SGG 3643
Computer Programming III

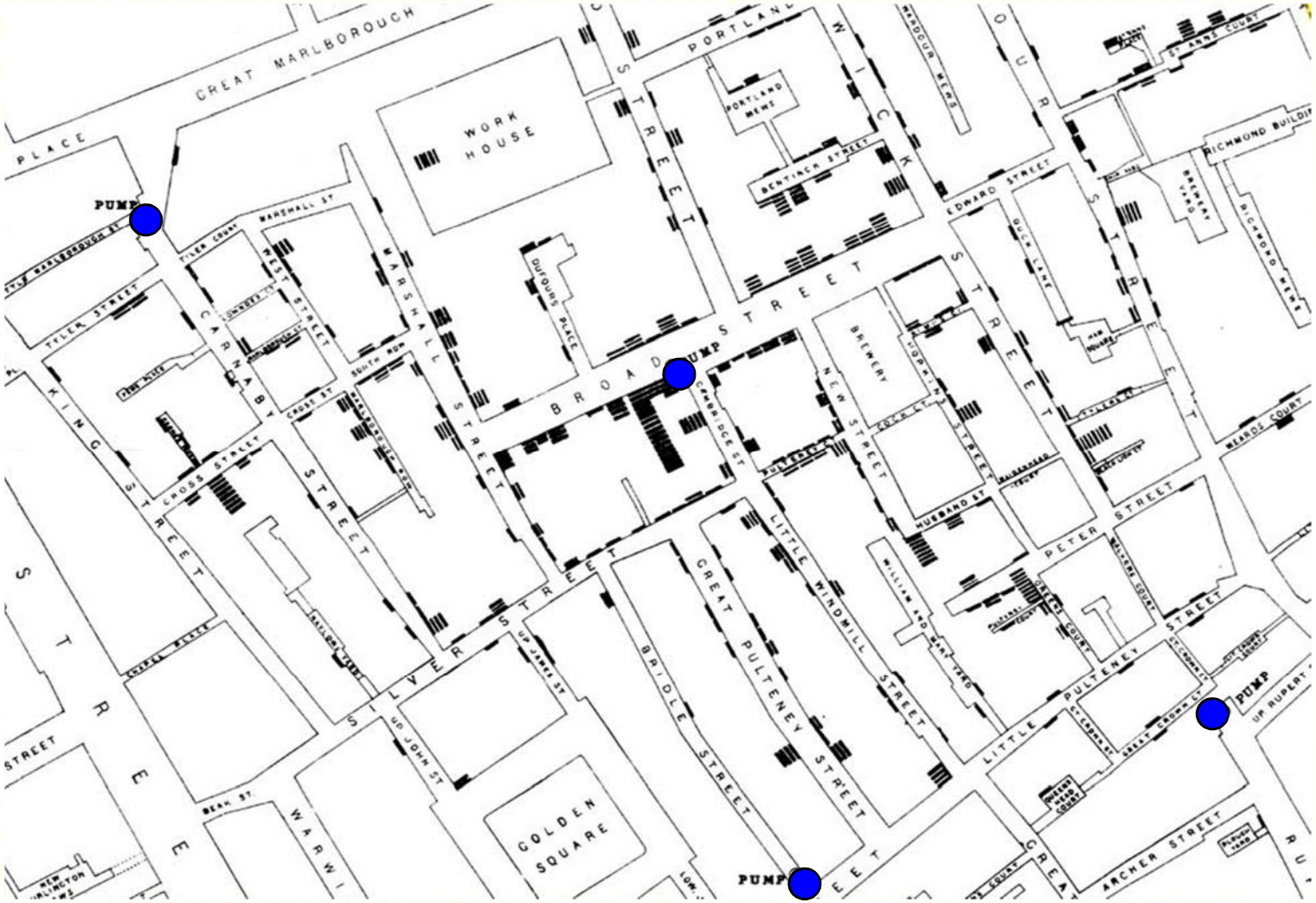
GeoVisualization

Ivin Amri Musliman

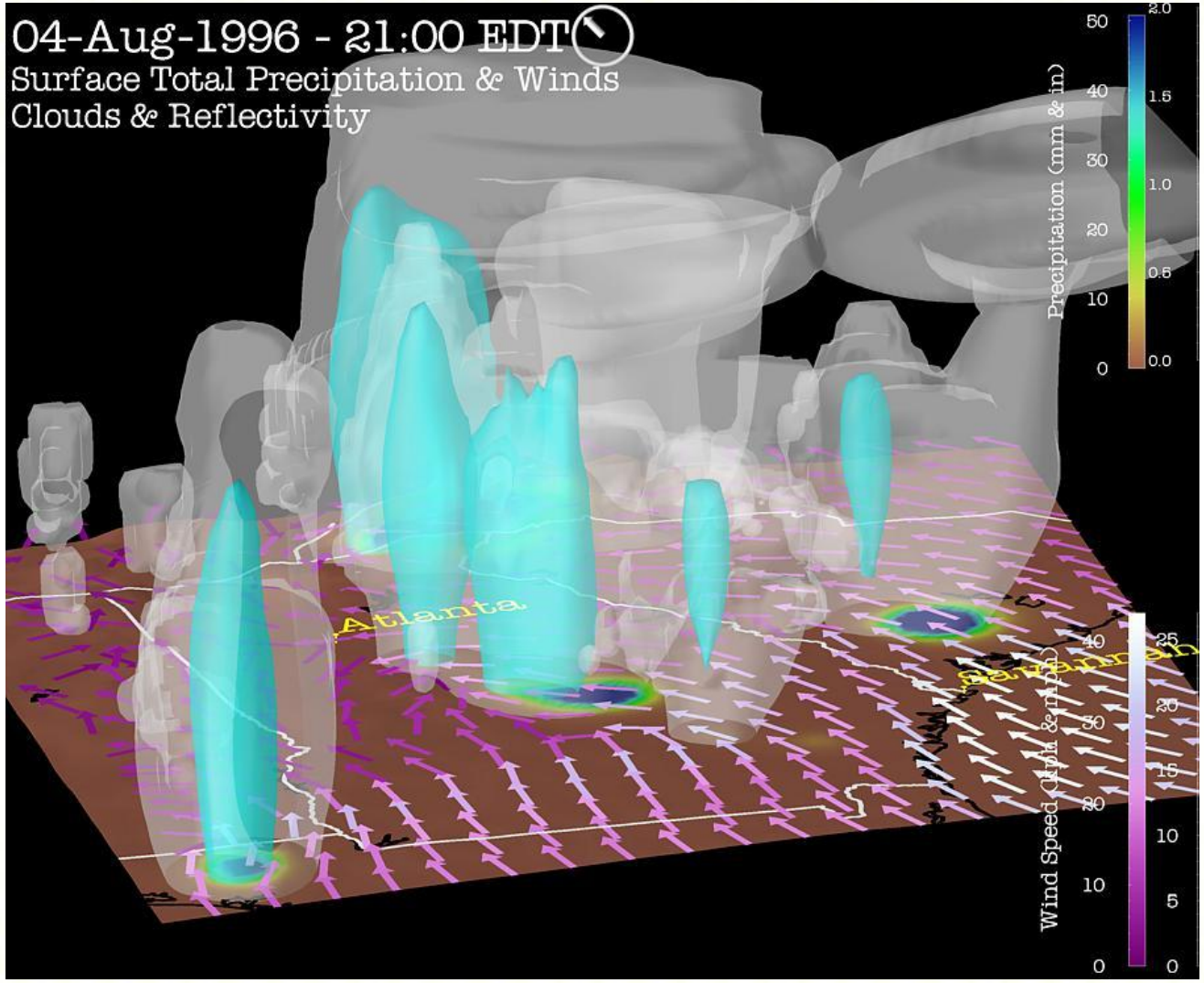
Outline

- Introduction
- Fundamentals
  - VRML, X3D, GML, CityGML, KML
- Multiresolution Models
  - LoD
  - Progressive Meshes
  - Compression
- Augmented Reality

