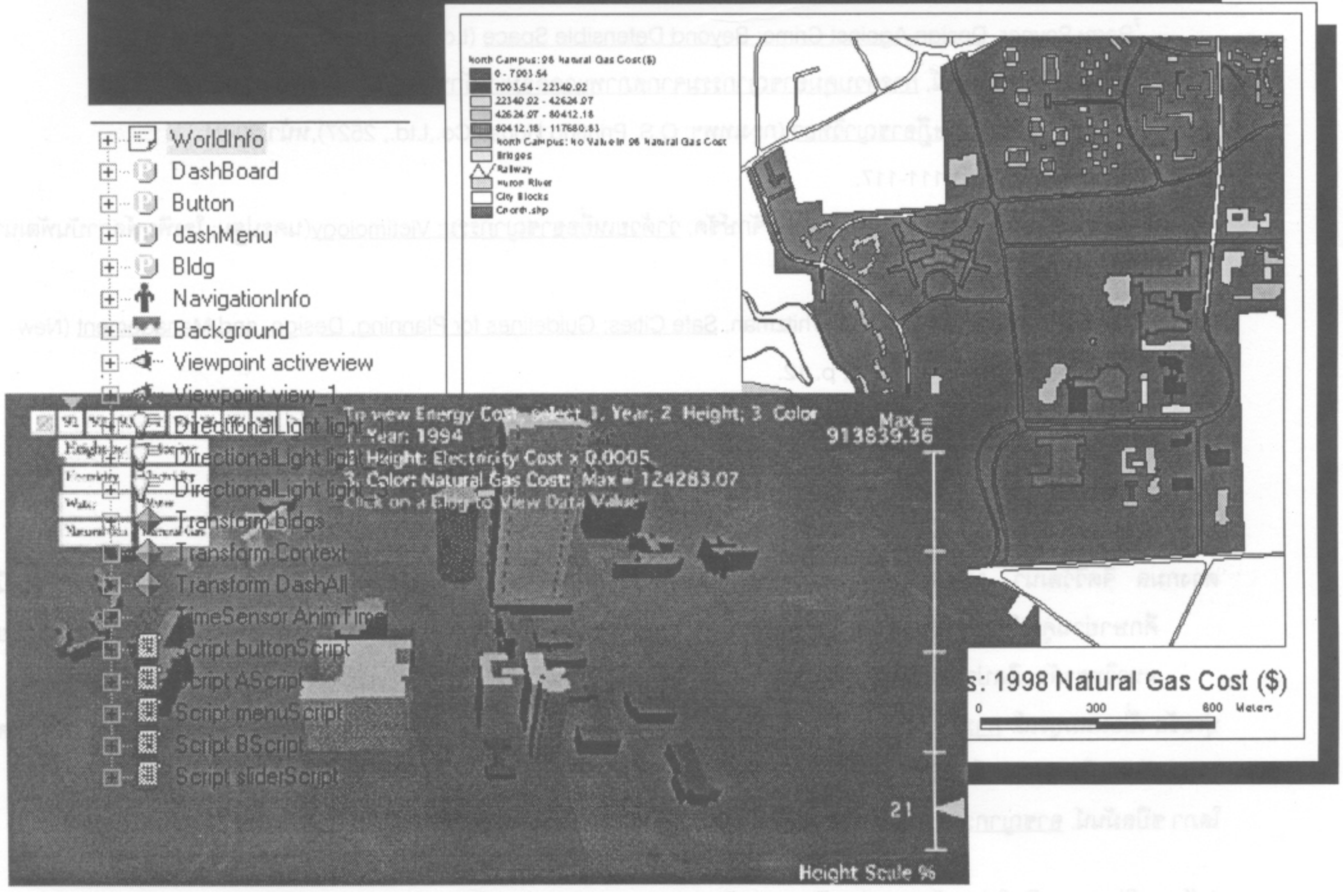
Visualization of Spatial/Geographic Information with Virtual Reality Modeling Language (VRML)

Report on the World Wide Web at:
http://www-personal.engin.umich.edu/~tnac/vrml/GISVisualization/html/reportMaster.html
VRML model is accessible at:
http://www-personal.engin.umich.edu/~tnac/vrml/GISVisualization


To view Energy Cost, select 1. Year; 2 Height; 3 Color
1. Year: 1994
2. Height: Electricity Cost x 0.0005
3. Color: Natural Gas Cost; Max = 124283.07
Click on a Bldg to View Data Value

Thana Chirapiwat
Urban, Technological, and Environmental Planning
Taubman College of Architecture and Urban Planning
University of Michigan, Ann Arbor
February 25, 2001

Visualization of Spatial/Geographic Information with Virtual Reality Modeling Language (VRML)

Background

Geographic Information System (GIS) is a system that enables an integration of a spatial data (maps, drawings) and databases. The integration of the two types of data sets create relational database. GIS is capable of analyzing the complex multivariate data sets and visualizing the data in mainly 2D thematic maps. There is a module that can be added on to add 3D capability. However, it has no effective navigation system allowing users to explore the data representation. The level of control over visual appearance available in many commercial GIS is rather limited (Gahegan, 2000:256). Often, it is restricted to small number of predefined spatial primitives with few visual attributes. Here is where the Virtual Reality Modeling Language (VRML) can be implemented to enhance the visualization of the GIS analysis. The problem for such application with VRML involves the process of transferring, transforming, and formatting the data sets so that the final visualization truly represents the integrity of the data.

There have been a number of developments to generate 3D model of geographic data such as terrain model for real time viewing with some degrees of navigation freedom. Huang and Lin (1999) developed a script, using Avenue Script, called GeoVR for ArcView to generate VRML model from conventional 2D vector data, ArcView's shape files, using an HTML's Common Gateway Interface (CGI) form as a front-end to receive parameters defined by users. The VRML model of the geographic information created with this method is still a static model with the degrees of navigation freedom provided by the browser's VRML plug-in. Thus, the interactivity in the model, besides the real-time navigation, does not exist. Similarly, other VRML models of geographic information often lack the abilities to interact with the model to explore different aspects of the data associated with the geographic entities. For example, the United State Environmental Protection Agency's (EPA) Geographic Information Systems and Visualization Center (GISVIS, http://www.epa.gov/gisvis/index.html) has applied the VRML to visualize a number of their geographic data with static VRML models. Berry et al (1998) apply 3D virtual reality to visualize realistic visualization of natural environment with static 3D model with real-time navigation. Their visualization of virtual forest was the implementation of the integration of GIS and CAD more than the integration of GIS and VR in the sense that the process was in the CAD modeling of the GIS data and using VRML as a mere viewing tool. The interface of their model was limited to navigation in and around the model. This paper explores the integration of the GIS data within the VRML model that can be explored interactively as the user navigates through the virtual environment.

The basic implementations of the 3D virtual environment using VRML standard have been largely on the simulation and modeling of places, real and virtual. VRML world is a dynamic and interactive 3D environment that has not been fully utilized in interactive visualization of geographic information. Objects in the VRML world can be self-activated animated or moved, rotated by user input. Moreover, complex behavior can be assigned to objects so that it reacts to user input. These capabilities provide possibility to integrate database information to the VRML world so that the data can be visualized in 3D environment. The ability to visualize complex data in 3D can greatly help us to analyze and understand the underlying features of the data.

Objectives

The objectives for this study are:

- To study the requirements and advances in interactive visualization of geographic information in 3D virtual environment.
- To investigate properties of VRML geometry and interactivity functions.
- To identify an effective way to visualize the integrated database resulting from the GIS spatial analysis with VRML. This includes ways of the transformation and conversion of file formats from different components to the final VRML programming.
- To prepare a VRML prototype that will allow future modifications and customization of a case study on energy consumption data for the University of Michigan's North Campus buildings.

3D Visualization Techniques for Geographic Information

In general, GIS produces visualization graphics in map format. These graphics can be categorized broadly as exploratory graphics, design graphics, reference graphics, and presentation graphics. Exploratory graphics portray the information generated from numerical modeling, especially where there is need to simplify for less ambiguity or more convincing. Design graphics are graphics used in the thinking process. They permit preliminary testing and comparison of solutions to a set of problems. Reference graphics are graphics prepared for a variety of purposes which can be extracted and put to new uses. Presentation graphics may be similar to that of reference graphics, but are usually simplified graphics designed to communicate specific concepts in a particular context (Turk, 1994: 28-29).

The visualization of geographic information is the integration of Visualization in Science Computing (VisSC) principles and cartographic representation methods (MacEachren, 1994). MacEachren et al (1994) introduces four dimensions of variability in typology of visualization techniques. The first is *purpose*—how the visualization tools are used: data exploration and confirmation (prompt for visual thinking) to the synthesis and presentation (vehicles for communication). Second dimension is the *levels of interactivity* to the audience. The third dimension is the *levels of abstraction* in the visualization. This ranges from the highly symbolic to the highly detailed and realistic. Abstract representations may not be accessible to untrained users, thus it is more private in nature. The fourth, and final, dimension is the *aspect of the phenomenon to information process*. This 4-dimension aspect of visualization can be illustrated in 3-dimension space in Figure 1 below.

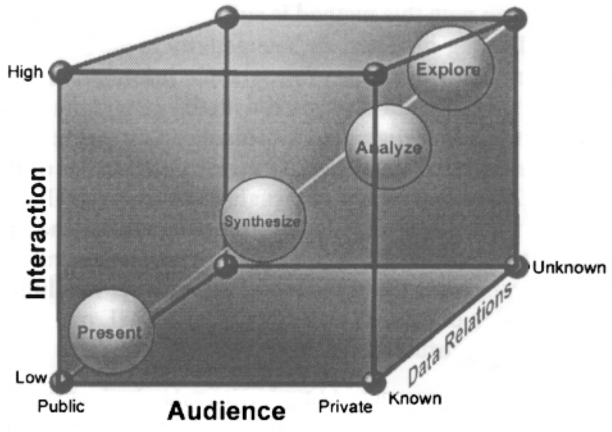


Figure 1. Use of geographic information visualization in 3D space of (1) audience correlating with the levels of abstraction, (2) levels of interactivity, and (3) data relations to the aspect of phenomena. (Source: adapted from MacEachren, 1998)

This study of the visualization of geographic and spatial information is expected to benefit the exploratory aspect of data visual analysis. The prototype would facilitate the following analyses:

- Phenomena change. 3D visualization can depict phenomena change over some specific of time period, or the rate of change of a phenomenon or one (or more) of its attributes.
- Visualizations of relationships between phenomena, such as spatial distribution and pattern of one of the attributes, relationships across various attributes.

The 3D environment allows us to examine the multivariate data with high dimension. For example, the prototype from the case study can display at least five dimensions of the data. The first two dimensions represent the geographical locations and the spatial extends of the objects of study. The third dimension, height, can show the value of one of the attributes of the spatial objects. The fourth dimension, color, represents another attribute value. The fifth dimension is the temporal dimension enabled by interactive menu to display data at different times. The high dimensions of the data visualization enable the visual analysis of the relationship among various attributes and among objects.