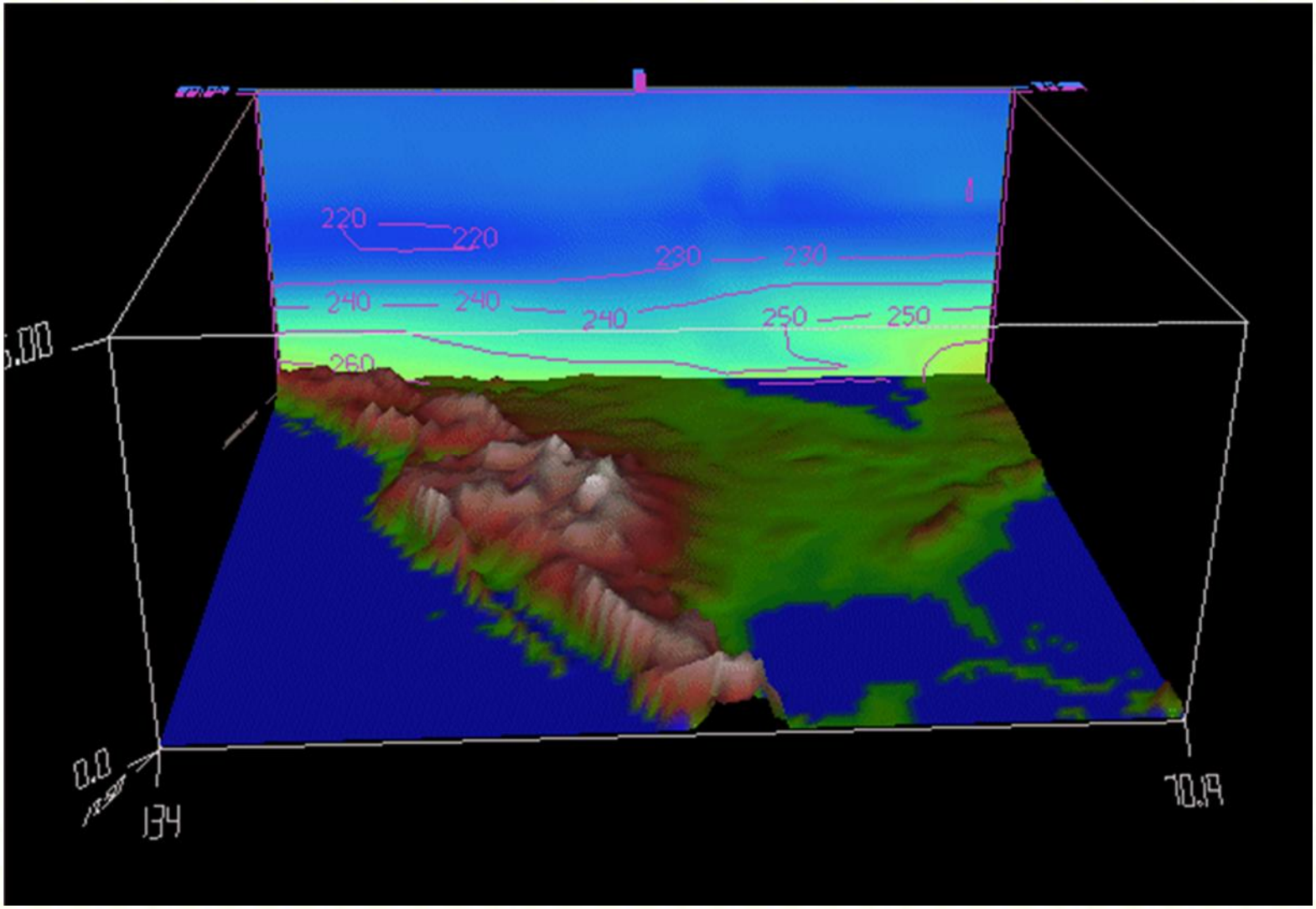
Example 1: Cholera Breakout London 1854



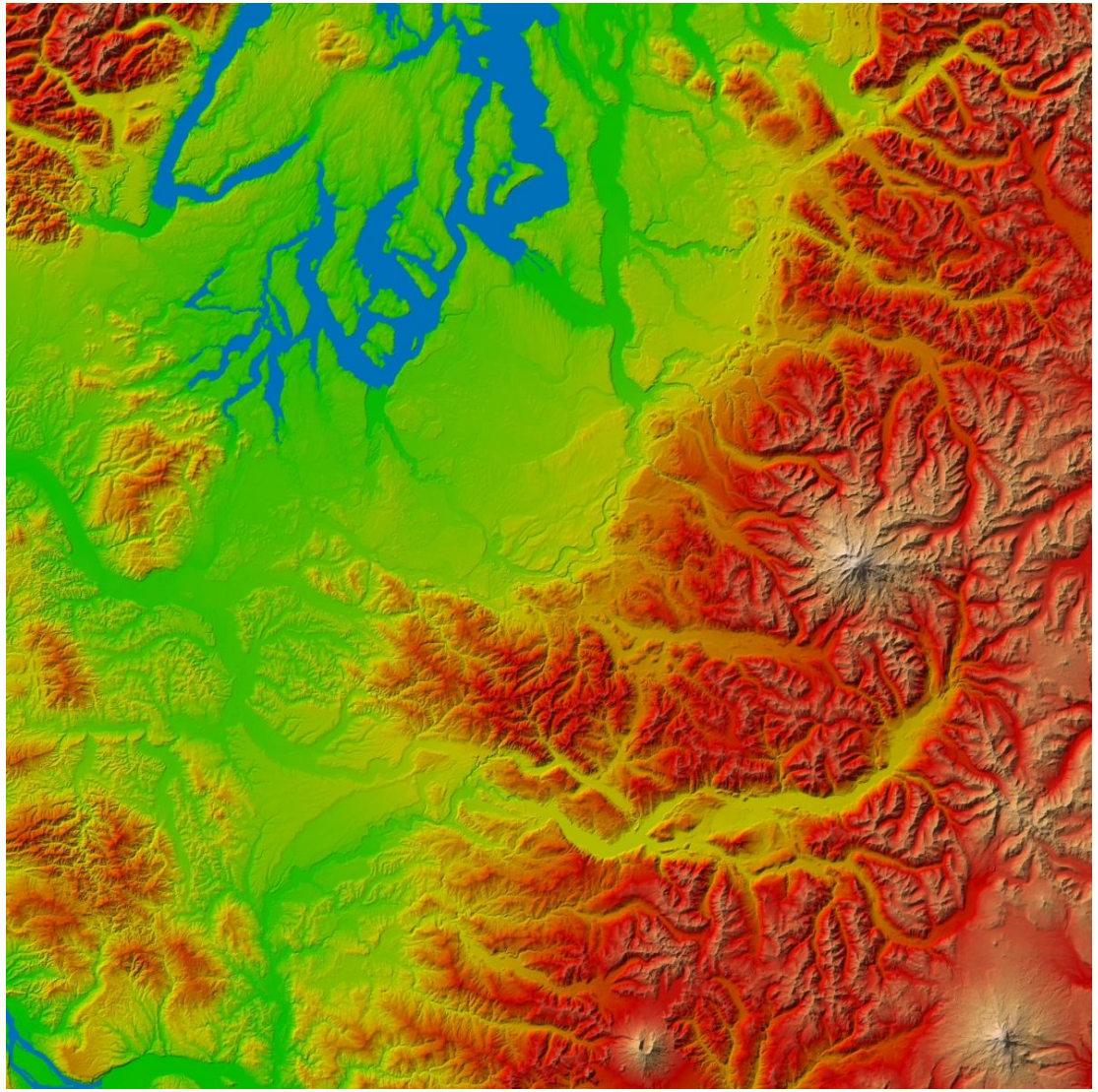
Example 2: Scientific Visualization - Weather Forecast



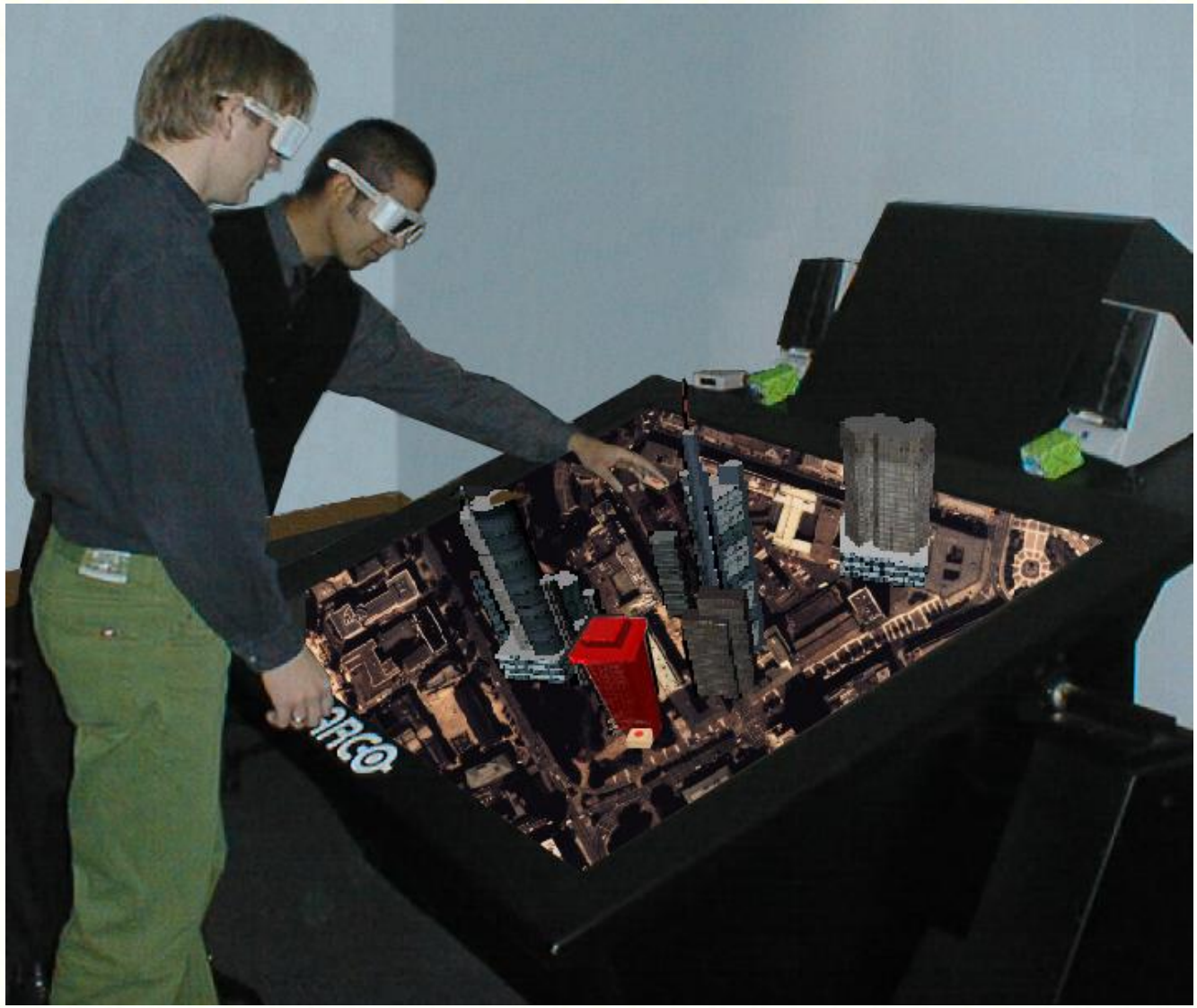
Example 3: Visual Exploration



Example 4: Terrain Visualization



Example 5: Urban Planning using Virtual Table

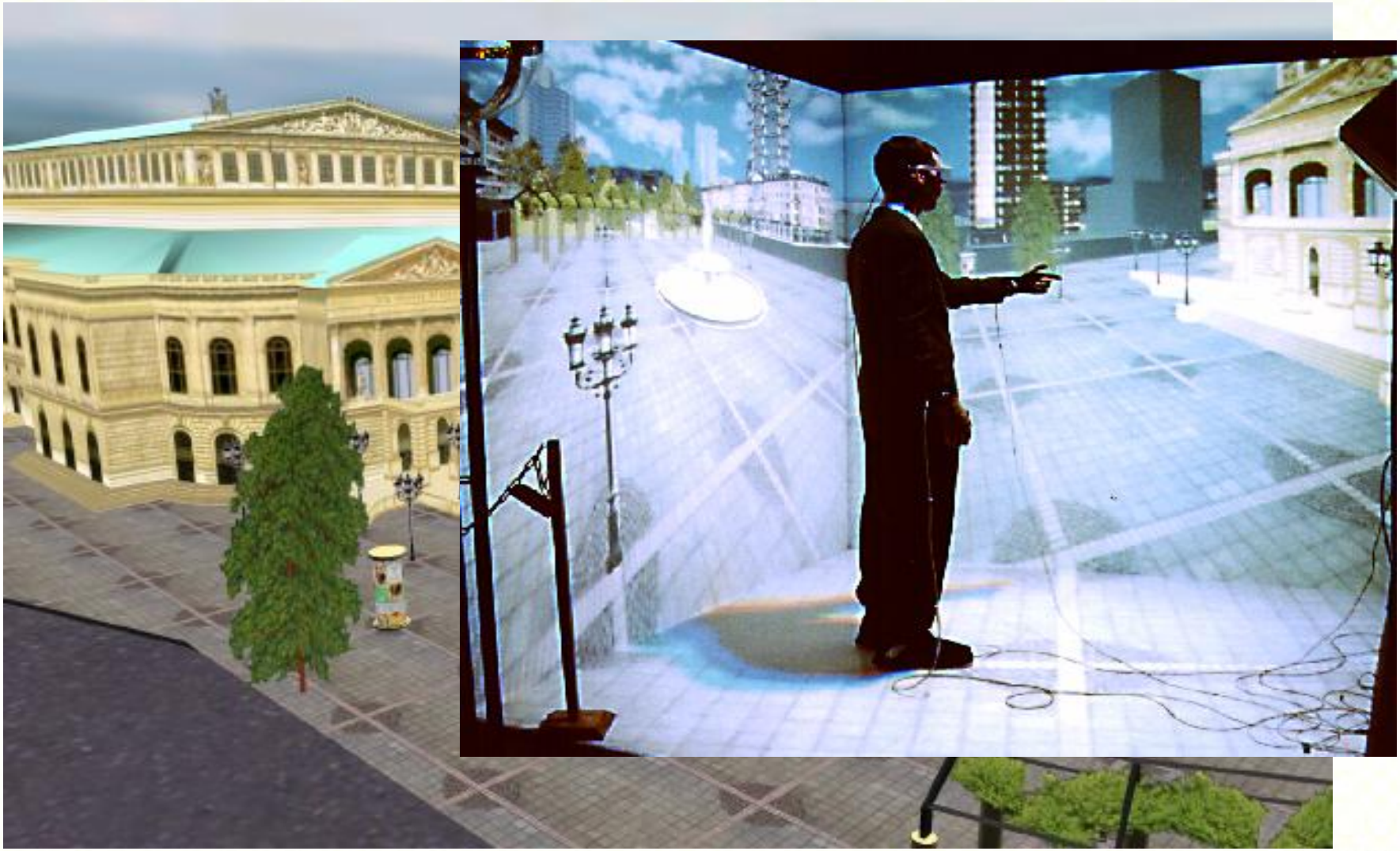


Example 6: Augmented Reality (Navigation)

Example 7: Augmented Reality (Planning)