Model for the Data Integration

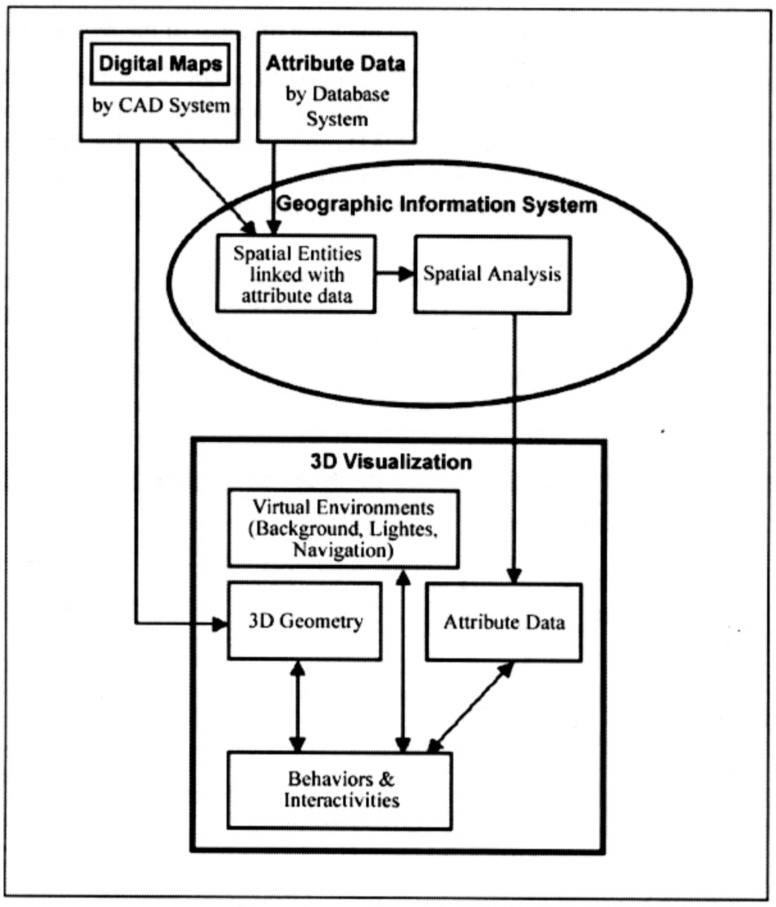

Digital Maps
by CAD System
Attribute Data
by Database System
Geographic Information System
Spatial Entities linked with attribute data
Spatial Analysis
3D Visualization
Virtual Environments (Background, Lightes, Navigation)
3D Geometry
Attribute Data
Behaviors & Interactivities

Figure 2. Model for the data Integration.

The implementation of VRML to visualize geographic information can be structured to integrate the four systems—CAD, database, GIS, and VRML (see Figure 2). The model starts with 2 types of data from different systems. CAD system provides spatial and geometric information in the form of digital map and prototype of 3D spatial objects. The prototype of 3D spatial objects was prepared base on the 2D spatial and geometric properties of the objects with a single unit, 1 meter, of the third dimension. This is to set the third dimension of the spatial objects ready to correspond to any value of the non-spatial attributes from the database system and any modifier. The 2D properties of the spatial objects are the input to the GIS. They are, then, linked with non-spatial attributes for spatial analysis. Spatial queries can be performed to filter only interested attributes to be further analyzed and visualized.

The following sections describe the elements of the four major modules of the model. The focus of this study is the implementation in VRML programming. Data from the other three modules—CAD, database, and GIS—are integrated in a number of prototype files (PROTOs) and partially embedded in the main VRML file.

The Data

The data sets, both digital maps and the database, for the input to GIS are often prepared with external tools. Computer Aided Drafting and Design (CAD) is usually the tool to create precise maps containing the spatial objects—points, lines, and polygons (areas). CAD allows the spatial objects to be organized in many ways using layers, colors, line attributes, etc. Database or spreadsheet program organizes the data pertaining to the spatial entities.

There are two types of data sets involved in the geographic information analysis—digital maps and attributes data. Both types of data are best to be created and prepared with external programs. GIS has very limited tools and capabilities to create and to modify either data types. Depending on the purposes of the project, the data need to be defined to serve the purposes of the analysis. The first data type is the digital map. It is necessary that the spatial entities are created in their real scale. This means the dimensions to the map elements must have the correct sizes that can be projected on to the earth surface. However, the level of detail may vary depending on the unit of the spatial analysis. The purposes of the maps need to be defined at the early stage in order to specify the level of detail of the digital map needed.

The digital map of the case study provided by the Facility Planning and Design Department of the University of Michigan. It was prepared with Microstation®, an Intergraph's CAD system. The data was converted into DXF standard vector format. The DXF file, then, was imported into a 3D CAD program, Form•Z®, by Autodessys, Inc., to prepare a uniform-height 3D model. The model was optimized for the VRML application by simplification of the geometries. Curves and complex polylines are straightened. This is to reduce the number of polygons to be rendered in real-time in VRML view.

The uniform-height 3D model was built on a flat ground plane. The reason for this is to restrict the reference of the three dimensional data visualization at the same datum. The actual ground of the University of Michigan North Campus is a non-uniform terrain. However, data visualization of the energy consumption on the terrain surface can easily mislead the users and make it difficult to visually compare the data value because the terrain shifts the bases of the buildings to different elevations. Therefore, the terrain of the ground for the model was eliminated to assist the visual exploratory analysis of the data.

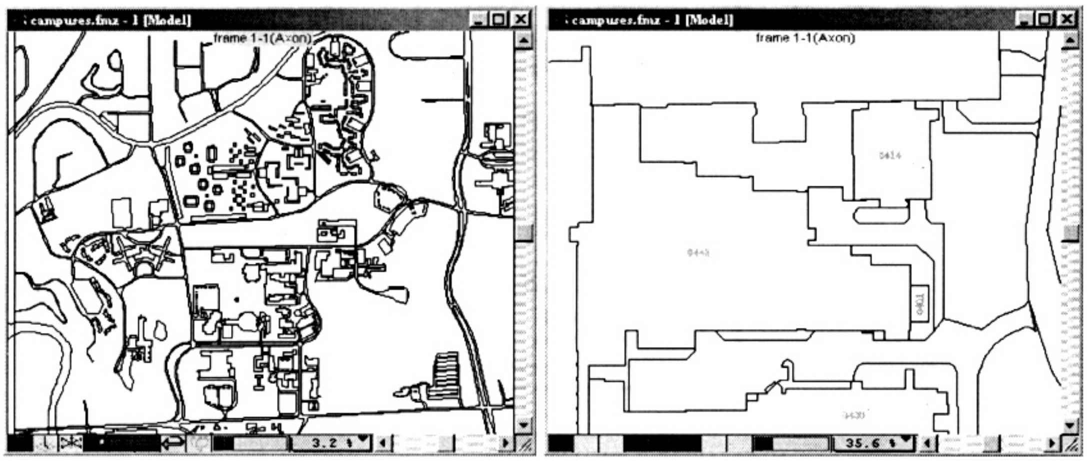


Figure 3. CAD drawing of the University of Michigan North Campus.

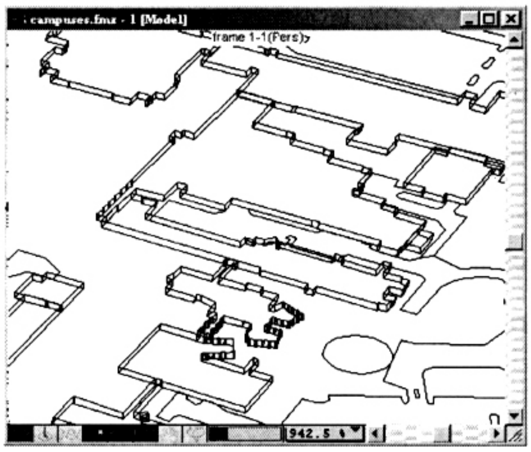


Figure 4. 3D model prepared in Form•Z® with a uniform height.

The 3D uniform-height models of the buildings were named with the Bldgno. The names assigned are important as to be the identifier for the buildings that match the attributes data. When the CAD model was exported into a VRML format, these building names are used in the instancing of the building in VRML programming. The VRML instances can receive events and allow the events to modify they properties such as colors, scale, rotation, and translation.

The second data type is database. There often are multiple data sets for the spatial entities. The attribute data sets can basically be delimited text file. The database in delimited text format is highly transportable. It can be read by most of the spreadsheet, database, and statistical applications. The multiple sets of data need to have proper index in order to be linked in a relational data structure. The GIS imports the data sets and links them with the spatial entities. Each data set can be represented as a layer in the map visualization. GIS combines various layers of information and database in the analysis. Thus, multidimensional analysis can be accomplished.

The database of the case study was obtained as several separate data sets. They are originally tab delimited data files. The files were imported into the GIS to be linked to the spatial objects by the Bldgno. The combined data was sorted and analyzed. Data of the North Campus buildings (Campus Code=500) was queried and filtered out to a new GIS layer. The layer was converted to an ArcView shape file which contain the geometry and attributes table. The table was, then, exported into a delimited text. This is to prepare a data text for VRML programming.

GIS Database

GIS provides crucial tools for spatial analysis. A large number of records in various data sets can be linked in a relational structure. Spatial queries and measurements can be performed. In the study, the digital map of the City of Ann Arbor and the University of Michigan campuses were imported into the GIS. They were georeferenced to be overlaid properly. The campuses were shown as in distinguish colored areas—Central, Medical, North, Athletic, and others. Campuses' buildings were split into separate layers, thus, later, only the North campus will be exported for VRML application.

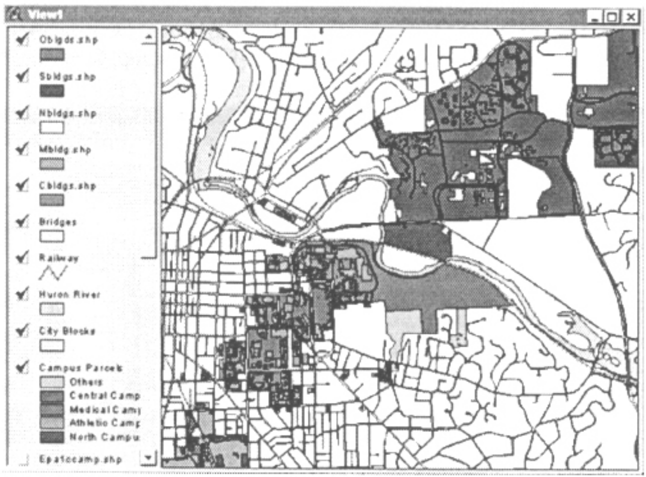
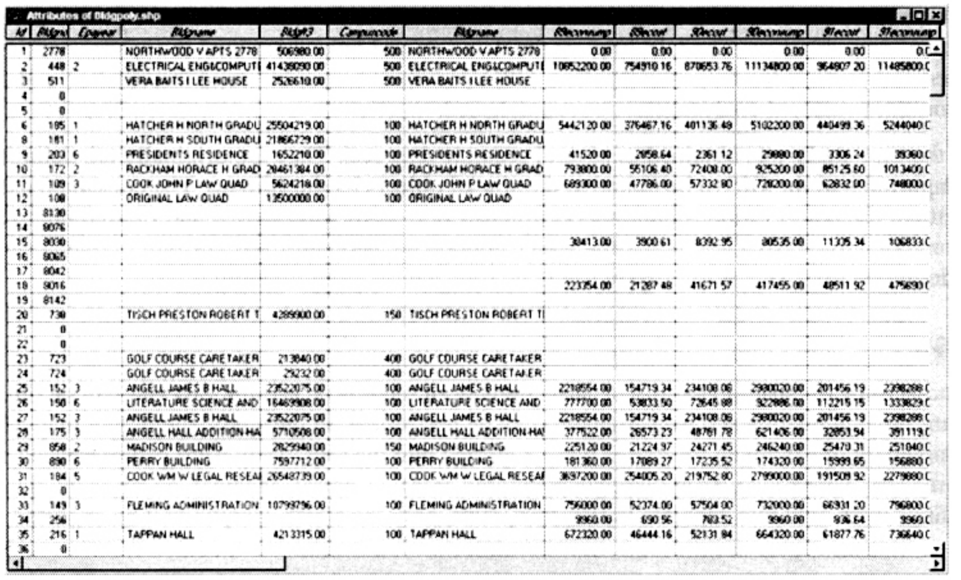


Figure 5. GIS overlays of city map and campus map. Features such as streets, railway, river, bridges are shown with the campus grounds and buildings.