Software In3D

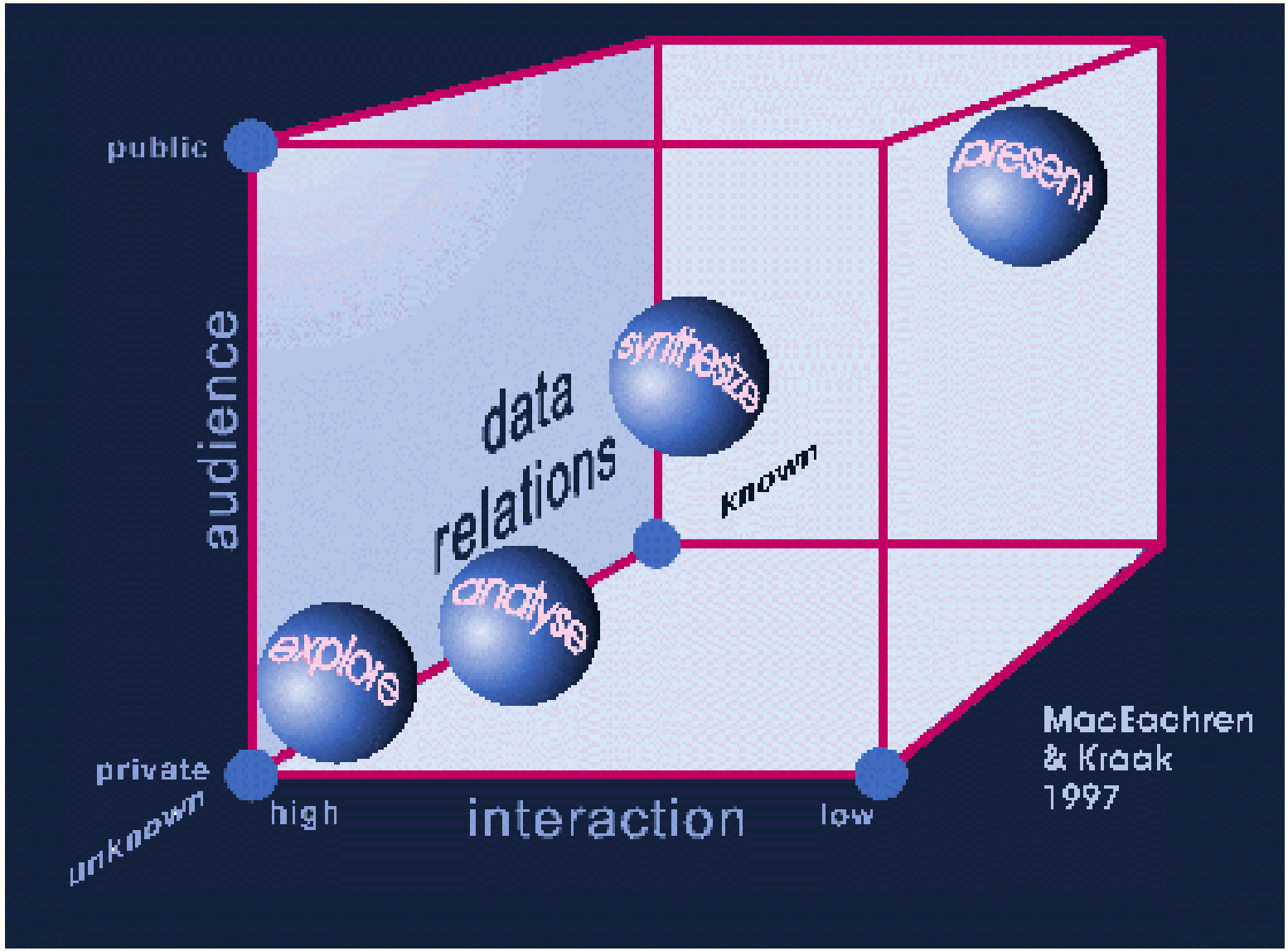
Virtual Reality



Geo-Visualization

- Making spatial data visible
- Transformation of spatial data into a picture
- “Geovisualization can be defined as a field on the use of visual geospatial displays – including Virtual Environments – to explore data and through that exploration to answer questions, generate a hypothesis, develop problem solution, and construct knowledge.”

MapCube



VRML

Introduction to VRML (1/7)

- **V**irtual **R**eality **M**arkup **L**anguage
- VRML markup language meant for displaying 3D object on the Web (with a plug-in) and allows users interaction and exploration.
- It is a scene description language. It is not a programming language.
- Created by working group in Web3DC (Web 3D Consortium) on 1994.
- The website: www.web3D.org or www.w3.org/MarkUp/VRML/
- The VRML97 specification – a document that describes the language.

Introduction to VRML (2/7)

- VRML file structure:

#VRML V2.0 utf8

#Comments

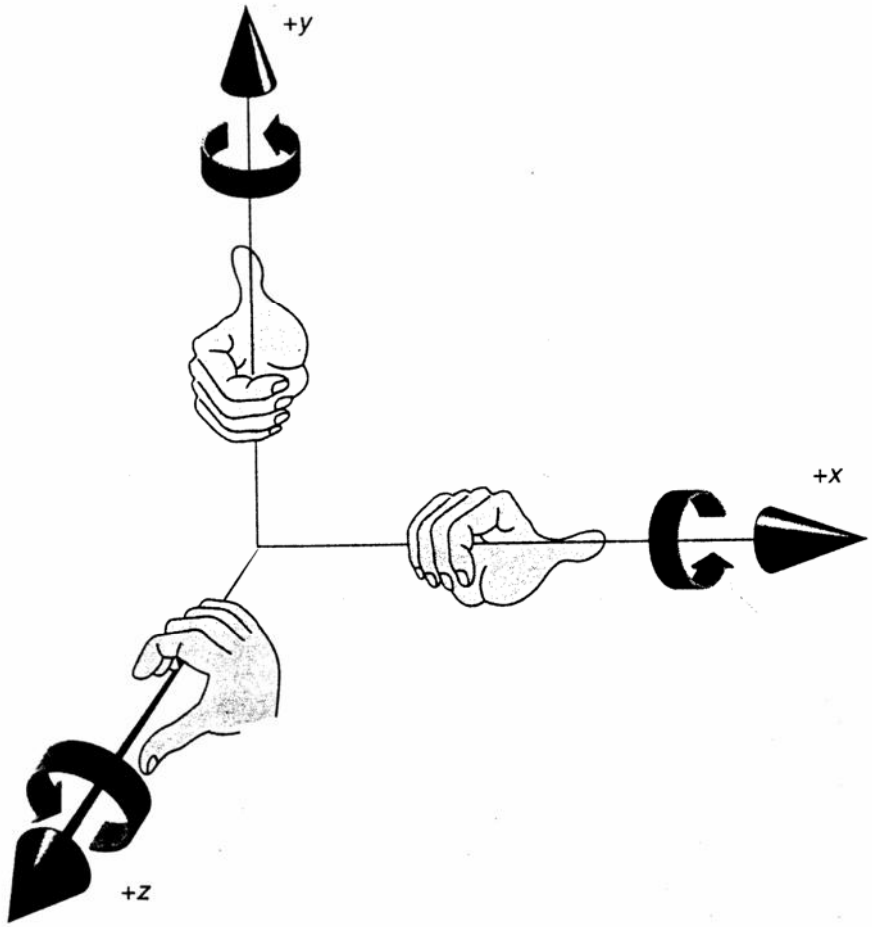
Nodes (55 nodes- **Group** & **children** node)

fields (22 fields)

Properties of the objects
(behavior, quality, appearance)

Introduction to VRML (3/7)

- XZ plane is **horizontal**, Y axis is **vertical** distance.
- VRML units – meters (linear distance), radians (angles), seconds (time), red-green-blue (RGB) (color).
- Rotation – positive value anticlockwise.



Introduction to VRML (4/7)

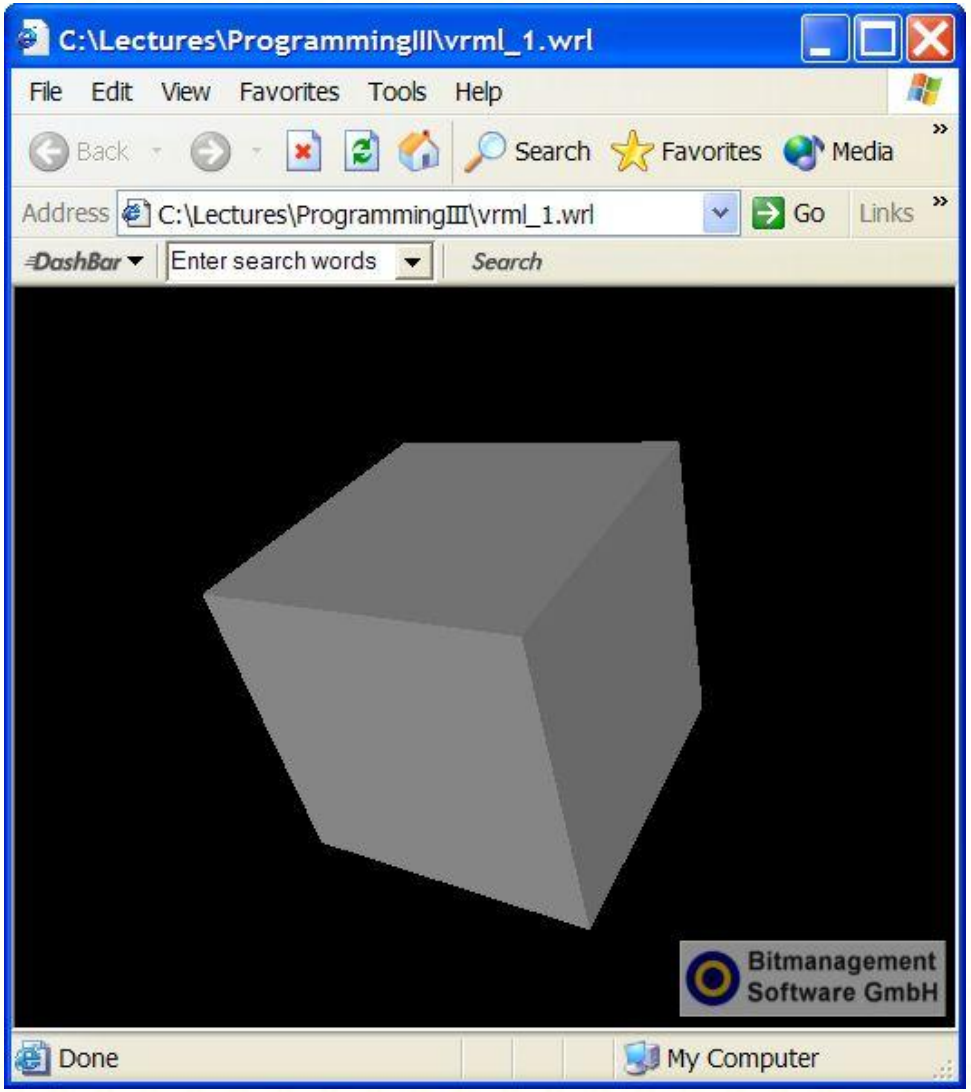
- A cube with default value

```
#VRML V2.0 utf8

WorldInfo {
     title "A cube with default value"
     info ["written by Mr Somebody"
          "FKSG, UTM, Skudai"]
     }
     DEF CUBE Shape {
          appearance Appearance   {
               material Material    {}
          }
          geometry Box {}
     }
```

Define node name

Box, Cone, Cylinder & Sphere. Default box- 2 units in each dimension (X, Y, Z), from -1 to +1

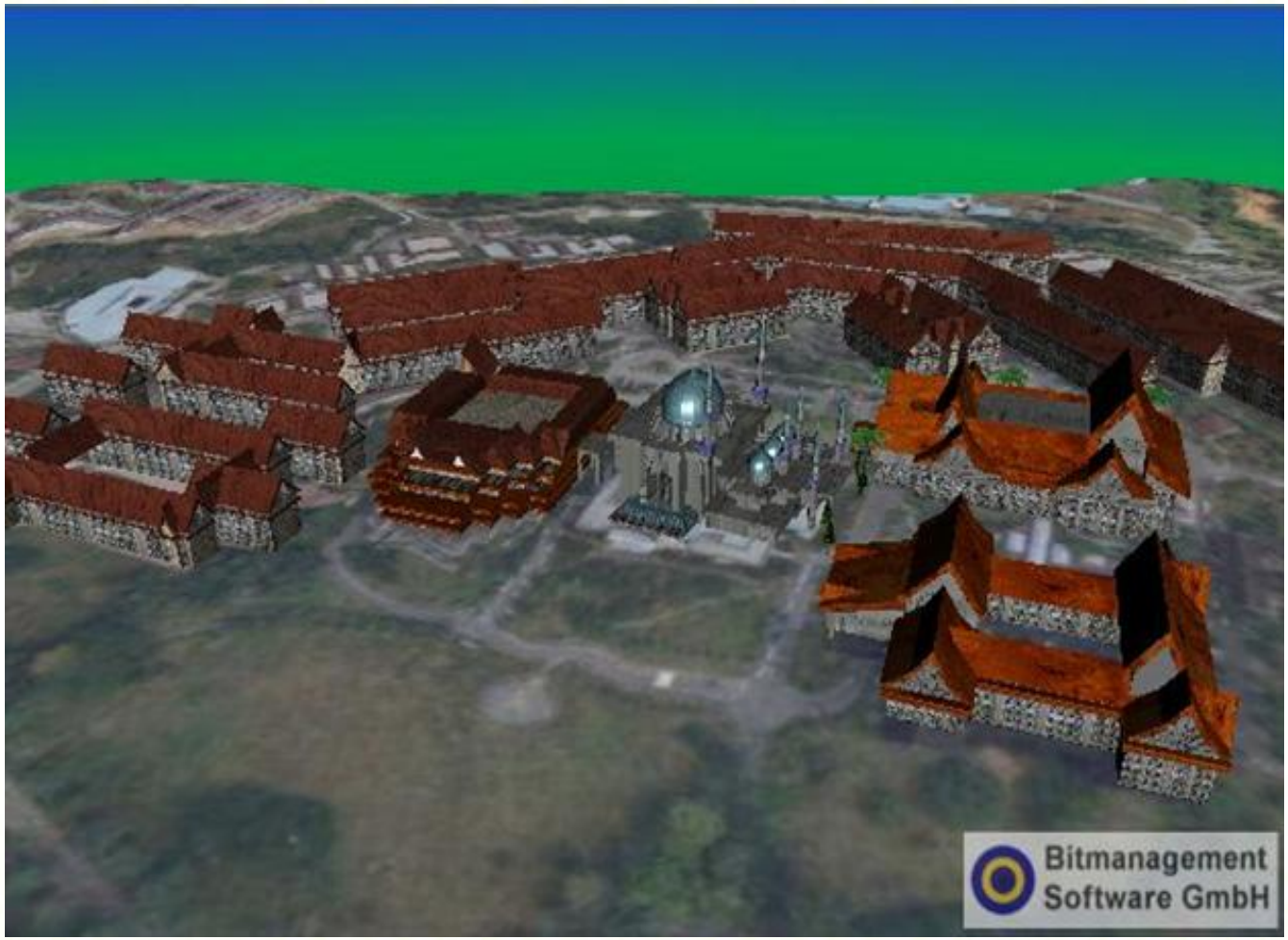
Introduction to VRML (5/7)