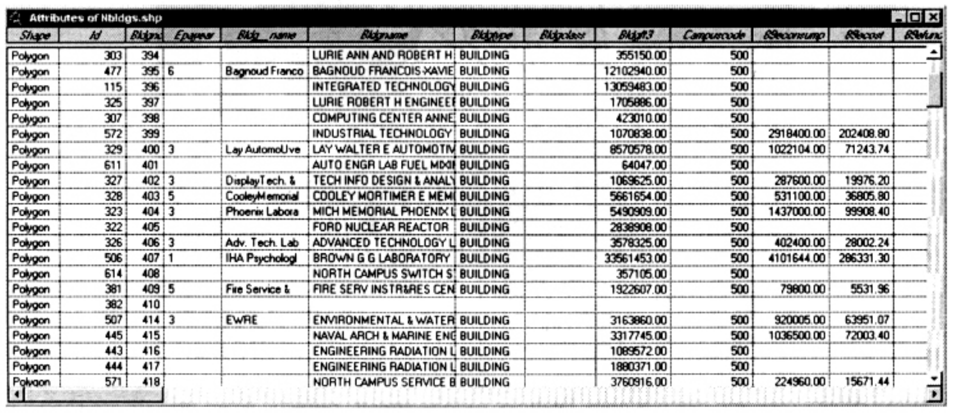
The building digital map contains basic index information, Bldgno. The attribute data sets that were imported into the GIS project file also contain Bldgno. Each attribute value from the data sets can be linked to a proper building. Three attributes of the buildings in North Campus—energy costs (electricity, water, and natural gas from 1991 to 1998)—were selected for this study. A query to filter only buildings with Campuscode = 500 (North Campus) was performed. The result of the query was converted into a separate theme (layer). This new theme contains the combined spatial objects (buildings) and their attributes. GIS can perform further complex queries with mathematical and Boolean operations such as to find buildings that have less electricity costs per square foot and natural gas costs per square foot in 1998 than in 1997. However, for the purpose of the development of VRML application in this study the raw data were needed to create an interactive visualization for data exploration that allows users to see the actual data values in a variety of ways with less artifacts.



Attributes of Bldgpoly.shp

Id	Bldgno	Epgear	Bldgname	Bldgft3	Campuscode	Bldgname	89econsump	89ecost	90ecost	90econsump	91ecost	91econsump
1	2778		NORTHWOOD V APTS 2778	506980.00	500	NORTHWOOD V APTS 2778	0.00	0.00	0.00	0.00	0.00	0.0
2	448	2	ELECTRICAL ENG&COMPUTE	41438090.00	500	ELECTRICAL ENG&COMPUTE	10852200.00	754910.16	870653.76	11134800.00	964807.20	11485800.0
3	511		VERA BAITS I LEE HOUSE	2526610.00	500	VERA BAITS I LEE HOUSE						
4	0											
5	0											
6	185	1	HATCHER H NORTH GRADU	25504219.00	100	HATCHER H NORTH GRADU	5442120.00	376467.16	401136.49	5102200.00	440499.36	5244040.0
8	181	1	HATCHER H SOUTH GRADU	21866729.00	100	HATCHER H SOUTH GRADU						
9	203	6	PRESIDENTS RESIDENCE	1652210.00	100	PRESIDENTS RESIDENCE	41520.00	2858.64	2361.12	29880.00	3306.24	39360.0
10	172	2	RACKHAM HORACE H GRAD	20461384.00	100	RACKHAM HORACE H GRAD	793800.00	55106.40	72408.00	925200.00	85125.60	1013400.0
11	109	3	COOK JOHN P LAW QUAD	5624218.00	100	COOK JOHN P LAW QUAD	689300.00	47786.00	57332.80	728200.00	62832.00	748000.0
12	108		ORIGINAL LAW QUAD	13500000.00	100	ORIGINAL LAW QUAD						
13	8130											
14	9076											
15	8030						38413.00	3900.61	8392.95	80535.00	11305.34	106833.0
16	8065											
17	8042											
18	9016						223354.00	21287.48	41671.57	417455.00	48511.92	475690.0
19	8142											
20	730		TISCH PRESTON ROBERT T	4289900.00	150	TISCH PRESTON ROBERT T						
21	0											
22	0											
23	723		GOLF COURSE CARETAKER	213840.00	400	GOLF COURSE CARETAKER						
24	724		GOLF COURSE CARETAKER	29232.00	400	GOLF COURSE CARETAKER						
25	152	3	ANGELL JAMES B HALL	23522075.00	100	ANGELL JAMES B HALL	2218554.00	154719.34	234108.06	2980020.00	201456.19	2398288.0
26	150	6	LITERATURE SCIENCE AND	16463908.00	100	LITERATURE SCIENCE AND	777700.00	53833.50	72645.88	922886.00	112215.15	1333829.0
27	152	3	ANGELL JAMES B HALL	23522075.00	100	ANGELL JAMES B HALL	2218554.00	154719.34	234108.06	2980020.00	201456.19	2398288.0
28	175	3	ANGELL HALL ADDITION HA	5710508.00	100	ANGELL HALL ADDITION HA	377522.00	26573.23	48761.78	621406.00	32853.94	391119.0
29	858	2	MADISON BUILDING	2829940.00	150	MADISON BUILDING	225120.00	21224.97	24271.45	246240.00	25470.31	251040.0
30	890	6	PERRY BUILDING	7597712.00	100	PERRY BUILDING	181360.00	17089.27	17235.52	174320.00	15999.65	156880.0
31	184	5	COOK WM W LEGAL RESEAI	26548739.00	100	COOK WM W LEGAL RESEAI	3697200.00	254005.20	219752.80	2799000.00	191509.92	2279880.0
32	0											
33	149	3	FLEMING ADMINISTRATION	10799796.00	100	FLEMING ADMINISTRATION	759000.00	52374.00	57504.00	732000.00	66931.20	796800.0
34	256						9960.00	690.56	783.52	9960.00	936.64	9960.0
35	216	1	TAPPAN HALL	4213315.00	100	TAPPAN HALL	672320.00	46444.16	52131.84	664320.00	61877.76	736640.0
36	0											

Figure 6. Linked data for the entire University of Michigan campuses—buildings' identification variables (Bldgno, Bldgname, Bldgft3, Campuscode) and attribute data of the energy consumption (89econsump, 89ecost, …, 98ngconsump, 98ngcost).

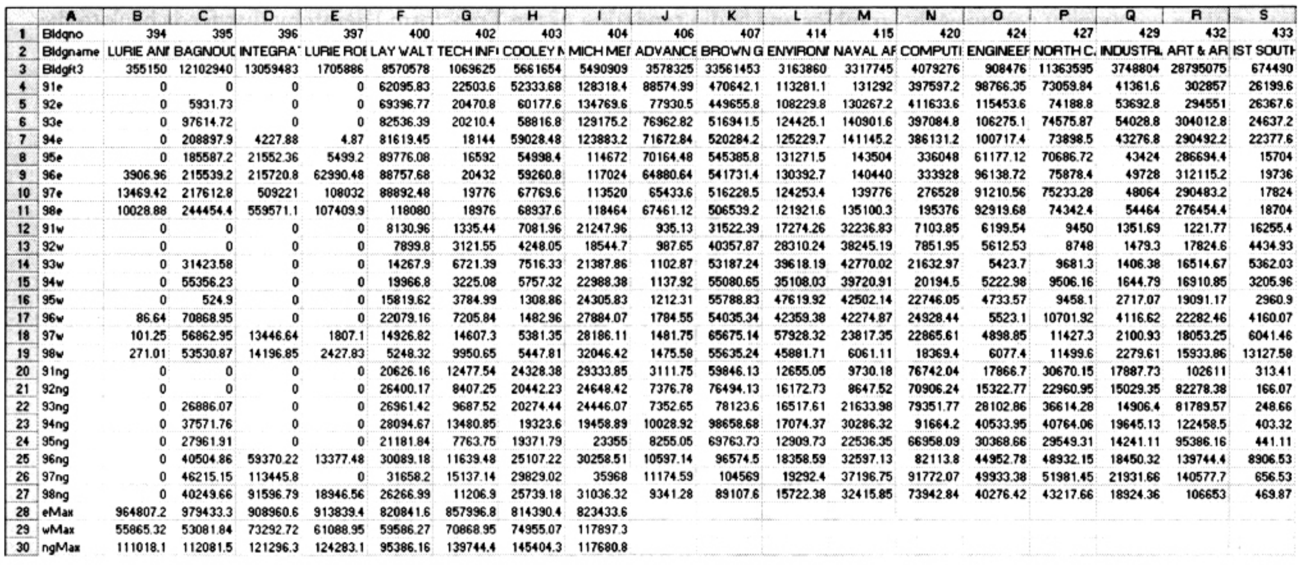


Attributes of Nbldgs.shp

Shape	Id	Bldgno	Epgear	Bldg_name	Bldgname	Bldgtype	Bldgclass	Bldgft3	Campuscode	89econsump	89ecost	89efunc
Polygon	303	394			LURIE ANN AND ROBERT H	BUILDING		355150.00	500			
Polygon	477	395	6	Bagnoud Franco	BAGNOUD FRANCOIS-XAVIE	BUILDING		12102940.00	500			
Polygon	115	396			INTEGRATED TECHNOLOGY	BUILDING		13059483.00	500			
Polygon	325	397			LURIE ROBERT H ENGINEEI	BUILDING		1705886.00	500			
Polygon	307	398			COMPUTING CENTER ANNE	BUILDING		423010.00	500			
Polygon	572	399			INDUSTRIAL TECHNOLOGY	BUILDING		1070838.00	500	2918400.00	202408.80	
Polygon	329	400	3	Lay AutomoUve	LAY WALTER E AUTOMOTIV	BUILDING		8570578.00	500	1022104.00	71243.74	
Polygon	611	401			AUTO ENGR LAB FUEL MIXII	BUILDING		64047.00	500			
Polygon	327	402	3	DisplayTech. &	TECH INFO DESIGN & ANALY	BUILDING		1069625.00	500	287600.00	19976.20	
Polygon	328	403	5	CooleyMemorial	COOLEY MORTIMER E MEMI	BUILDING		5661654.00	500	531100.00	36805.80	
Polygon	323	404	3	Phoenix Labora	MICH MEMORIAL PHOENIX L	BUILDING		5490909.00	500	1437000.00	99908.40	
Polygon	322	405			FORD NUCLEAR REACTOR	BUILDING		2838908.00	500			
Polygon	326	406	3	Adv. Tech. Lab	ADVANCED TECHNOLOGY L	BUILDING		3578325.00	500	402400.00	28002.24	
Polygon	506	407	1	IHA Psychologl	BROWN G G LABORATORY	BUILDING		33561453.00	500	4101644.00	286331.30	
Polygon	614	408			NORTH CAMPUS SWITCH S	BUILDING		357105.00	500			
Polygon	381	409	5	Fire Service &	FIRE SERV INSTR&RES CEN	BUILDING		1922607.00	500	79800.00	5531.96	
Polygon	382	410										
Polygon	507	414	3	EWRE	ENVIRONMENTAL & WATER	BUILDING		3163860.00	500	920005.00	63951.07	
Polygon	445	415			NAVAL ARCH & MARINE ENG	BUILDING		3317745.00	500	1036500.00	72003.40	
Polygon	443	416			ENGINEERING RADIATION L	BUILDING		1089572.00	500			
Polygon	444	417			ENGINEERING RADIATION L	BUILDING		1880371.00	500			
Polygon	571	418			NORTH CAMPUS SERVICE B	BUILDING		3760916.00	500	224960.00	15671.44	

Figure 7. Linked data of North Campus buildings.