Simple object – cube in VRML

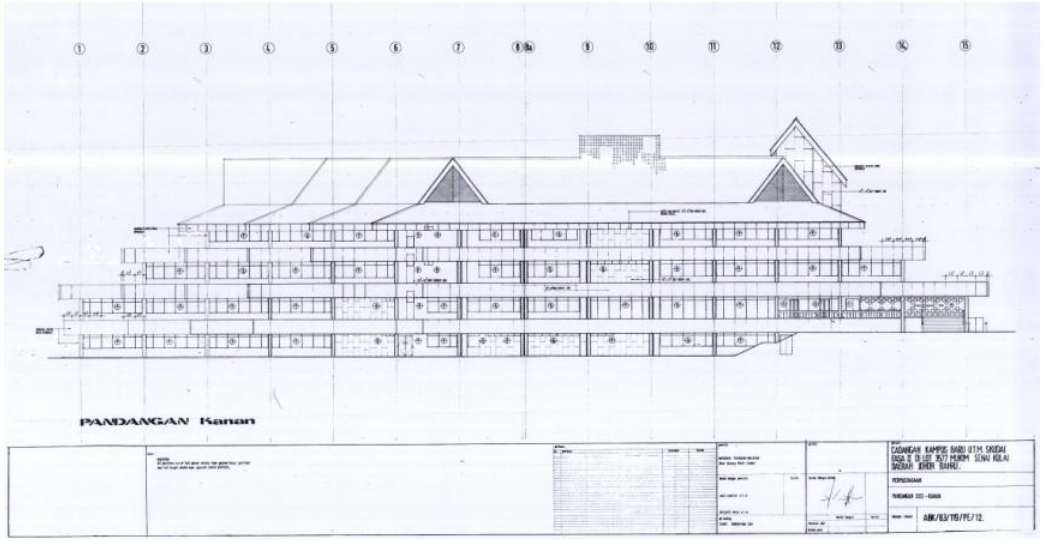
Introduction to VRML (6/7)

. . . more complex objects in VRML



Introduction to VRML (7/7)

- Data sources:



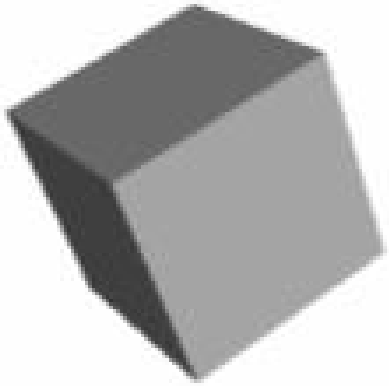
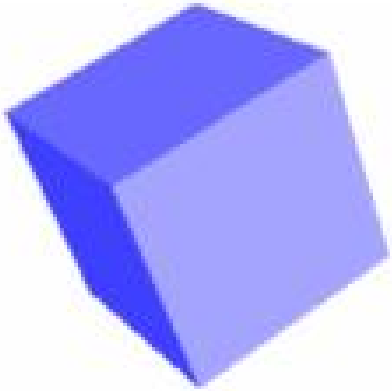
Architecture drawing

The drawing shows front, back, right, left elevation, roof plan, and the plans for every level. These drawings provide the attributes of **length**, **width** and **height** of the UTM buildings.

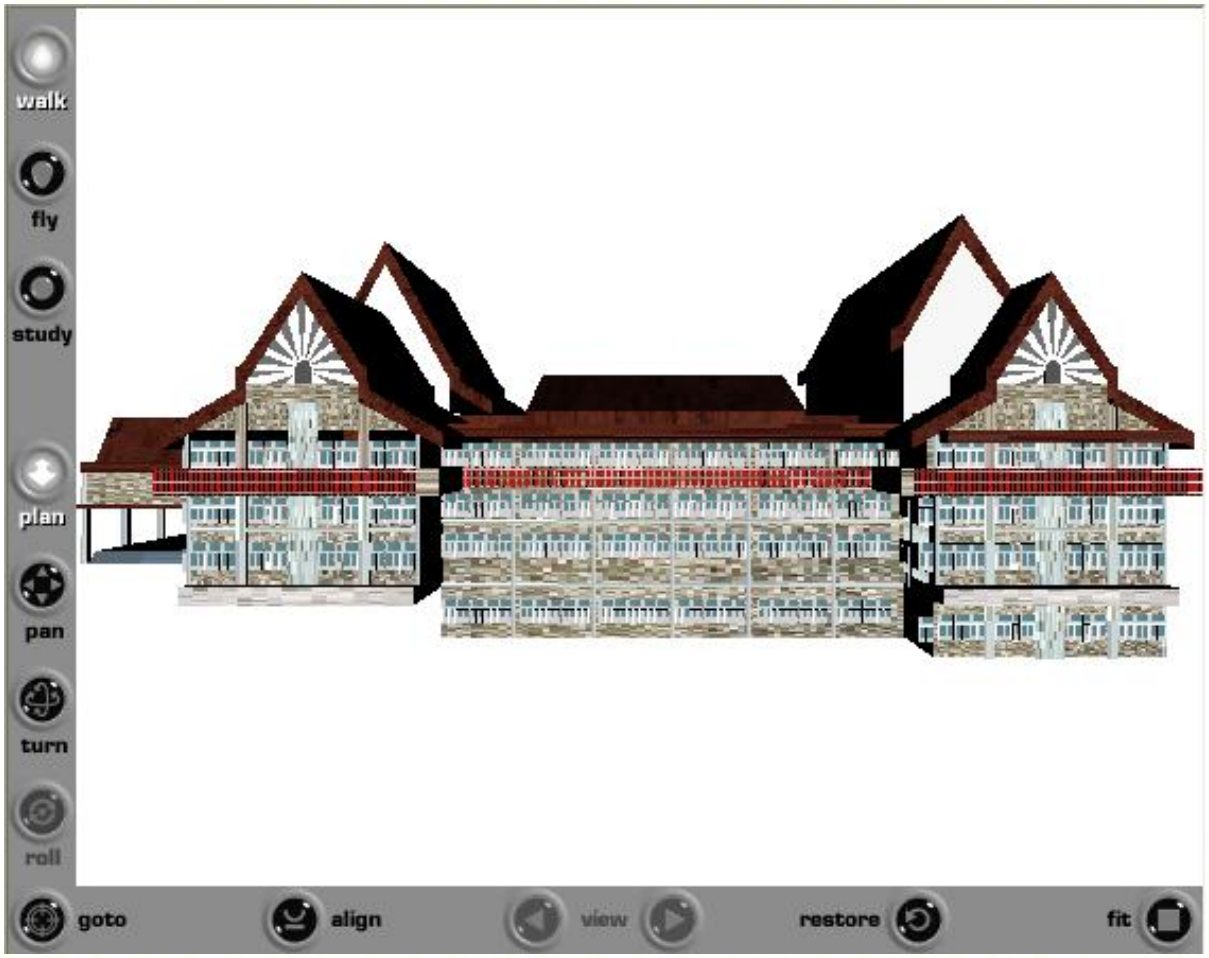
Orthophoto

Aerial photographs (year 2001) at a scale of 1: 10,000 and processed using the Leica/Helava System to produce the **DEM**, the **surface texture** as well as the **spatial coordinates** for each buildings.

Comparison among the cube based on their appearance

Cube			
Appearance	It looks dull and not attractive.	It looks well, more attractive and really good but less of realism.	It looks realistic, attractive, liking a real wooden box in the real world.

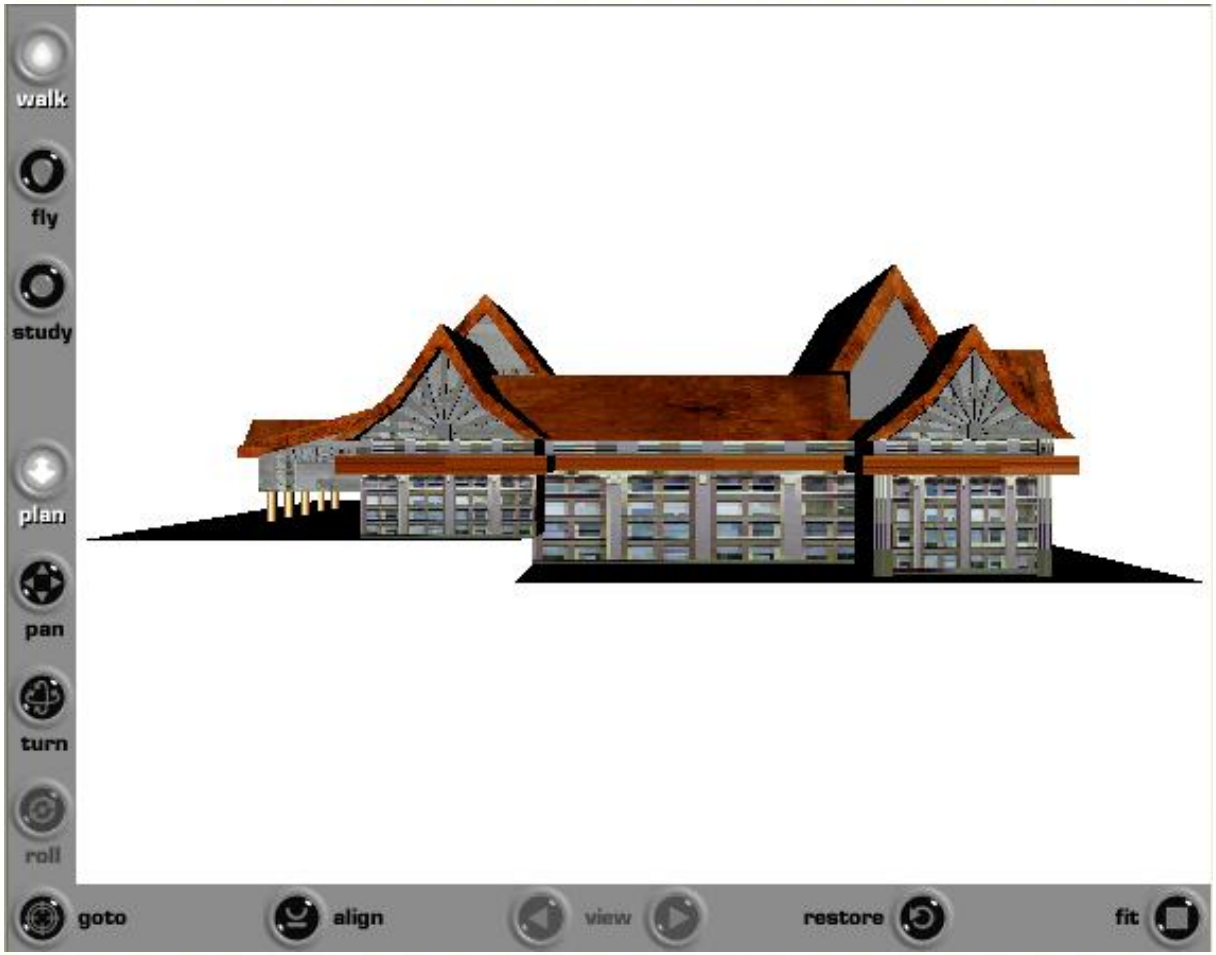
UTM administrative building created using 3D StudioMax