The data was, then, exported into a delimited text. The data was transposed in a spreadsheet application to rearrange the fields and records. The data was arranged so that a building and its attributes were in a column format. This was to prepare an attributes to be an array indexed by Bldgno. Maximums and minimums of the energy types were calculated and arranged as an array indexed by the year (1991 to 1998). A comma delimited text was created from the prepared table.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Bldgno	394	395	396	397	400	402	403	404	406	407	414	415	420	424	427	429	432	433
2	Bldgname	LURIE ANI	BAGNOUD	INTEGRA'	LURIE ROI	LAY WALT	TECH INFI	COOLEY N	MICH MEI	ADVANCE	BROWN G	ENVIRONI	NAVAL AF	COMPUTI	ENGINEEF	NORTH C.	INDUSTRL	ART & AR	IST SOUTH
3	Bldgft3	355150	12102940	13059483	1705886	8570578	1069625	5661654	5490909	3578325	33561453	3163860	3317745	4079276	908476	11363595	3748804	28795075	674490
4	91e	0	0	0	0	62095.83	22503.6	52333.68	128318.4	88574.99	470642.1	113281.1	131292	397597.2	98766.35	73059.84	41361.6	302857	26199.6
5	92e	0	5931.73	0	0	69396.77	20470.8	60177.6	134769.6	77930.5	449655.8	108229.8	130267.2	411633.6	115453.6	74188.8	53692.8	294551	26367.6
6	93e	0	97614.72	0	0	82536.39	20210.4	58816.8	129175.2	76962.82	516941.5	124425.1	140901.6	397084.8	106275.1	74575.87	54028.8	304012.8	24637.2
7	94e	0	208897.9	4227.88	4.87	81619.45	18144	59028.48	123883.2	71672.84	520284.2	125229.7	141145.2	386131.2	100717.4	73898.5	43276.8	290492.2	22377.6
8	95e	0	185587.2	21552.36	5499.2	89776.08	16592	54998.4	114672	70164.48	545385.8	131271.5	143504	336048	61177.12	70686.72	43424	286694.4	15704
9	96e	3906.96	215539.2	215720.8	62990.48	88757.68	20432	59260.8	117024	64880.64	541731.4	130392.7	140440	333928	96138.72	75878.4	49728	312115.2	19736
10	97e	13469.42	217612.8	509221	108032	88892.48	19776	67769.6	113520	65433.6	516228.5	124253.4	139776	276528	91210.56	75233.28	48064	290483.2	17824
11	98e	10028.88	244454.4	559571.1	107409.9	118080	18976	68937.6	118464	67461.12	506539.2	121921.6	135100.3	195376	92919.68	74342.4	54464	276454.4	18704
12	91w	0	0	0	0	8130.96	1335.44	7081.96	21247.96	935.13	31522.39	17274.26	32236.83	7103.85	6199.54	9450	1351.69	1221.77	16255.4
13	92w	0	0	0	0	7899.8	3121.55	4248.05	18544.7	987.65	40357.87	28310.24	38245.19	7851.95	5612.53	8748	1479.3	17824.6	4434.93
14	93w	0	31423.58	0	0	14267.9	6721.39	7516.33	21387.86	1102.87	53187.24	39618.19	42770.02	21632.97	5423.7	9681.3	1406.38	16514.67	5362.03
15	94w	0	55356.23	0	0	19966.8	3225.08	5757.32	22988.38	1137.92	55080.65	35108.03	39720.91	20194.5	5222.98	9506.16	1644.79	16910.85	3205.96
16	95w	0	524.9	0	0	15819.62	3784.99	1308.86	24305.83	1212.31	55788.83	47619.92	42502.14	22746.05	4733.57	9458.1	2717.07	19091.17	2960.9
17	96w	86.64	70868.95	0	0	22079.16	7205.84	1482.96	27884.07	1784.55	54035.34	42359.38	42274.87	24928.44	5523.1	10701.92	4116.62	22282.46	4160.07
18	97w	101.25	56862.95	13446.64	1807.1	14926.82	14607.3	5381.35	28186.11	1481.75	65675.14	57928.32	23817.35	22865.61	4898.85	11427.3	2100.93	18053.25	6041.46
19	98w	271.01	53530.87	14196.85	2427.83	5248.32	9950.65	5447.81	32046.42	1475.58	55635.24	45881.71	6061.11	18369.4	6077.4	11499.6	2279.61	15933.86	13127.58
20	91ng	0	0	0	0	20626.16	12477.54	24328.38	29333.85	3111.75	59846.13	12655.05	9730.18	76742.04	17866.7	30670.15	17887.73	102611	313.41
21	92ng	0	0	0	0	26400.17	8407.25	20442.23	24648.42	7376.78	76494.13	16172.73	8647.52	70906.24	15322.77	22960.95	15029.35	82278.38	166.07
22	93ng	0	26886.07	0	0	26961.42	9687.52	20274.44	24446.07	7352.65	78123.6	16517.61	21633.98	79351.77	28102.86	36614.28	14906.4	81789.57	248.66
23	94ng	0	37571.76	0	0	28094.67	13480.85	19323.6	19458.89	10028.92	98658.68	17074.37	30286.32	91664.2	40533.95	40764.06	19645.13	122458.5	403.32
24	95ng	0	27961.91	0	0	21181.84	7763.75	19371.79	23355	8255.05	69763.73	12909.73	22536.35	66958.09	30368.66	29549.31	14241.11	95386.16	441.11
25	96ng	0	40504.86	59370.22	13377.48	30089.18	11639.48	25107.22	30258.51	10597.14	96574.5	18358.59	32597.13	82113.8	44952.78	48932.15	18450.32	139744.4	8906.53
26	97ng	0	46215.15	113445.8	0	31658.2	15137.14	29829.02	35968	11174.59	104569	19292.4	37196.75	91772.07	49933.38	51981.45	21931.66	140577.7	656.53
27	98ng	0	40249.66	91596.79	18946.56	26266.99	11206.9	25739.18	31036.32	9341.28	89107.6	15722.38	32415.85	73942.84	40276.42	43217.66	18924.36	106653	469.87
28	eMax	964807.2	979433.3	908960.6	913839.4	820841.6	857996.8	814390.4	823433.6										
29	wMax	55865.32	53081.84	73292.72	61088.95	59586.27	70868.95	74955.07	117897.3										
30	ngMax	111018.1	112081.5	121296.3	124283.1	95386.16	139744.4	145404.3	117680.8										

Figure 8. Attribute data of the energy consumption of the North Campus buildings.

The 3D uniform-height model and the comma delimited data were the two essential data for the development of the VRML visualization. Further interactive components were to be built within the programming of the VRML model.

VRML Standard

VRML, an acronym for the Virtual Reality Modeling Language, was conceived in the spring of 1994 at the first World Wide Web conference, held in Geneva. As a three-dimensional graphical visualization tool, VRML was intended to become the standard language for interactive simulations within the World Wide Web and was rapidly adopted by the wider Internet community. A subset of the Open Inventor ASCII file format was used to form the basis of the language, and the development of VRML was speeded up considerably by the contribution of a VRML file parser into the public domain by the company Silicon Graphics. This and further development took place over the Internet via an Internet mailing list and later through a number of news-groups. Web3d consortium was founded in 1999 as the successor of the original VRML Consortium, founded in 1997.

The Virtual Reality Modeling Language (VRML) is a "file format for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia" (www.web3d.org). A VRML document takes the form of a human-readable text file describing a three-dimensional scene. It is capable of representing static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. This text file comprises in effect a list of programming syntax which tell the computer to place objects, of given sizes and colors, at specific locations within a virtual world. Simple 2D and 3D objects can be constructed out of what are referred to as primitives—simple, pre-defined primitive geometric shapes, for example cubes and spheres. More complex objects, for example curved surfaces of landscape and non-rectilinear buildings, that cannot be adequately modeled using simple primitives can be referred to as a face or polygon. Most landscapes and free-form objects, given their near infinite complexity, require many thousands of individual faces, or polygons. This results in much larger text files, thus, slower real-time navigation. Other parts of the virtual environment, such as lights, backgrounds, navigation speed, can also be defined within the same file. A VRML file can be distributed over the Internet and parsed by a browser program which sensually renders the document into an interactive form. All browsing is done on the client machine resulting in low bandwidth requirements and hardware independent distribution. VRML browser is a presentation application that accepts user input through real-time responsive user interface that allows users to manipulation of objects and navigation using an input device. The three main components of the browser are: Parser, Scene Graph, and Audio/Visual Presentation (see Figure 2). The Parser component reads the VRML file and creates the Scene Graph. The Scene Graph component consists of the Transformation Hierarchy of nodes and the Route Graph. The Scene Graph also includes the Execution Engine that processes events, reads and edits the Route Graph, and makes changes to the Transform Hierarchy of nodes. User input generally affects sensors and navigation, and thus is wired to the Route Graph component, defined by sensors, and the Audio/Visual Presentation component. The Audio/Visual

Presentation component performs the graphics and audio rendering of the Transform Hierarchy that feeds back to the user (www.web3d.org).

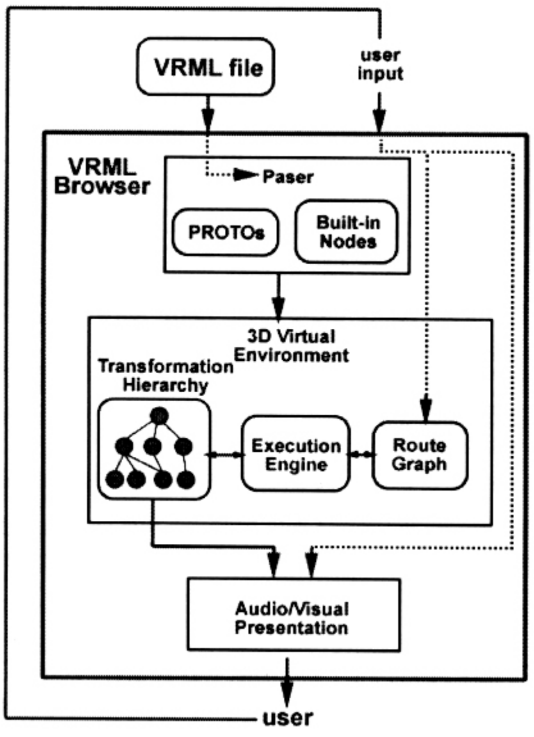


Figure 9. Conceptual of VRML browser (source: adapted from www.web3d.org).

Scene Graph Structure

A typical VRML file consists of the header, the scene graph, and event routing. The contents of this file are processed for presentation and interaction by a program known as a browser. The scene graph contains nodes which describe objects and their properties, as well as the environment such as lights, predefined viewpoints. It contains hierarchically grouped geometry to provide an audio-visual representation of objects, as well as nodes that participate in the event generation and routing mechanism.

The header is the identifier for a VRML browser to recognize the file. The standard VRML 97 header is

```
#VRML V2.0 utf8
```

"utf8" is the encoding type. The header can have optional comments following the encoding type. It has to end with line terminator, either a linefeed or a carriage-return character.

Following the header are a number of nodes that make up a scene graph. The order of nodes within the scene graph is critical, as changes in position or orientation all subsequent nodes. Nodes represent the building blocks of VRML and describe shapes, lights, cameras, position and orientation (Gillings and Goodrick, 1996). Standard unit in VRML is meter. The coordinate system is a screen base which has x as the horizontal axis, y as the vertical axis, and z as the perpendicular axis to the screen.

A tree-like structure can be created using separator nodes which allows parts of the scene graph to be functionally independent and isolated from subsequent nodes. Specific states, for example color, will then be saved before entering the separator to make specific state changes, and restored upon leaving. All of the objects need not necessarily be defined within the one file. Instances of objects on the same server or elsewhere on the Internet may be included within the file. Instancing using DEF statement is used to define objects and USE to subsequently re-use the object, thus allowing some degree of efficient, modular programming. The use of instancing, DEF and USE, optimizes the memory used. When the parser read the nodes from the VRML text file, it puts each node in a memory location and associates it with that memory location whenever it uses that node during rendering. So, whenever the USE is encountered, the computer does not put another copy in memory, but instead keeps a pointer to the original. Therefore, it save time in typing codes, network download time, browser loading time, computer memory usage, and rendering time (Marrin and Campbell, 1997).

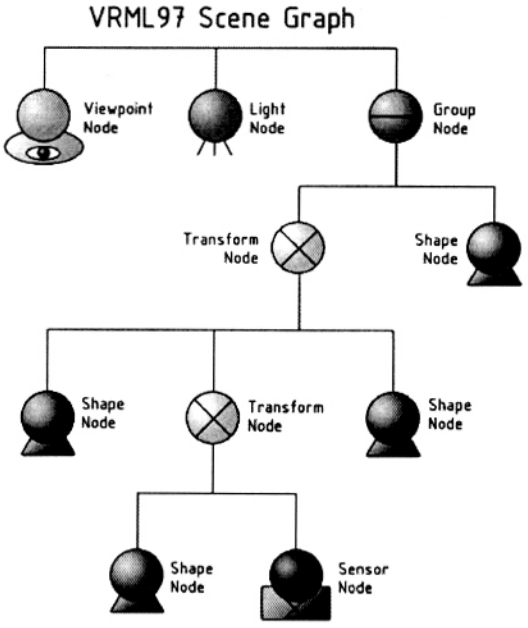


Figure 10. Tree structure of scene graph.

Many CAD applications are capable of translating the layers or grouping of the models into proper hierarchical structure and instances in the VRML scene graph. The simple structure of VRML makes the production of macro-language routines for the export of models from CAD formats to VRML relatively straightforward. Routines are already available for packages such as Autodesk's 3-D Studio, form•Z®. CAD programs also provide the ability to fine-tune scenes; surfaces may be smoothed, component polygon counts reduced for greater speed of rendering, and cameras can be added to provide predefined viewpoints. In addition, the GIS, such as Arc/Info and ArcView, have conversion utilities that can convert GIS layers or themes to VRML. There are also various stand-alone tools for the conversion of the popular AutoCad data exchange format (DXF) into VRML, making the conversion of existing 3-dimensional CAD drawings relatively straightforward (Gillings and Goodrick, 1996). In this study, the 3D uniform-height models created in form•Z® were named and organized hierarchically with layers in the same fashion required in VRML scene graph.

Although the VRML standard has been released over 3 years, different VRML browsers and CAD modeling programs may use quite different interpretations of the VRML standards to implement very different features. Despite the presence of complex modeling programs to assist in the production of VRML documents, it is often necessary to manually edit the exported VRML code to be able to tweak files to satisfy different browsers and to use capabilities provided in the VRML specification, such as animation with interpolators and behavioral control with sensors.

There are as many as 54 VRML nodes plus instancing, routing and prototype/external prototype statements. The VRML nodes can be classified into 5 major types—geometry and appearances, scene environment components, groupings, behaviors, and miscellaneous nodes. Geometry and appearances define objects in the scene and their appearances. Digital image and movie can be mapped onto the object's surfaces (polygons). Scene environment components comprise lights, viewpoints, navigation, background, fog, and sounds. Light nodes define 3 different lighting models to light up the objects' surfaces. Grouping nodes are used to arrange the hierarchical structure of the scene graph. Behavior nodes include sensors, interpolators, and script nodes. Miscellaneous nodes are special nodes which may not visually affect the scene graph. For example, the WorldInfo node contains strings of title and text information about the file, PROTO defines customized dynamic node which is not displayed until the node is called in the scene graph.

Table 1. Node types

Geometry & Appearances	Scene Environment Components	Grouping	Behaviors	Miscellaneous Nodes and Statements
Appearance	AudioClip	Anchor	ColorInterpolator	DEF
Box	Background	Billboard	CoordinateInterpolator	EXTERNPROTO
Color	DirectionalLight	Collision	CylinderSensor	PROTO
Cone	Fog	Group	NormalIntrepolator	ROUTE
Coordinate	NavigationInfo	Inline	OrientationInterpolator	USE
Cylinder	PointLight	LOD	PlaneSensor	WorldInfo
ElevationGrid	Sound	Switch	PositionInterpolator	
Extrusion	SpotLight	Transform	ProximitySensor	
FontStyle	Viewpoint		ScalarInterpolator	
IndexedFaceSet			Script	
IndexedLineSet			SphereSensor	
ImageTexture			TimeSensor	
Material			TouchSensor	
MovieTexture			VisibilitySensor	
Normal				
PixelTexture				
PointSet				
Shape				
Sphere				
Text				
TextureCoordinate				
TextureTransform				

A node contains a list of fields which hold values that define parameters for its properties and functions. A node specification is defined with the following syntax:

```
NodeName {
        class          type          name          [initial value]
}
```

i.e.

```
Viewpoint {
        eventIn        SFBool        set_bind
        exposedField   SFFloat       fieldOfView   0.785398
        exposedField   SFBool        jump          TRUE
        exposedField   SFRotation    orientation   0 0 1 0
        exposedField   SFVec3f       position      0 0 10
        field          SFString      description   ""
        eventOut       SFTime        bindTime
        eventOut       SFBool        isBound
}
```

The texts in bold, node name and field names, are the part to be typed in the VRML file. If a field is not specified, the initial value will be used. The events, *eventIn* and *eventOut*, are the functions of the node that allow values to be pass from and to other nodes via ROUTE statement. *eventIn* defines a property of the node that can receive a new value passed from another node. *eventOut* send a value out from the node. The *eventOut* functions as soon as the node is loaded by the browser. These properties and functions of the VRML nodes allow great flexibility for interactivity.

Interactivity in VRML scenegraph

Interaction allows user to control behaviors of the objects while exploring the scene. User can touch an object to start moving the object or a group of objects, to change light intensity, or to change the object's appearance. Interactivity in VRML (version 2 or 97) is executed by passing a series of properties around to the various objects in the scene graph. When one node wants to pass some information to another node, it creates an *event*. An *event* is "something between a function call and an assignment in a programming language. An event contains two pieces of information—the data and a timestamp. A timestamp is when the event was originally generated. In order to maintain a coherent scene, the order of execution is important. The sequences are facilitated by the timestamps of events. The timestamp is the browser's internal representation of when the event occurred so that it can maintain the correct sequence (Roehl, et al., 1997).

VRML does not contain a function call mechanism for passing information; instead, an explicit connection between two fields of the nodes is created. Table 2 shows accessibility of the node's properties (*class specifiers*). A *class specifier* contains a value. This value has a fixed field type (*type specifier*), i.e. SFBool, SFVec3f, SFRotation, etc. The value passed from and to node's property is required to have a matching field type. Either a ROUTE statement or a direct access can execute the value passed from a node to another node with a function in a Script node.

Table 2. VRML class specifiers and access types

Class Specifier	Access Available to other nodes
field	No external access
eventIn	Write only
eventOut	Read only
exposedField	Read and write

Source: adapted from Roehl et al., 1997.

Most, but not all, *eventIns* correspond to an *exposedField* in the node. Fore example, the `Transform` node has `set_translation` *eventIn* corresponding to `translation` field which is an SFVec3f type. An *exposedField* class can be set to a particular value or changed when its node receives a corresponding *eventIn*. But, a *field* class is not exposed. It can be set to a particular value in the file format, but cannot be changed on-the-fly by an event. Most nodes that have an *eventIn* corresponding to a field also have an *eventOut* corresponding to that same field. For example, the `Transform` node, in addition to a `set_translation` *eventIn*, also has a `translation_changed` *eventOut*. This event is sent whenever the `set_translation` *eventIn* is received. This allows the chaining of events through many nodes.

Figure 11 illustrates an example of event routings from a click of the mouse on *TouchSensor* node (typically a sensor is nested with a geometry as children of a grouping node) to both `startTime` of a *TimeSensor* and a Script. The *eventOut* from the *TouchSensor* activates the *TimeSensor* and toggle a *SFBool* field, `toggle_changed` to `TRUE` or `FALSE`. The *SFBool* value of the `toggle_changed` is then sent to toggle `enabled` field of the *TimeSensor* to activate or deactivate it. This *TimeSensor*, then, can be used to interact with an interpolator to animate an object or a group of objects.

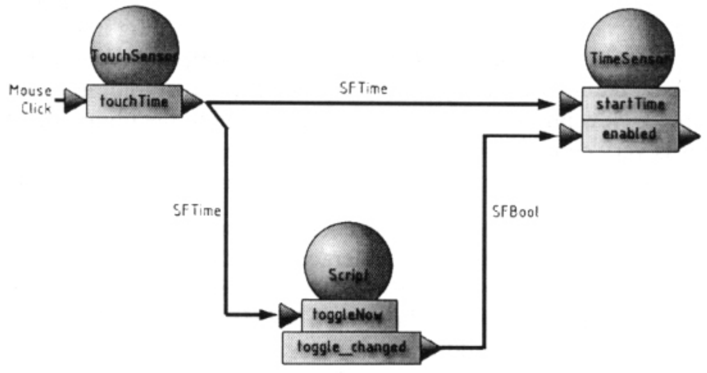


Figure 11. Example of event routing.

In addition to passing value from an *eventOut* of a node to and *eventIn* of another node to change the value of corresponding *exposedField*, the *exposedField* can be instantaneously changed by the direct access from a function in a `Script` node. Script node allows arbitrary, author-defined fields and events, and the event processing. VRML supports several programming languages for writing scripts, including the popular Java language, javascript, and VRMLscript. The script is a powerful and flexible way to create interactivity and behaviors for VRML objects. An event received by a Script node causes the execution of a script function which has the ability to send events through the normal event-routing mechanism, or bypass this mechanism and send events directly to any node to which the Script node has a reference. Scripts can also dynamically add or delete routes and thereby change the event-routing topology (Carey and Bell, 1997).

Prototype: PROTO

"The PROTO statement defines a new node type in terms of already defined (built-in or prototyped) node types. Once defined, prototyped node types may be instantiated in the scene graph exactly like the built-in node types" (www.web3d.org). PROTO statement allows customized node to be defined. This customized node is a prototype. A prototype is a combination of standard VRML nodes. Thus, a prototype can be a complex set of geometries, appearances with textures, and behaviors. Once declared, a prototype can be used like any of the standard VRML nodes in the scene graph. Class and type specifiers can be assigned to a prototype with arbitrary names. Prototype need not be in the same VRML file. It can be referred externally using EXTERNPROTO statement.

Prototype is useful for creating parts libraries. Libraries become sets of standard parts that can be reused through many different scenes over and over again. Moreover, the properties of the parts can be modifiable when the class specifiers of the prototype are exposedFields. In this study, prototypes were used extensively for parts of the interactive menu items and the library of building geometries database. This allows the libraries to be updated externally with convenience. In addition, the properties of the parts, such as appearances and position, used in the final VRML scene graph can be articulated differently each time they are used.