- More than 8 000 polygons.
- Users can't navigate smoothly and continuously through the virtual world, with scene rendering instantly on the screen as users move

Using geometry

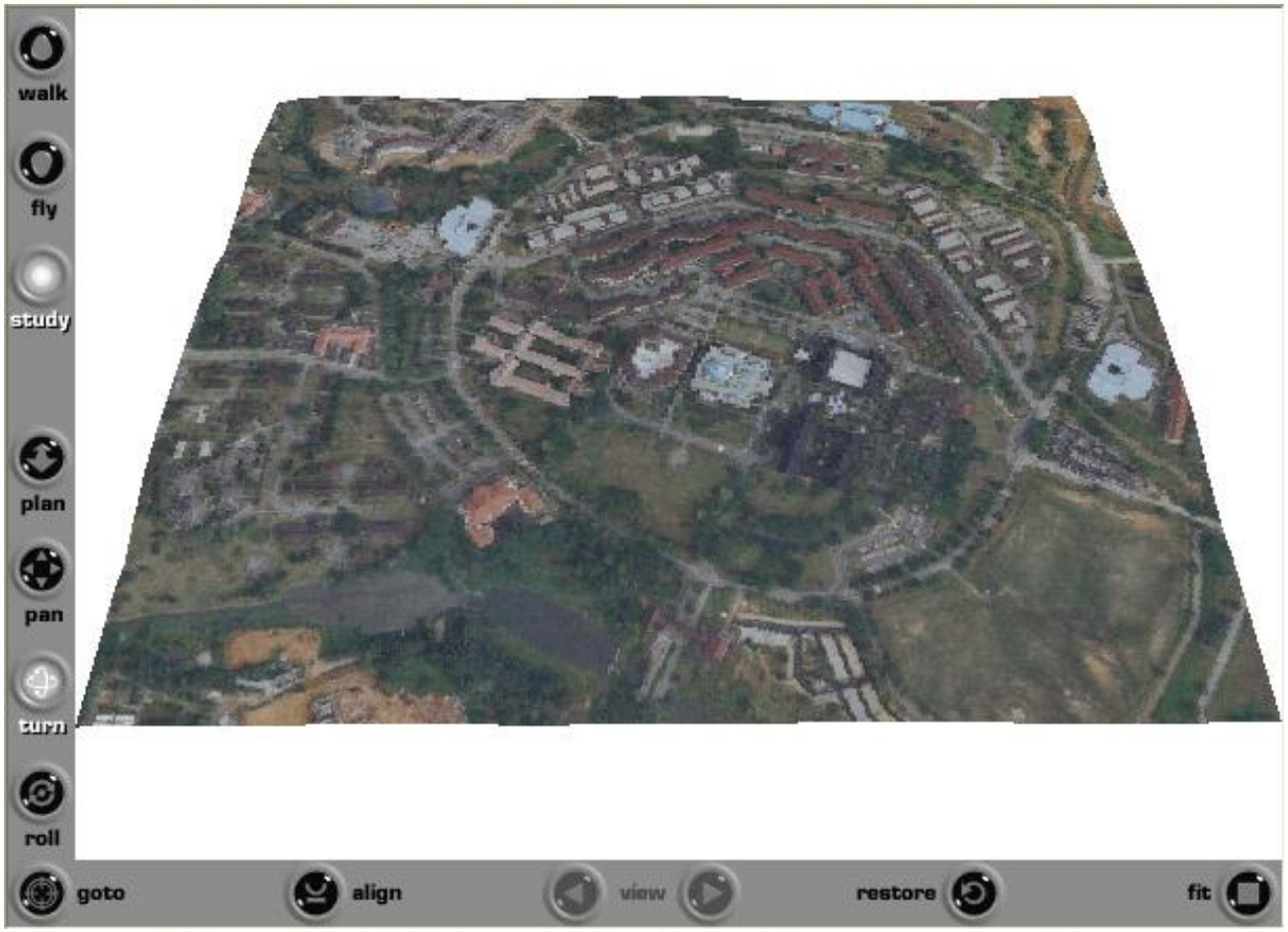
UTM administrative building created using 3D StudioMax



- The use of texture to represent the outlook of a building.
- It is cheaper in terms of performance than drawing a large number of polygons which represent of windows of the building.

Using texture

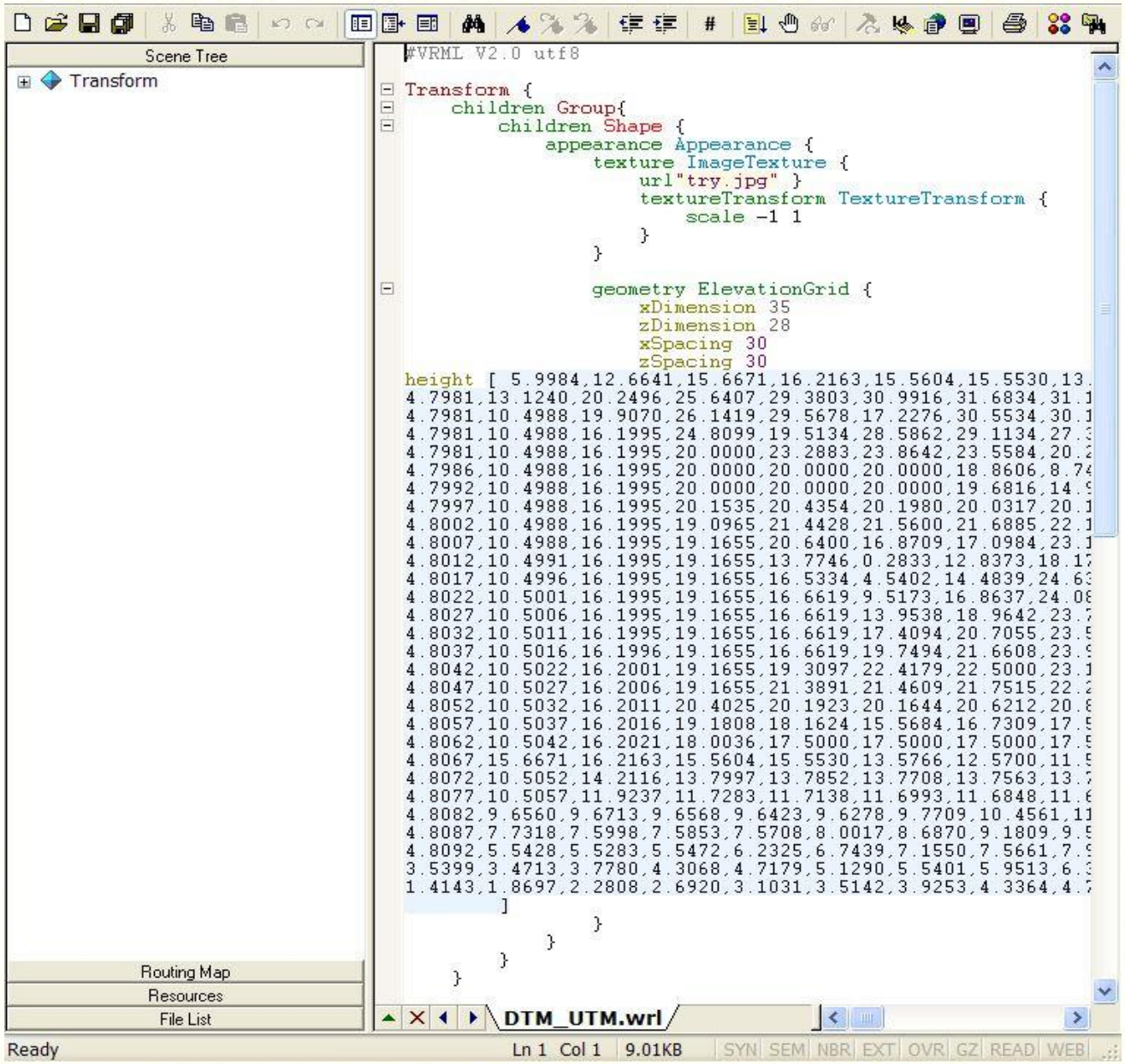
Creating terrain model with ElevationGrid node



- Not so close to the reality, may be due to the number of spacing (i.e. quite large - 30 meters).
- And, due to the interpolated contour lines.

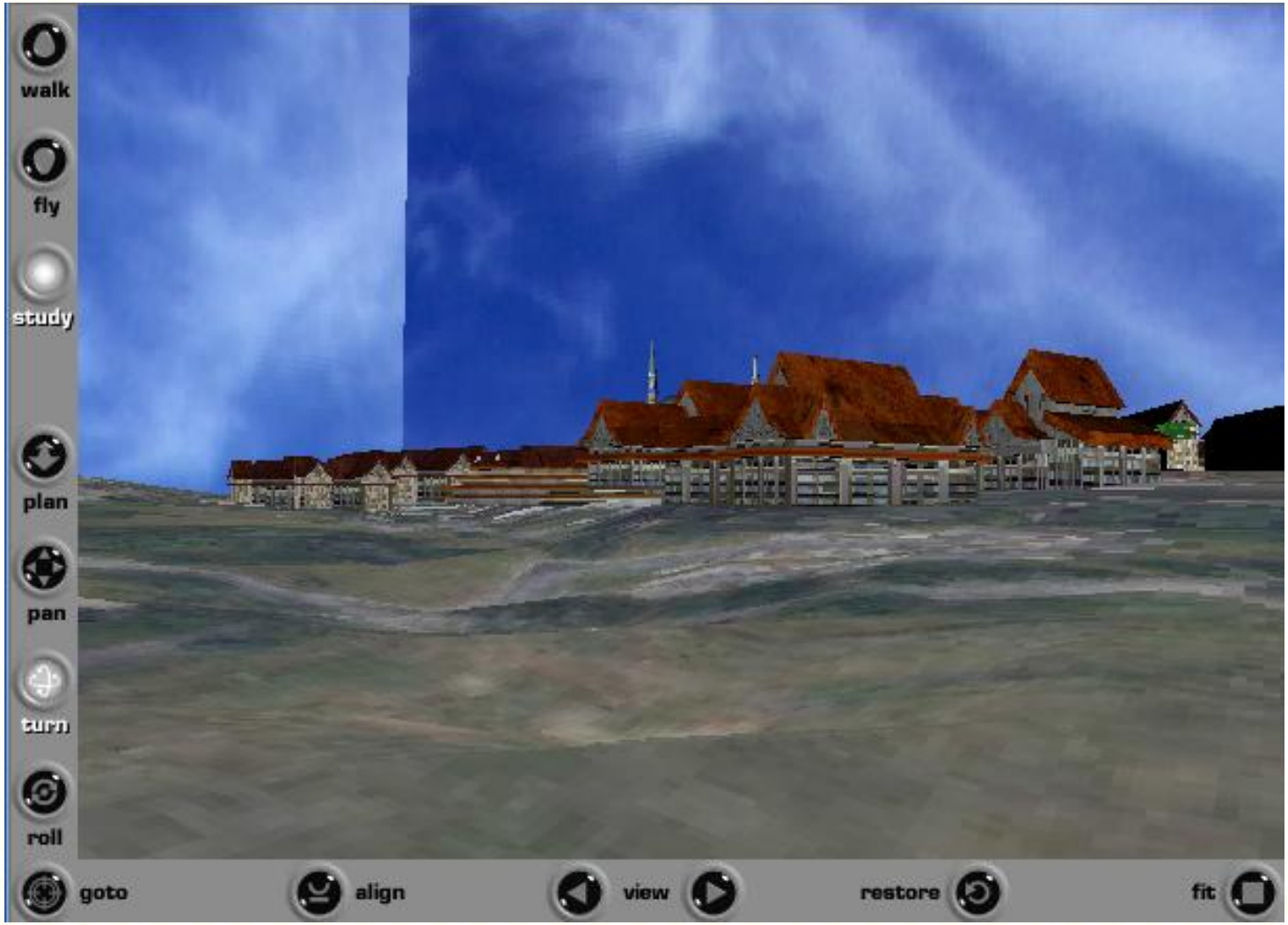
VRML node with “ElevationGrid”

```
#VRML V2.0 utf8

Transform {
    children Group{
        children Shape {
            appearance Appearance {
                texture ImageTexture {
                    url"try.jpg" }
                    textureTransform TextureTransform {
                        scale -1 1
                    }
                }

            geometry ElevationGrid {
                xDimension 35
                zDimension 28
                xSpacing 30
                zSpacing 30
```