Case Study: VRML Components

There are actually many ways to implement the VRML visualization of such data. One technique is to create several versions of the 3D models that represent the values of the energy cost of all the years. With many predefined heights, VRML scene graph can call in a proper version of the 3D model to be visible when a menu is selected. This technique can be accomplished with Anchor node that links to external models or with Switch node with external model files are linked (inlined) to the scene graph. However, the models are static with predefined properties. This makes the updating of the building geometries or the energy data become a tedious task because all models have to be update properly. Moreover, the interactivity is rather limited. For example, it would be difficult to implement the sequential animation of the height values through all the years.

Another possibility of implementing VRML for this type of visualization is to build prototype to contain all building's energy variables. This prototype has an exposed children field to accept the geometry of a building specified at the time programmed in the scene graph. With this prototype, author or programmer needs to input the data—building geometry and energy data—at the time the scene graph file is created. This means a building in the scene graph has all the data attributes built-in the node. Any attribute can be accessed and passed to script's functions for interactivities. However, the building geometries need to be maintained in separate files, each file for a building. Then, a building file is inlined into the prototype's children field. New building geometries can be added to the library of files; and the existing file can be modified individually. This is very convenient to maintain the building geometries data. The problem for this technique is that it is a tedious task to manually input in a large number of energy data values into each building code.

The method implemented in this study, the visualization the energy consumption of the University of Michigan North Campus buildings with VRML, was to create stand-alone interactive VRML with external prototypes and external static models of the context. The structure of this visualization was modeled with three separate sets of VRML files—prototypes, models of the contextual environments, and the main VRML scene graph. They were structured to provide flexibility in updating the databases, either the building geometries or the energy data. The components of this VRML implementation can be illustrated in Figure 12 below. Figure 12 shows the linkages of external prototypes and external VRML models to the main scene graph. In addition, The main VRML scene graph shows its subcomponents and their events routing. The energy consumption database was embedded inside a Script in form of arrays. The data values in each array were indexed by the building numbers. The comma delimited text output of the energy consumption data from GIS was simply cut and pasted into the arrays in the Script. Therefore, these arrays of data in the Script can be easily updated and modified.

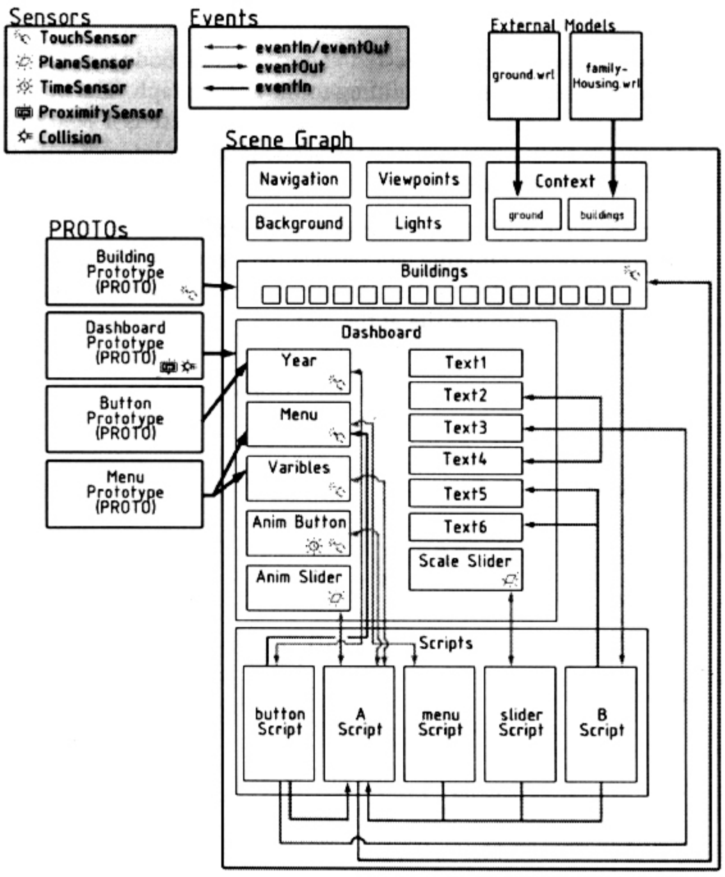


Figure 12. VRML components in the visualization of energy consumption case study

Prototypes

There are four prototypes created for the interactive visualization—buildings, dashboard, button, and menu prototypes. They provide a set of library database of both geometries and functionalities. The buildings prototype is a database of all buildings' geometries which can be called to the scene graph one at a time. The building called from this prototype has a built-in TouchSensor to send and receive events with other components in the scene graph. The dashboard prototype is mainly a set of functions that update the positions and orientations of its children to be visible at a relative position to the viewer at all time. This prototype accepts any VRML nodes as children when specified in the main scene graph file. The button and menu prototypes store geometries and appearances of the menu buttons that make the menu buttons responds to the on/off stages. The button and menu prototypes were used as children of the dashboard in the scene graph.

Buildings:

This prototype has all buildings' geometrical database nested under a Switch node that has `whichChoice` field exposed. This allows the prototype to be use and call a specific building to the scene graph. The buildings' scale and material are exposed to allow access of new value to modify its properties. In addition, there is a *TouchSensor* built-in, with `enabled` and `touchTime` fields exposed, to allow a building called to the scene graph to interact with the user and perform functions specified in the main VRML scene graph.

The prototype can be easily modified. Each building in an instance, named with the building number. They were sorted by the building number. New geometry of the building can replace the old one by simply cut and paste the new syntax over the old one. New buildings can also be added to the database. The new buildings can be inserted to maintain the ordering by building numbers or be appended to the file at the end.

```
PROTO Bldg [
     exposedField SFInt32 BldgChoice -1
     exposedField SFNode Mat Material {diffuseColor 0.8 0.7 0.6 specularColor 0.8 0.8 0.8}
     exposedField SFVec3f Scale 1 1 1
     exposedField SFBool Enabled FALSE
     eventOut SFTime Touch
]
{
DEF bldgScale Transform { scale IS Scale
     children [
     DEF Sensor TouchSensor {
          touchTime IS Touch
          enabled IS Enabled
     }
DEF BldgSwitch Switch {
     whichChoice IS BldgChoice
     choice [
          DEF b394 Transform {
               children [
                    Shape {
                         geometry IndexedFaceSet {
                              coord Coordinate {
                                   point [
                                        -3.11667 1.1949e-05 0.733333
                                        -3.11667 0 -2.84217e-16
                                        .........
                                        .........
                                        -4.5 1 1.09375
                                        -4.5 1 4.28642e-12
                                        -3.11667 1 4.28614e-12
                                   ]
                              }
                              convex FALSE
                              coordIndex [
                                   0 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 -1
                                   26 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 -1
                                   0 26 27 25 -1
                                   .........
                                   .........
                                   0 1 51 26 -1
                                   50 51 1 2 -1
                              ]
                         }
                         appearance DEF bApp1 Appearance {
                              material IS Mat
                         }
                    }
               ]
          }

.........

.........
```

Figure 13. Buildings prototype.

Dashboard:

The dashboard prototype defines parameters and sensors for the children nodes to be properly displayed in front of the viewer as it moves through the 3D environment. This prototype has an open children field that allows geometries and other VRML nodes to be added for flexible functionalities of the dashboard interface. The *ProximitySensor* and *Collision* nodes are the main properties of this prototype. The *ProximitySensor* tracks the current position of the viewer and send translation (position) and orientation (rotation) of the viewer to update the translation and orientation of the dashboard's children nodes so that all children maintain their relative position to the view at all time.

```
PROTO Dash1 [
     exposedField MFNode buttonNode []
]
{
     DEF Dashboard Group {
          children [
               DEF PrxS1 ProximitySensor { size 100000 100000 100000}
               DEF CN Collision {
                    collide FALSE
                    children [
                         DEF Board Transform {
                              children [
                                   DEF Button Transform {
                                        translation -0.12 -0.11 0.06
                         children IS buttonNode
                                   }
                                   DEF DL2 DirectionalLight {
                                        direction -0.5 0.3 -1
                                   }
                              ]
                         } #theBoard
                    ]
               ROUTE PrxS1.position TO Board.translation
               ROUTE PrxS1.orientation TO Board.rotation
               } #Collision
          ]
     } #Group
} #PROTO
```

Figure 14. Dashboard prototype.

Button & Menu:

The button prototype is a library for a square button which can have two texture maps. The purpose of the prototype of a button with two texture maps is to allow the appearance of the button to change when clicked to be on or off. The URLs of the two texture maps are exposed, *buttonInactiveTexture* and *buttonActiveTexture*, so that the image texture file can be specified when used in the VRML scene graph. The menu prototype has the exactly the same properties as the button prototype, except the geometry is a rectangle made for the variables menus.

The prototype was built with a Switch node that contains two choices. The Switch node has whichChoice exposed, defined with a customized name *buttonSwitch*. Each choice has an exposed texture's URL that can be specified when used in the scene graph. One choice is for the "off" stage with one texture map; the other is for "on" stage, when clicked, with another texture map. The prototype needs a *TouchSensor* and a Script to register the "on/off" toggle stages in the main scene graph.

```
PROTO dashButton [
     exposedField SFString buttonName "select menu"
     exposedField MFString buttonActiveTexture []
     exposedField MFString buttonInactiveTexture []
     exposedField SFInt32 buttonSwitch 0
]
{
     Transform {
               children [
                    DEF switchButton Switch {
                            whichChoice IS buttonSwitch
                         choice [
                              DEF inactiveButton Transform {
                                   children [
                                     Shape {
                                          geometry DEF face IndexedFaceSet {
                                               coord ....
                                               .......
                                          }
                                          appearance DEF y91 Appearance {
                                               material DEF buttonColor Material {}
                                               texture DEF ImText ImageTexture { url IS buttonInactiveTexture }
                                          }
                                     }
                                ]
                              }
                              DEF inactiveButton Transform {
                                   children [
                                     Shape {
                                          geometry USE face
                                          appearance Appearance {
                                              texture DEF ImText2 ImageTexture { url IS buttonActiveTexture}
                                              material USE buttonColor
                                          }
                                     .......
                                     .......
     }
}
```

Figure 15. Button prototype.

External Models of the Context

There are two files of static models of the context for the North Campus academic buildings—family housing complex and the ground, with streets and parking lots layout. The two models were products of the 3D CAD models. The family housing complex was exported separately from the overall campus model. The streets and parking lots were, however, converted into an image of the plan (top) view. This is because of the geometries of these elements have far too many polygons. Their large number of polygons impede the real-time interactive performance of the final VRML model. Moreover, they are 2-dimensional objects that need not be interactive in the VRML application. They provide visual references for the viewer. Thus, in order to optimize the performance, the VRML model for the ground with streets and parking lots was done by creating a large rectangle and texture mapped with the image of the plan view of the campus. The two models were inlined into the final VRML scene graph.

VRML Scene Graph: Master File

Environment Enhancement

As shown in the Figure 12, the prototypes and external VRML models of the context were brought into the scene graph. In addition, the scene environment components were added. NavigationInfo defines the navigation speed, types of navigation. Light sources enhance the visual quality of the model at any view angle. Background adds color gradients for the sky and ground, and defines the horizon. Scripts were the major component of the complex interactivity in the model. Viewpoints add predefined views of the model for easy navigation.