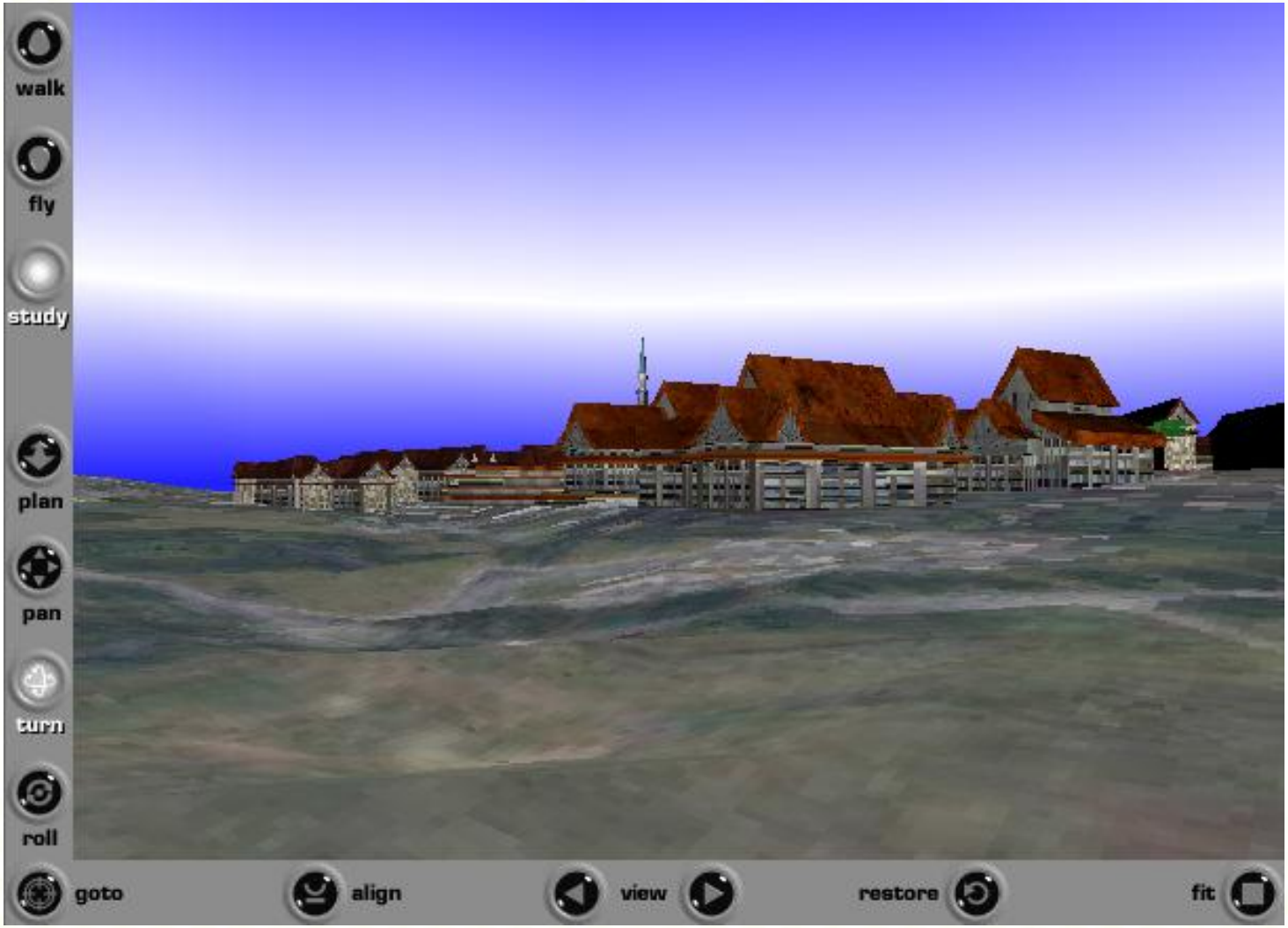

Scene Tree
Transform
Routing Map
Resources
File List
#VRML V2.0 utf8
Transform {
children Group{
children Shape {
appearance Appearance {
texture ImageTexture {
url"try.jpg" }
textureTransform TextureTransform {
scale -1 1
}
}
geometry ElevationGrid {
xDimension 35
zDimension 28
xSpacing 30
zSpacing 30
height [5.9984,12.6641,15.6671,16.2163,15.5604,15.5530,13.
4.7981,13.1240,20.2496,25.6407,29.3803,30.9916,31.6834,31.1
4.7981,10.4988,19.9070,26.1419,29.5678,17.2276,30.5534,30.1
4.7981,10.4988,16.1995,24.8099,19.5134,28.5862,29.1134,27.3
4.7981,10.4988,16.1995,20.0000,23.2883,23.8642,23.5584,20.2
4.7986,10.4988,16.1995,20.0000,20.0000,20.0000,18.8606,8.74
4.7992,10.4988,16.1995,20.0000,20.0000,20.0000,19.6816,14.9
4.7997,10.4988,16.1995,20.1535,20.4354,20.1980,20.0317,20.1
4.8002,10.4988,16.1995,19.0965,21.4428,21.5600,21.6885,22.1
4.8007,10.4988,16.1995,19.1655,20.6400,16.8709,17.0984,23.1
4.8012,10.4991,16.1995,19.1655,13.7746,0.2833,12.8373,18.17
4.8017,10.4996,16.1995,19.1655,16.5334,4.5402,14.4839,24.63
4.8022,10.5001,16.1995,19.1655,16.6619,9.5173,16.8637,24.08
4.8027,10.5006,16.1995,19.1655,16.6619,13.9538,18.9642,23.7
4.8032,10.5011,16.1995,19.1655,16.6619,17.4094,20.7055,23.9
4.8037,10.5016,16.1996,19.1655,16.6619,19.7494,21.6608,23.9
4.8042,10.5022,16.2001,19.1655,19.3097,22.4179,22.5000,23.1
4.8047,10.5027,16.2006,19.1655,21.3891,21.4609,21.7515,22.2
4.8052,10.5032,16.2011,20.4025,20.1923,20.1644,20.6212,20.8
4.8057,10.5037,16.2016,19.1808,18.1624,15.5684,16.7309,17.5
4.8062,10.5042,16.2021,18.0036,17.5000,17.5000,17.5000,17.5
4.8067,15.6671,16.2163,15.5604,15.5530,13.5766,12.5700,11.5
4.8072,10.5052,14.2116,13.7997,13.7852,13.7708,13.7563,13.7
4.8077,10.5057,11.9237,11.7283,11.7138,11.6993,11.6848,11.6
4.8082,9.6560,9.6713,9.6568,9.6423,9.6278,9.7709,10.4561,11
4.8087,7.7318,7.5998,7.5853,7.5708,8.0017,8.6870,9.1809,9.5
4.8092,5.5428,5.5283,5.5472,6.2325,6.7439,7.1550,7.5661,7.9
3.5399,3.4713,3.7780,4.3068,4.7179,5.1290,5.5401,5.9513,6.3
1.4143,1.8697,2.2808,2.6920,3.1031,3.5142,3.9253,4.3364,4.7
]
}
}
}
}
DTM_UTM.wrl
Ready
Ln 1 Col 1
9.01KB
SYN SEM NBR EXT OVR GZ READ WEB

Combining several objects to one world

Panorama scene using texture image



Panorama scene by specifying sky color



FKSG in VRML (Textured building model)

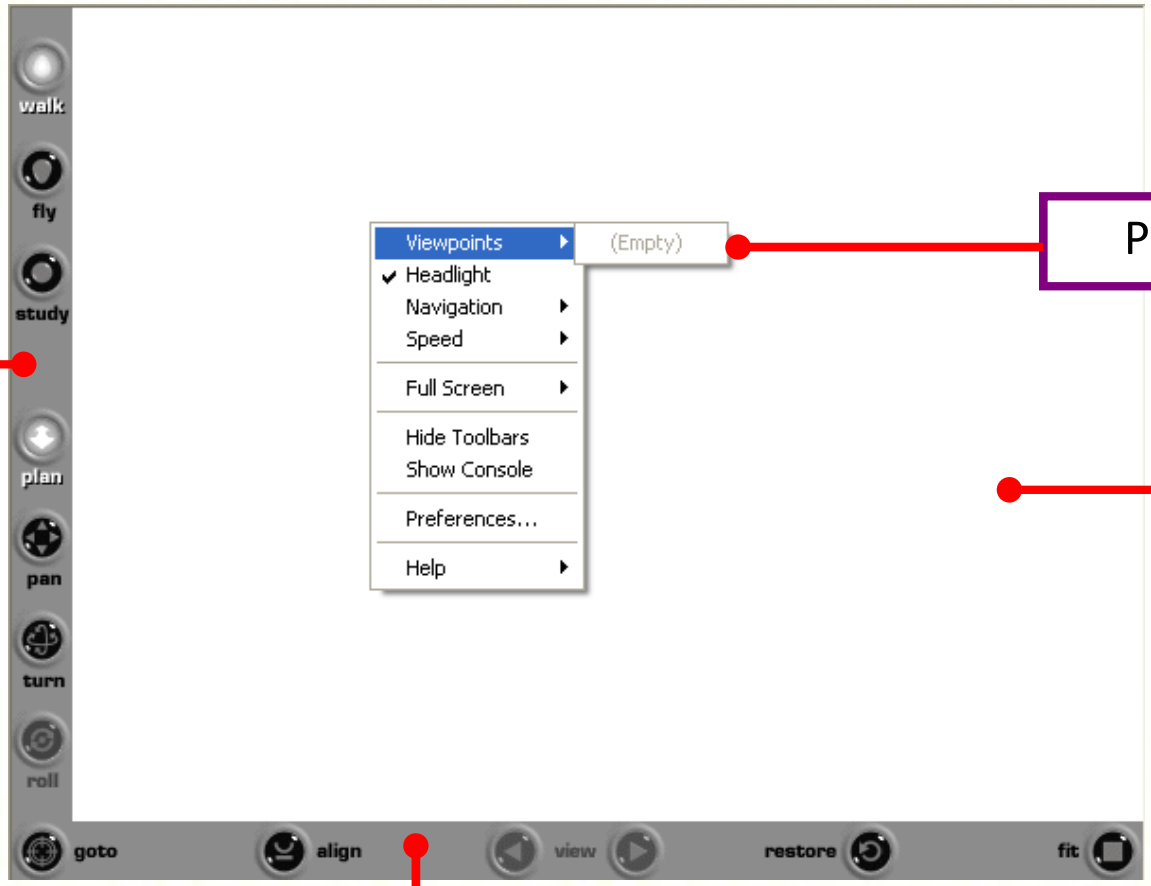
Block C02 (FKSG) in VRML

VRML requirements

- **Text editor**
  - Emacs (linux, windows, unix) : http://www.xemacs.org/Download/index.html
  - HTML-Kit, Notepad, WordPad (windows)
- **Web browser**
  - Internet Explorer
  - Mozilla
  - Opera
  - Netscape
- **Plug-in**
  - Cortona (the one we use) : http://www.parallelgraphics.com/products/cortona/
  - CosmoPlayer

VRML requirements

- The Cortona VRML Client



Vertical toolbar

Specify navigation type in a world

Pop-up menu

3D Window

Horizontal toolbar

Predefined action to change your position in the world

Adding VRML to your website

- Put VRML code in a separate file.
- Filename should end with ".vrml" or ".wrl"
- Use the "embed" command in HTML to add the file, i.e.:

< embed src="test.vrml" width="600" height="200" >

VRML file structure : The Basics

- File header
- Comments
- Brackets
- Shapes
- Colors
- 3D coordinates and transforms
- Lights

VRML file structure : File Header

- IMPORTANT: Your VRML file must always start with the:

#VRML V2.0 utf8

  - Tells that the file is a VRML file.
  - Tells the file is compatible with version 2.0 of the VRML specification.
  - Tells that the file is encoded with the international standard utf8.
  - "*The VRML header not found or unsupported encoding type*".