Models

Buildings were added to the scene graph using the library in the buildings prototype. The prototype made this part of the programming truly simple. Figure 16 is the list of syntax to add all the buildings into the scene graph. Each building was instanced with building number. Each building is equipped with a TouchSensor (built-in in the prototype). The sensor receives an event from the viewer by the mouse click on the building and passes the value to a Script, Bscript. The Script, then, assigns an index value and sends it to the database Script, Ascript. The Ascript retrieve data values from the data arrays and construct a couple of new strings to display detail information about the selected building—Bldgno, electricity, water, and natural gas costs for the selected year, and building name.

```
DEF bldgs Transform {    #loading bldgs from bldgsPROTO to the scenegraph
     children [
          DEF b394 Bldg {BldgChoice 0}
          DEF b395 Bldg {BldgChoice 1}
          DEF b396 Bldg {BldgChoice 2}
          DEF b427 Bldg {BldgChoice 14}
          DEF b432 Bldg {BldgChoice 16}
          DEF b440 Bldg {BldgChoice 20}
          DEF g441 Bldg {BldgChoice 21}
          DEF b442 Bldg {BldgChoice 22}
          DEF b443 Bldg {BldgChoice 23}
          DEF b445 Bldg {BldgChoice 24}
          DEF b555 Bldg {BldgChoice 30}
          DEF g510 Bldg {BldgChoice 28}
          DEF b420 Bldg {BldgChoice 12}
          DEF b407 Bldg {BldgChoice 9}
          DEF b414 Bldg {BldgChoice 10}
          DEF b447 Bldg {BldgChoice 26}
          DEF g424 Bldg {BldgChoice 13}
          DEF b446 Bldg {BldgChoice 25}
          DEF b439 Bldg {BldgChoice 19}
          DEF b403 Bldg {BldgChoice 6}
          DEF b429 Bldg {BldgChoice 15}
          DEF b397 Bldg {BldgChoice 3}
          DEF g435 Bldg {BldgChoice 18}
          DEF b400 Bldg {BldgChoice 4}
          DEF b406 Bldg {BldgChoice 8}
          DEF b402 Bldg {BldgChoice 5}
          DEF g415 Bldg {BldgChoice 11}
          DEF g515 Bldg {BldgChoice 29}
          DEF g404 Bldg {BldgChoice 7}
          DEF g448 Bldg {BldgChoice 27}
          DEF g433 Bldg {BldgChoice 17}
     ]
}
```

Figure 16. VRML code to add all buildings into the scene graph.

Dashboard

All the menu items were children of the dashboard prototype. There are three major groups of elements on the dashboard. The first group is the set of menu buttons. Initially, only Year menu buttons are visible. The second group is the instructional texts. The texts are dynamically updated according to the menu buttons selected. The third group is the scale slider to control the height scale of the interactive models. The behaviors of these groups are described below.

Menus

Menu	Graphical Representation	Functions
Year menu	'90 '91 '92 '93 '94 '95 '96 '97 '98 (inactive) '90 '91 '92 '93 '94 '95 '96 '97 '98 (active) Reset button (inactive) Reset button (active)	• *TouchSensor* and *buttonScript* toggle inactive/active states of the appearances. The *buttonScript* assigns value to the selected year to the database Script, *Ascript*, to identify proper index of the database arrays. • Enables the animation marker and moves it over the selected Year button. • Resets the buildings' heights and Height Scale slider. • Toggles Height by and Color by menus. • Changes line 1 of the instructional text to show the value of the selected year. • Enables buildings' *TouchSensor*. *Reset button* • Resets buildings' heights, colors, animation marker's position, instructional texts, Height Scale slider to initial state. • Disables animation marker and buildings' *TouchSensor*.
Height by & Color by menu	Height by: Color by: (inactive) Height by: Color by: (active)	• Toggle inactive/active states of their appearances. • Toggle variable menus and animation Play/Pause button (but not enable it). • Reset buildings' heights, colors, Height Scale marker and value to 100% position.
Variables menu (Electricity, Water, and Natural Gas) under *Height by* and *Color by* menus	(inactive) (active) Electricity Electricity Water Water Natural Gas Natural Gas	• Under *Height by* menu, enable animation Play/Pause button. • Change Max. value text on the Height Scale's scale bar. • Assign the indexing value for either height or color to the *Ascript*. • Under *Color by* menu, set buildings' colors to represent the data value in red color theme—brighter reds for higher energy costs. • If there is a building selected (with blue color added) to show data values in the Text5 and Text6, click on the selected color variable again will reset the selected building's color to the red theme. • Change either line 2 or line 3 of the instructional text to show the selected variable and Min. and Max. values of the selected variable of the selected year.
Animation Controls	(Play) (Pause) '91 '92 '93 '94 '95 '96 '97 '98 (Animation Marker)	*Play/Pause* • Starts TimeSensor to animate the buildings' heights according to the selected Height by variable. • Play button disables Year, Height by, Color by, and Reset menu buttons and Animation Marker. Pause button enables them back. *Animation Marker* • Dynamically sets buildings' heights by assigning selected year indexes as it is moving over the year buttons.

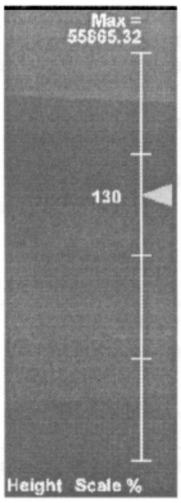
Instructional Texts

These texts are located next to the menu buttons, at the top around the center of the screen. Three lines of the instructional texts can promptly display information about the selected menu items—year, height by, and color by. The texts' strings are directly accessed by the *Ascript* to update or reset the values of the strings when the menu buttons are clicked. The text strings are predefined in a series of arrays and variables. The value in an array is retrieved when a corresponding menu button is selected.

```
Text2String = new Array('1. Year: 1991', '1. Year: 1992', '1. Year: 1993', '1. Year: 1994', '1. Year: 1995', '1. Year:
   1996', '1. Year: 1997', '1. Year: 1998', 'Model is reset. Please select Year, Height, Color');
Text3String = new Array('2. Height: Electricity  Cost x 0.0005',     '2. Height: Water Cost x 0.0005', '2. Height:
   Natural Gas Cost x 0.0005');
Text4String = new Array('3. Color: Electricity Cost', '3. Color: Water Cost', '3. Color: Natural Gas Cost');
DefText2 = '1. Select Year'                                //default Text2     string
DefText3 = '2. Select Variable for Height';                //default Text3     string
DefText4 = '3. Select Variable for Color';                 //default Text4     string
DefText5 = 'Click on a Bldg to View Data Value';           //default Text5     string
DefText6 = '';
```

Figure 17. Instructional texts were predefined in *Ascript*.

Height Scale Slider



Because of the heights of the buildings are used to represent the energy data which can be as high as 979,000, the buildings will be so tall that, in the predefined views, viewer cannot see the complete model. The Height Scale control was created to allow viewer to manipulate the heights of the buildings to a preferred size and fit all the buildings in the window. The scaling of the buildings' height maintains the relative proportion of the data values of the selected energy variable of the selected year.

The Height Scale slider is equipped with a *PlaneSensor* that receive event from the user and send the event to the slider's marker to change its position, and to a Script, *sliderScript*, to change the scale of the buildings as well as to change the text string of the scale factor. The scale controlled by this slider ranges from 0.01 to 200%. There is also a text string at the top of the slider's ruler displaying the maximum value of the selected energy variable of the selected year. This is to provide further information for the user to understand the relative proportion of the buildings' height while manipulating the scale control.

Figure 18. Height Scale slider.

The Scripts

There are a total of five Scripts to facilitate the interactivity among the buildings and the dashboard elements. Figure 19 illustrates how events are to be passed to from and to the Script and VRML nodes. Complex paths of events are centered at the *AScript*, the Script that contains the energy database (see Figure 20) and directly accesses all the buildings' geometries. The Scripts were created to correspond to the set of VRML components described above. Two of the Scripts, *buttonScript* and *menuScript*, handle the events from the menu items. They assign proper identifier values for the user's selections of variables. Then, the values are sent to the *AScript* to index the values in the database arrays. When a Height by variable is selected for a year, new height values are calculated and used to replace the vertical scale of the buildings. Similarly, when a Color by variable for a year selected, new color values are calculated for the Red value of the buildings' *diffuseColor*. Moreover, *AScript* interact directly with the animation components (*AnimPS*, *AnimTime*, and *tAnim*). The *buttonScript* receives values when a Year button is clicked. The Script assigns a pointer value for the selected year and directly access the selected button's prototype to switch its appearance. The year pointer is, then, sent to *AScript*. While the *AScript* store the value from the *buttonScript*, it accesses line 1 of the instructional text, *Text2*, and update the string to display the value of the selected year. The menuScript receives events from the Height by, Color by, and the three energy types—Electricity, Water, and Natural Gas—menu buttons and assigns index values for the selected items. It, then, passes the values to the AScript for functions to interact with the buildings' properties. In addition, the instructional texts, *Text3* and *Text4*, will be updated to display the current selected menu items for Height and Color.

The *buttonScript* will enable the *TouchSensors* on the buildings when a Year menu is selected. Each building's *TouchSensor*, once enabled, will receive an event from user and send it to the *BScript* for indexing. The selected building index is then sent to the *AScript*. The *AScript* will select proper data values of the selected building and selected year to update the text strings in the instructional texts, *Text5* and *Text6*. Another feature implemented in the interaction with the buildings is to adjust the color property of the selected building. The *AScript* receives *BTouch* index value for the selected buildings and replace a new value, 0.8, to the blue channel in the RGB color property of the selected building. Clicking on the menu button of the variable selected for Color by menu will reset the color representing the selected data value.

The *sliderScript* interacts with the Height Scale slider to manipulate the height scale of the buildings representing the data value to fit user's viewing preference. The Script also directly interact with the Height Scale components on the dashboard.