VRML file structure : Brackets

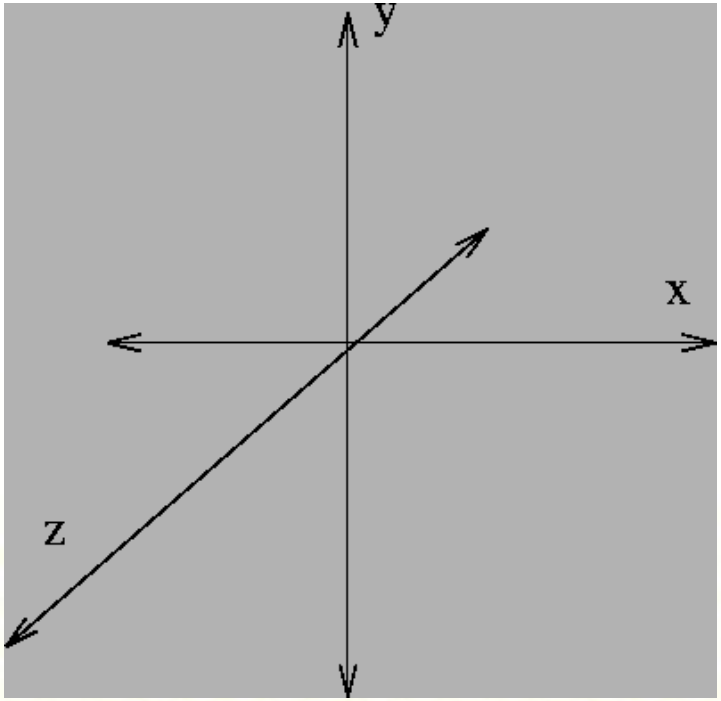
- Brackets “{” and “}” are used in order to group things together logically.
- Using indentation with brackets is VERY important and makes code much easier to read.
- Example:

```
Shape{geometry Box { size 2.0 2.0 2.0}appearance
  Appearance{material Material { diffuseColor 0 1 0 }}}
```

```
Shape
{
  geometry Box { size 2.0 2.0 2.0 }
    appearance Appearance
    {
      material Material { diffuseColor 0 1 0 }
    }
}
```

VRML file structure : 3D Coordinates

- Points are represented by 3 variables corresponding to the x, y and z axis.
- The point (0,0,0) is referred to as “the origin”.
- Other points are referenced in relation to the origin, e.g. (1,1,1), (-1,0,0).



VRML file structure : Placing objects in VRML

- Use the “Transform” function along with 3D coordinates. For example,

```
Transform
 {
   translation 1 1 1
   children
   [
     #objects (shapes etc.) added here will be at the
     point 1 1 1
   ]
 }
#objects added here will be at the origin
```

- Note: Transforms can also change scale and cause rotations.

X3D

Introduction to X3D

- Extensible 3D.
- Open standards file format to represent and communicate 3D scenes and objects using XML.
- X3D is the successor to the Virtual Reality Modeling Language (VRML). It improves upon VRML with new features, advanced APIs, additional data encoding formats, stricter conformance, and a componentized architecture using profiles that allows for a modular approach to supporting the standard and permits backward compatibility with legacy VRML data.
- Main difference between VRML and X3D is the definition of the event model.

Introduction to X3D

- The VRML specification left many decisions up to the browser implementer and therefore a lot of content was incompatible.
- The major issue was dealing with the way scene graph changes were propagated when the user code wrote to the field. In the Java language, the values would need to be delivered immediately.
- X3D provides both the XML-encoding and the Scene Authoring Interface (SAI) to enable both web and non-web applications to incorporate real-time 3D data, presentations and controls into non-3D content.
- Additional features (MPEG-4 multimedia standard support, XML and SVG compatible).

Introduction to X3D

X3D Supports:

- 3D graphics and programmable shaders – Polygonal geometry, parametric geometry, hierarchical transformations, lighting, materials, multi-pass/multi-stage texture mapping, pixel and vertex shaders, hardware acceleration,
- 2D graphics – Spatialized text; 2D vector graphics; 2D/3D compositing,
- CAD data – Translation of CAD data to an open format for publishing and interactive media,
- Animation – Timers and interpolators to drive continuous animations; humanoid animation and morphing,

Introduction to X3D

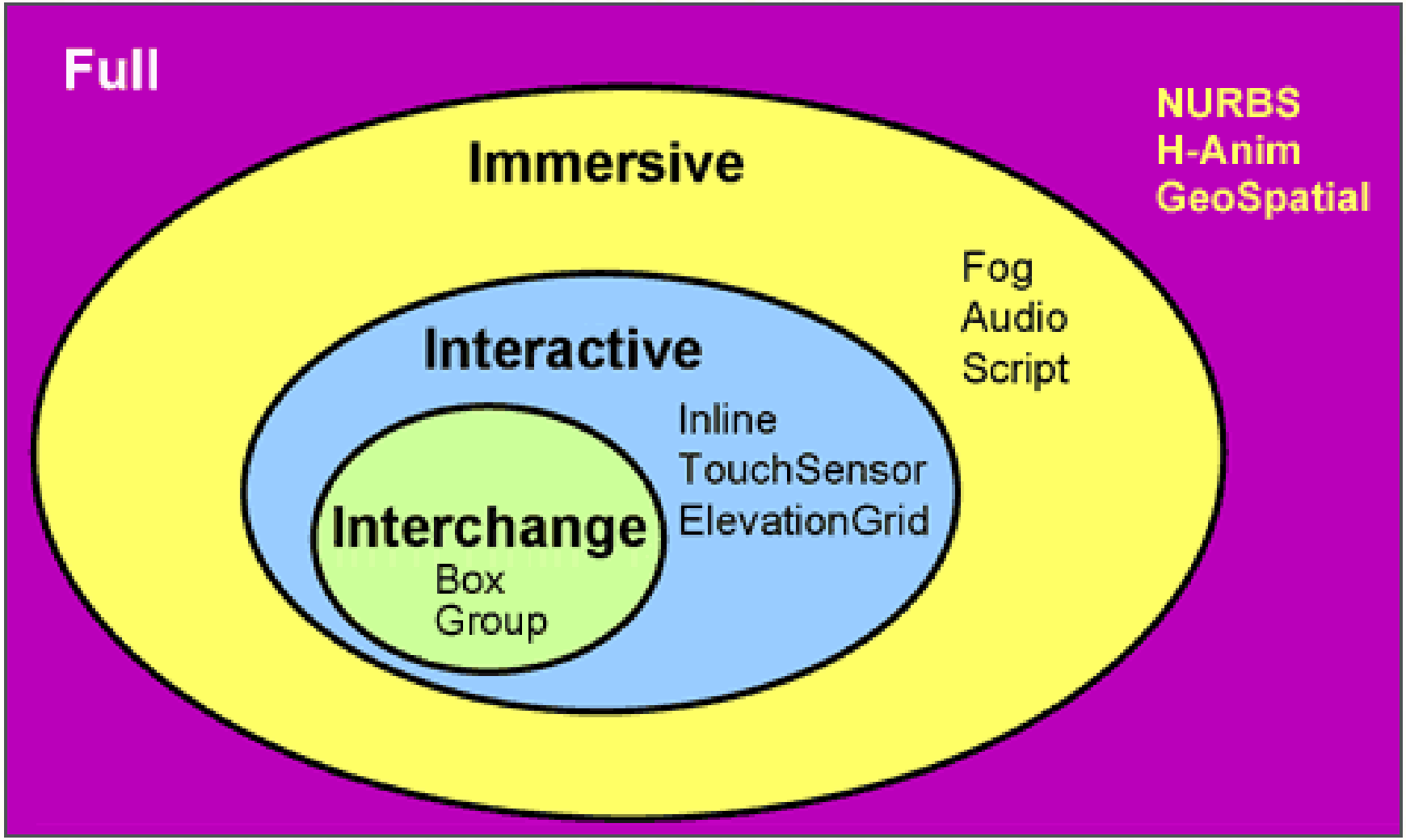
X3D Supports:

- Spatialized audio and video – Audio-visual sources mapped onto geometry in the scene,
- User interaction – Mouse-based picking and dragging; keyboard input,
- Navigation – Cameras; user movement within the 3D scene; collision, proximity and visibility detection,
- User-defined objects – Ability to extend built-in browser functionality by creating user-defined data types,
- Networking – Ability to hyperlinking of objects to other scenes or assets located on the World Wide Web.

Introduction to X3D

- The modular architecture of X3D allows for layered "profiles" that can provide:
  - increased functionality for immersive environments and enhanced interactivity or,
  - focused data interchange formats .
- Three profiles in X3D architecture:
  - Interchange,
  - Interactive,
  - Immersive.

Introduction to X3D



Introduction to X3D

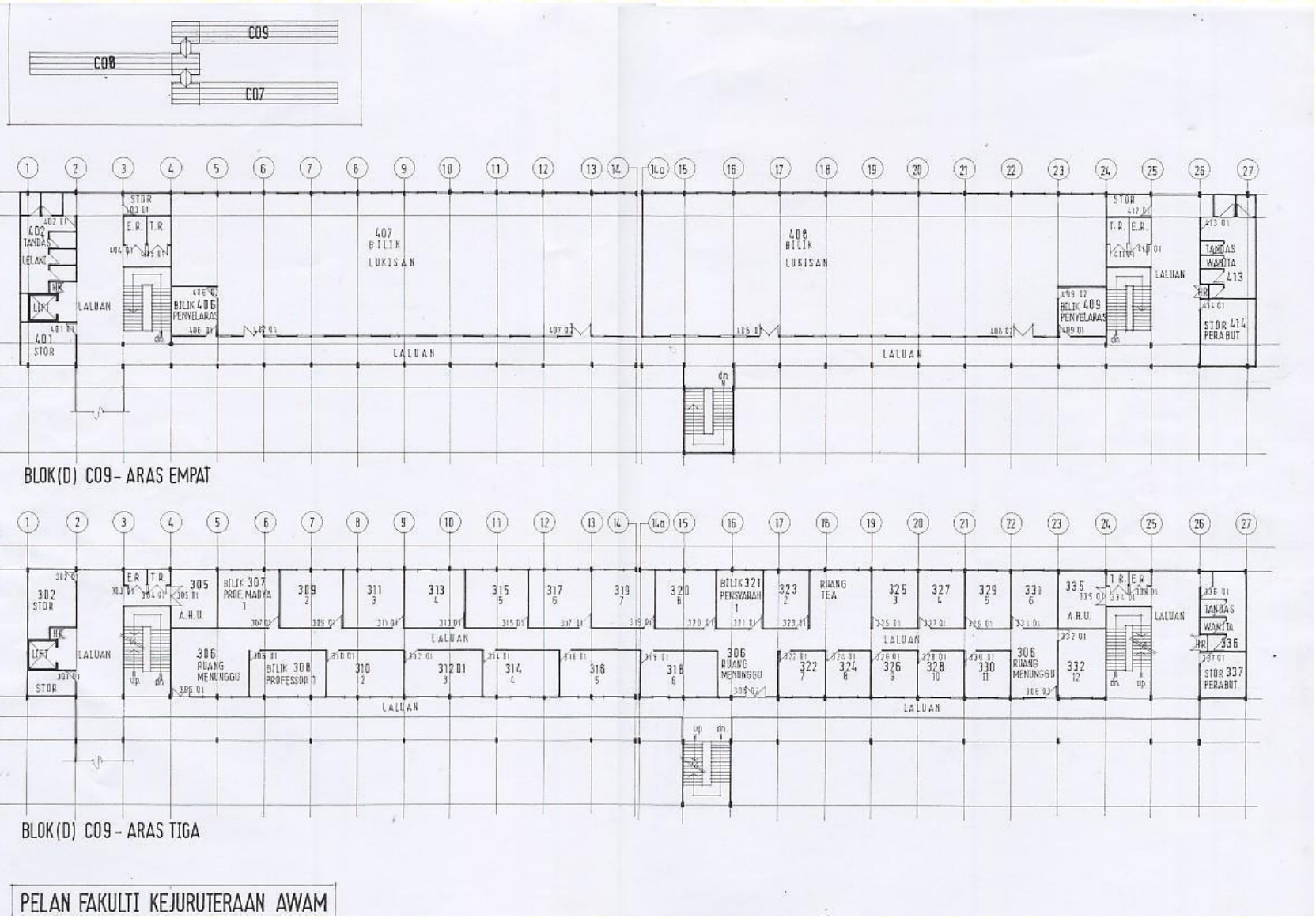
- **Why use X3D over VRML97?**
  - X3D is a considerably more mature refined standard than VRML so authors can achieve the behaviors they expect.
  - VRML compatible.
  - XML encoding to integrate smoothly with other applications.
  - X3D scenes and environments operate predictably between different players - A major problem with VRML is that it is difficult to develop VRML environments that play on all conformant browsers/players.
  - X3D is componentized - X3D is componentized which allows for the specification of profiles tailored to a particular large market segment (e.g., CAD, Medical, Visualization).