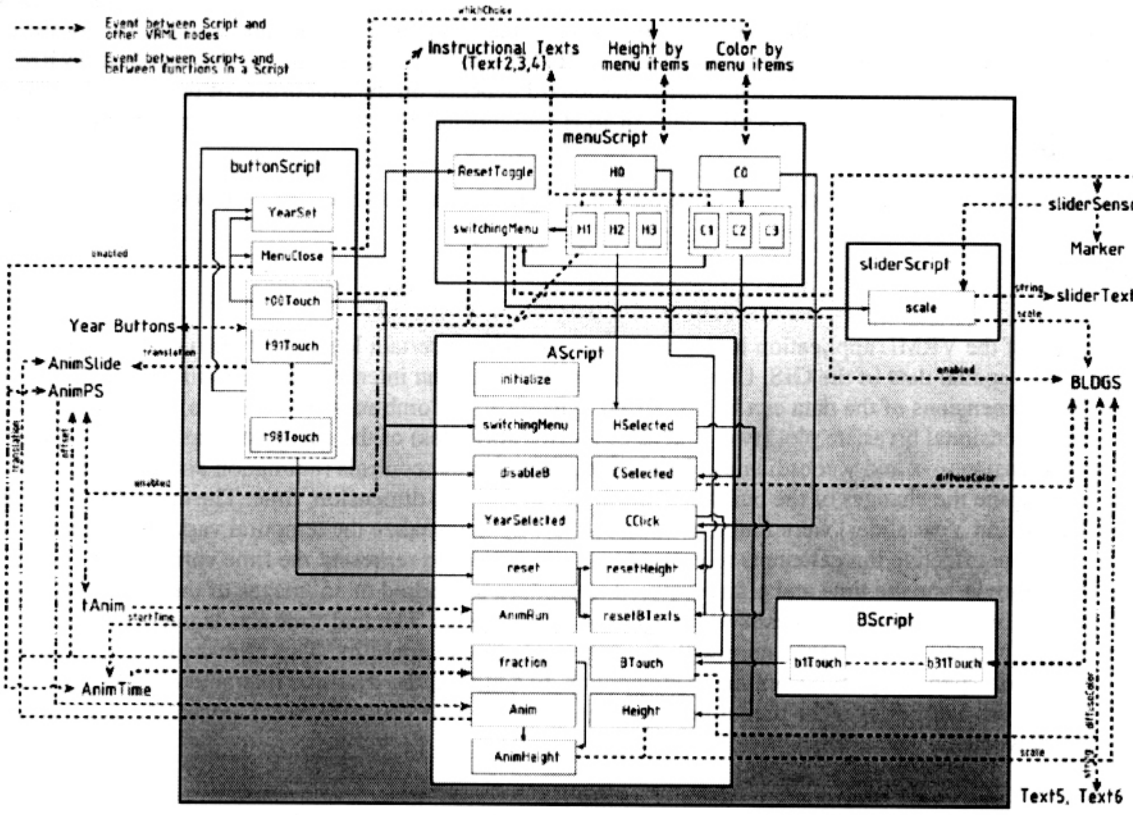


Figure 19. Scripts and functions.

```
url "javascript:
    function initialize() {                          //the main database
        BldgNo = new Array(394,395,396,397,400,402,403,404,406,407,414,415,420,424,427,429,4
        BldgName = new Array('LURIE ANN AND ROBERT H TOWER','BAGNOUD FRANCOIS-XAVIER BUILDIN
        Bldgft3 = new Array(355150,12102940,13059483,1705886,8570578,1069625,5661654,5490909,
        e91 = new Array(0,0,0,0,62095.83,22503.6,52333.68,128318.4,88574.99,470642.05,113281.
        e92 = new Array(0.00,5931.73,0.00,0.00,69396.77,20470.8,60177.6,134769.6,77930.5,449
        e93 = new Array(0.00,97614.72,0.00,0.00,82536.39,20210.4,58816.8,129175.2,76962.82,5
        e94 = new Array(0.00,208897.92,4227.88,4.87,81619.45,18144,59028.48,123883.2,71672.8
        e95 = new Array(0.00,185587.2,21552.36,5499.2,89776.08,16592,54998.4,114672,70164.48,
        e96 = new Array(3906.96,215539.2,215720.82,62990.48,88757.68,20432,59260.8,117024,64
        e97 = new Array(13469.42,217612.8,509221.04,108032,88892.48,19776,67769.6,113520,654
        e98 = new Array(10028.88,244454.4,559571.12,107409.92,118080,18976,68937.6,118464,67
        e99 = new Array(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1); //fo
        w91 = new Array(0.00,0.00,0.00,0.00,8130.96,1335.44,7081.96,21247.96,935.13,31522.39,
        w92 = new Array(0.00,0.00,0.00,0.00,7899.8,3121.55,4248.05,18544.7,987.65,40357.87,2
        w93 = new Array(0.00,31423.58,0.00,0.00,14267.9,6721.39,7516.33,21387.86,1102.87,531
        w94 = new Array(0.00,55356.23,0.00,0.00,19966.8,3225.08,5757.32,22988.38,1137.92,550
        w95 = new Array(0.00,524.9,0.00,0.00,15819.62,3784.99,1308.86,24305.83,1212.31,55788.
        w96 = new Array(86.64,70868.95,0.00,0.00,22079.16,7205.84,1482.96,27884.07,1784.55,5
        w97 = new Array(101.25,56862.95,13446.64,1807.1,14926.82,14607.3,5381.35,28186.11,14
        w98 = new Array(271.01,53530.87,14196.85,2427.83,5248.32,9950.65,5447.81,32046.42,14
        w99 = new Array(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1); //fo
        ng91 = new Array(0.00,0.00,0.00,0.00,20626.16,12477.54,24328.38,29333.85,3111.75,598
        ng92 = new Array(0.00,0.00,0.00,0.00,26400.17,8407.25,20442.23,24648.42,7376.78,7649
        ng93 = new Array(0.00,26886.07,0.00,0.00,26961.42,9687.52,20274.44,24446.07,7352.65,
        ng94 = new Array(0.00,37571.76,0.00,0.00,28094.67,13480.85,19323.6,19458.89,10028.92,
        ng95 = new Array(0.00,27961.91,0.00,0.00,21181.84,7763.75,19371.79,23355,8255.05,697
        ng96 = new Array(0.00,40504.86,59370.22,13377.48,30089.18,11639.48,25107.22,30258.51,
        ng97 = new Array(0,46215.15,113445.79,0.00,31658.2,15137.14,29829.02,35968,11174.59,
        ng98 = new Array(0.00,40249.66,91596.79,18946.56,26266.99,11206.9,25739.18,31036.32,
        ng99 = new Array(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1); //f
        eMax = new Array(964807.2,979433.28,908960.64,913839.36,820841.6,857996.8,814390.4,8
        wMax = new Array(55865.32,53081.84,73292.72,61088.95,59586.27,70868.95,74955.07,1178
        ngMax = new Array(111018.1,112081.51,121296.34,124283.07,95386.16,139744.41,145404.2
        bldgHeight = new Array(36,16,18,16,8,8,8,8,8,16,12,4,24,8,12,12,15,24,12,4,8,8,12,8,
        noOfBldgs = 31;
```

Figure 20. Energy data arrays in Ascript.

The Product

There are 7 .wrl files—one master file, 4 prototypes, and 2 external models—used in this application of VRML visualization of the University of Michigan North Campus energy consumption (cost) from 1991 to 1998, along with 34 images created for the appearances of the interactive buttons and the contextual ground, and a sound file for the effect when an object is clicked. The scene was embedded into an HTML format to control the proportion of the VRML scene. This is because the visibility of the menu items on the dashboard depends on the proportion of the projection plane (window or frame size). The embedded VRML scene in an HTML document prevent any adjustment of the proportion, thus, proper view is maintained.

(The interactive model can be access at http://www-personal.engin.umich.edu/~tnac/vrml/GISVisualization)

Conclusion

The development of the VRML application in this study accomplishes a certain levels of dynamic real-time interactive visualization of the spatial data of the GIS. User can interact with the menu interface and also directly with the buildings themselves. Five dimensions of the data can be visualized dynamically. Combination of objects' heights and colors are used as a three dimensional bivariate plot (two dimensions—two attributes) of the energy data at their geographical location (two dimensions—x and y coordinate). In addition, the temporal controls (animation and slider through years) allows user to examine the changes of the buildings' attributes in the fifth dimension, time. The animation features (Play/Puase button and Year slider) were found to be effective way to visualize the temporal variables. This capability of VRML is far more effective than attempts to create computer movies to represent the time variable. The principle problem with the movie was the time and effort required to produce a hundred or so images of the models and record each one into the movie sequences (Openshaw, Waugh, and Cross, 1994). The real-time six-degree-of-freedom navigation in VRML enables the visualization with superior exploratory capability. This allows user to explore the data—spatial and non-spatial aspect—from many viewpoints. In addition, the dynamic interactive VRML model provides additional advantages over the predefined animation movies with ability to select any single or a pair of variables to be viewed.

This study focused on the interactive and dynamic manipulation of the visualization of the spatial/geographic information rather than the realism of the scene. Because in virtual reality application the performance of the system varies upon the level of interactivity and the realism. There is a trade off for the virtual reality application between the level of interactivity and the realism. The two factors have an inverse relationship to each other. When both are high, the system performance would be impeded. Thus, a virtual reality developer must carefully consider what would be the optimal combination of these factor in order to accomplish the objectives of a specific visualization of the geographic information.

The future of the VRML visualization of geographic information would be improved with the improvements in two areas—hardware performance and new generation of the VRML standard and specification. The computing power of the computers was predicted to accelerate to faster speed and higher capability in 3D graphics. This would significantly affect the development and the application of virtual reality. In addition, there has been an attempt, by the Web3D consortium, to improve the specification of the VRML 97 standard to add more features and capabilities of future VRML, X3D, to handle geographic information. For example, in the new draft of the X3D standard, there are several new nodes to facilitate the terrain models and georeferencing coordinate system; these nodes are: GeoCoordinate, GeoElevationGrid, GeoInline, GeoLocation, GeoLOD, GeoMetadata, GeoOrigin, GeoPositionInterpolator, GeoTouchSensor, and GeoViewpoint. In addition, there are new type of geometry to be included in the new generation of VRML; that is the NURBS (Non-Uniform Rational B-Spline). The new nodes for NURBS would include NurbsCurve, NurbsCurve2D, NurbsGroup, NurbsPositionInterpolator, NurbsSurface, and NurbsSurfaceTextureCoordinate (Web3d.org). These new nodes, geographic and NURBS, would improve the capability of the virtual reality application using the VRML, not only the new types of geometry but also the new interactive functions (interpolators and sensors). Once the new standard is released and accepted, it would be implemented by many of the 3D modeling software packages as a transportable 3D file format, and also, hopefully, there will be development of a new type of softwares that integrate this standard with the 3D modeling into a complete virtual reality authoring packages.

Bibliography

Berry, Joseph K., David J. Buckley and Craig Ulbricht (1998), Visualize Realistic Landscapes: 3-D Modeling Helps GIS Users Envision Natural Resources, *GeoWorld* (http://www.gw.geoplace.com/gw/1998/0898/898vis.asp)

Carey, Rikk and Bell, Gavin (1997), *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley.

Gahegan, Mark. (2000), Visualization as a tool for GeoComputation, in Openshaw, Stan and Abrahart, Robert J. (eds.) *GeoComputation*, London: Taylor & Francis, pp. 253-74.

Gallop, J. (1994), State of the art in visualization software, in Hearnshaw, Hilary M. and Unwin, David J. (eds.) *Visualization in Geographic Information Systems*, Chichester: John Wiley & Sons, pp. 42-47.

Gillings, Mark and Goodrick, Glyn (1996), Sensuous and Reflexive GIS Exploring Visualisation and VRML, http://intarch.ac.uk/journal/issue1/gillings_toc.html.

Huang, Bo and Lin, Hui (1999), GeoVR: a web-based tool for virtual reality presentation from 2D GIS data, *Computers & Geosciences*, 25, pp. 1167-1175.

Kim, Kyong-Ho, Kiwon Lee, Ho-Geun Lee, and Young-Lyol Ha (1998), Virtual 3D GIS's Functionalities Using Java/VRML Environment, in J. STROBL and C. BEST (eds.) *Proceedings of the Earth Observation & Geo-Spatial Web and Internet Workshop '98*, Volume 27. Instituts für Geographie der Universität Salzburg. http://www.sbg.ac.at/geo/eogeo/authors/kim/kim.html.

MacEachren, A. M. and Taylor, D. R. F. (ed.) (1994), *Visualization in Modern Cartography*. Oxford, UK: Pergamon.

MacEachren, A., et al (1994) Introduction to Advances in Visualizing Spatial Data, in Hearnshaw, Hilary M. and Unwin, David J. (eds.) *Visualization in Geographic Information Systems*, Chichester: John Wiley & Sons, pp. 51-59.

MacEachren, A. M. (1998), *Visualization-Cartography for the 21st century*. http://www.geog.psu.edu/ica/icavis/poland1.html.

Openshaw, S., Waugh, D., and Cross, A. (1994), Map Animation as a Spatial Analysis Tool, in Hearnshaw, H.M., and Unwin, D.J. (eds.) *Visualization in Geographic Information Systems*, Chichester: John Wiley & Son, pp.131-138.

Roehl, B., et al. (1997), *Late Night VRML 2.0 with Java*. Emeryville: Ziff-Davis.

Turk, A. (1994), Cogent GIS Visualizations, in Hearnshaw, Hilary M. and Unwin, David J. (eds.) *Visualization in Geographic Information Systems*, Chichester: John Wiley & Sons, pp. 26-33.

Web3d Consortium (2000), *The Virtual Reality Modeling Language International Standard ISO/IEC 14772:200x*,http://www.web3d.org/TaskGroups/x3d/specification/index.html.