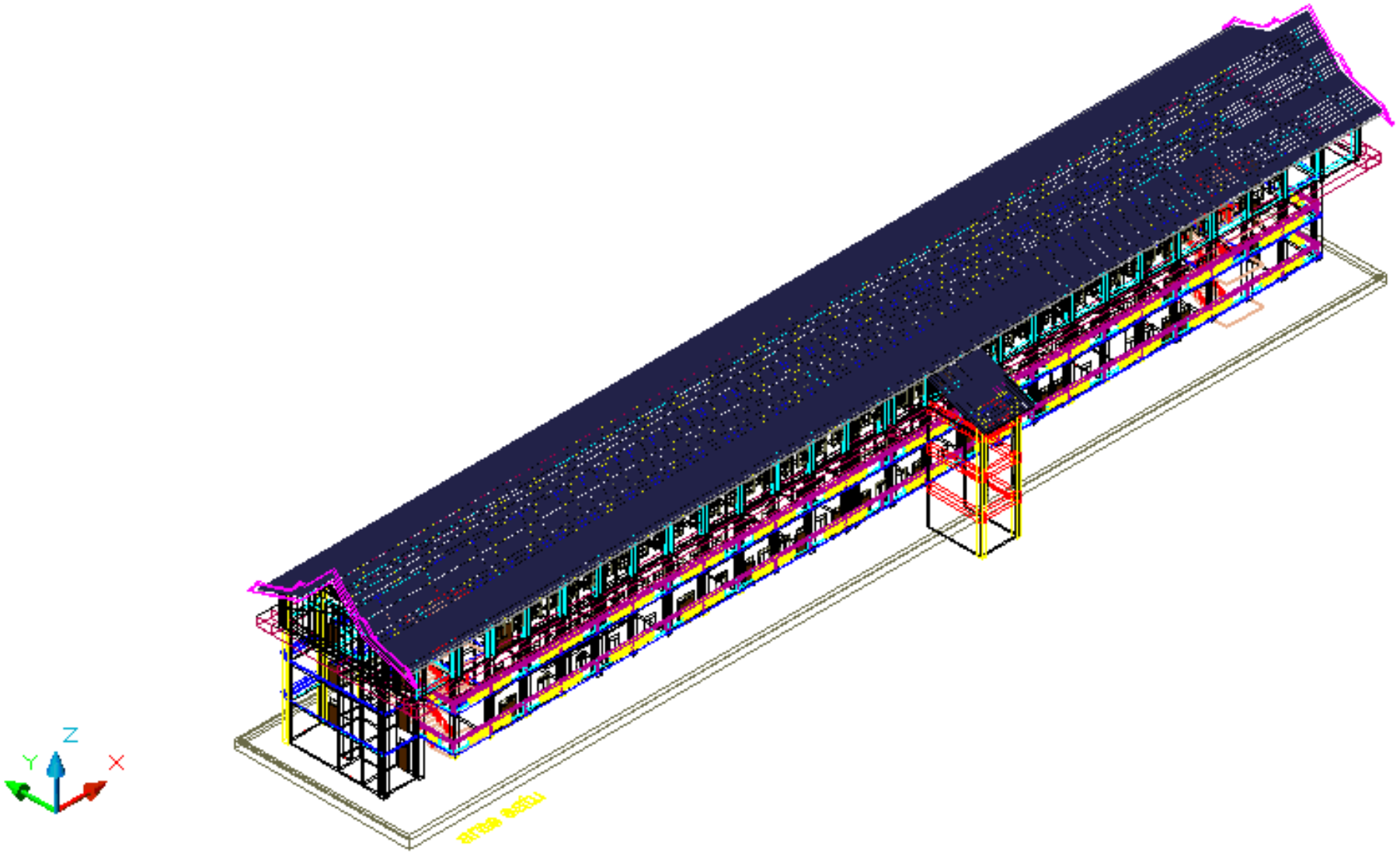
Introduction to X3D

- **Why use X3D over VRML97?**
  - X3D is more feature rich.
  - X3D is continually being enhanced and updated - X3D is growing in functionality. The Proposed Draft Amendment 1 specification that adds such things as 3D textures and shading languages is available.
  - The structure of X3D makes it much easier to update on a regular basis. It is also easier to add new features that adapt to the changing graphics and commercial markets.
  - X3D binary format offers encryption (i.e. security) and compression (i.e. speed).

X3D requirements

- **Text editor**
  - HTML-Kit, Notepad, WordPad (windows)
- **Web browser**
  - Internet Explorer
  - Mozilla
  - Opera
  - Netscape
- **Plug-in**
  - Octaga Player. Full free X3D player for Window and Linux.
  - BS Contact VRML X3D VRML/X3D.
  - Flux Web3D Engine from Vivaty. Free player X3D player for Windows (IE/Firefox). The flux player is open source and available on SourceForge.


C09
C08
C07
STOR
E.R.
T.R.
407 BILIK LUKISAN
408 BILIK LUKISAN
BILIK 406 PENYELARAS
BILIK 409 PENYELARAS
TANDAS WANITA 413
STOR 414 PERABUT
402 TANDAS LELAKI
401 STOR
LIFT
LALUAN
BLOK(D) C09 - ARAS EMPAT
302 STOR
305
BILIK 307 PROF MADYA 1
309 2
311 3
313 4
315 5
317 6
319 7
320 8
BILIK 321 PENSYARAH 1
323 2
RUANG TEA
325 3
327 4
329 5
331 6
335
A.H.U.
306 RUANG MENUNGGU
BILIK 308 PROFESSOR 1
310 2
312 01 3
314 4
316 5
318 6
306 RUANG MENUNGGU
322 7
324 8
326 9
328 10
330 11
306 RUANG MENUNGGU
332 12
TANDAS WANITA 336
STOR 337 PERABUT
BLOK(D) C09 - ARAS TIGA
PELAN FAKULTI KEJURUTERAAN AWAM



```
<Transform DEF="C09WALL2"
        translation="-2.363 0 -4.619"
        rotation="0 -1 0 -0.7854">

    <Transform translation ="0 5.566 0">
        <Shape>
            <Box size="2 11.13 9.76"/>
            <Appearance>
                <Material diffuseColor="0.5 0.5 0.5"
                          ambientIntensity="0.2"
                          shininess="0.2"
                          specularColor="0 0 0"
                          transparency="0"/>
                <ImageTexture url = "dindingtepi.jpg"/>
                        </Appearance>
                    </Shape>
            </Transform>
        </Transform>
```

Transform node: It allows to position a group of objects in 3D space by translating, rotating and scale the size of objects.

Box node are used to make a wall of the building.

Image texture to display the real texture of the building.

```
<Transform DEF="C09POLE4"
                        translation="63.3 11.13 -70.28"
                        rotation="0 -1 0 -0.7854">

        <Shape>
          <IndexedFaceSet DEF="C09POLE4"
                          ccw="true"
                          solid="true"
                          coordIndex="
                    0, 2, 3, -1, 3, 1, 0, -1, 4, 5, 7, -1, 7, 6, 4, -1, 0, 1, 5, -1,
                    5, 4, 0, -1, 1, 3, 7, -1, 7, 5, 1, -1, 3, 2, 6, -1,
                    6, 7, 3, -1, 2, 0, 4, -1, 4, 6, 2, -1"
                          texCoordIndex="
                    9, 11, 10, -1, 10, 8, 9, -1, 8, 9, 11, -1, 11, 10, 8, -1,
                    4, 5, 7, -1, 7, 6, 4, -1, 0, 1, 3, -1, 3, 2, 0, -1,
                    4, 5, 7, -1, 7, 6, 4, -1, 0, 1, 3, -1, 3, 2, 0, -1">



             <Coordinate DEF="C09POLE4-COORD"
                         point="
                    -1.25 0 5.88, 1.25 0 5.88, -1.25 0 -5.88, 1.25 0 -5.88,
                    -1.25 0.762 5.88, 1.25 0.762 5.88, -1.25 0.762 -5.88,
                    1.25 0.762 -5.88"
                         containerField="coord"/>

              <TextureCoordinate DEF="C09POLE4-TEXCOORD"
                         point="0 0, 1 0, 0 1, 1 1, 0 0, 1 0, 0 1, 1 1, 0 0, 1 0, 0 1,1 1"/>

                                    </IndexedFaceSet>

                              <Appearance>
                              <Material diffuseColor="0.5804 0.4118 0.1961"
                                        ambientIntensity="0.2"
                                        shininess="0.2"
                                        specularColor="0 0 0"
                                        transparency="0"/>
                              <ImageTexture url = ""/>
                                            </Appearance>

                               </Shape>
                    </Transform>
```

IndexedFaceSet : Represents a 3D shape formed by constructing faces (polygons) from vertices listed in the *coord* field.

Generate DTM using X3D

- Digital terrain model (DTM) that defines the surface of the study area, could be obtained from the 1:10,000 scale aerial photographs.
- Using the Leica-Helava system, the contour lines were digitized manually with the setting of a 2.5 meters interval.
- However, they are not suitable for computing slopes.
- The TIN model was then converted to Grid.
- The DTM grid interval for X (east) and Y (north) was set to 30 meters.
- The DTM grid was then converted to ASCII file format, which contains the values of the grid points.

Mosaic derived from aerial photograph

```
 0 52980.0000 7920.0000 5.9984
 1 53010.0000 7920.0000 4.7981
 2 53040.0000 7920.0000 4.7981
 3 53070.0000 7920.0000 4.7981
 4 53100.0000 7920.0000 4.7981
 5 53130.0000 7920.0000 4.7986
 6 53160.0000 7920.0000 4.7992
 7 53190.0000 7920.0000 4.7997
 8 53220.0000 7920.0000 4.8002
 9 53250.0000 7920.0000 4.8007
10 53280.0000 7920.0000 4.8012
11 53310.0000 7920.0000 4.8017
12 53340.0000 7920.0000 4.8022
13 53370.0000 7920.0000 4.8027
14 53400.0000 7920.0000 4.8032
15 53430.0000 7920.0000 4.8037
16 53460.0000 7920.0000 4.8042
17 53490.0000 7920.0000 4.8047
18 53520.0000 7920.0000 4.8052
19 53550.0000 7920.0000 4.8057
20 53580.0000 7920.0000 4.8062
21 53610.0000 7920.0000 4.8067
22 53640.0000 7920.0000 4.8072
23 53670.0000 7920.0000 4.8077
24 53700.0000 7920.0000 4.8082
25 53730.0000 7920.0000 4.8087
26 53760.0000 7920.0000 4.8092
27 53790.0000 7920.0000 3.5399
28 53820.0000 7920.0000 1.4143
```

xDimension=28
zDimension=35

Scene Development

The buildings and terrain model created using X3D encoding are all in single X3D file. So, must combine the object into one scene.

```
<Anchor DEF="DTM"
                description="DTM">
        <Transform translation="7920 1.4143 52980">
        <Inline url="DTM.x3d"/>

                </Transform>
        </Anchor>

<Anchor DEF="C09"
        description=" C09 Building">
<Transform translation="8410.1090 16.9910 53707.2390" rotation="0 1 0 -0.52">
<Inline url="c09.x3d"/>

        </Transform>
        </Anchor>
```

Anchor node: Show buildings' name when user moves the cursor over an object (like Tooltips).

Transform node: Positioning X3D models in 3D space according to their spatial coordinates by translating (moving) or rotating it.

Inline node: A method of adding a copy of the original file into scene - allows reference external sources by specifying its URLs.

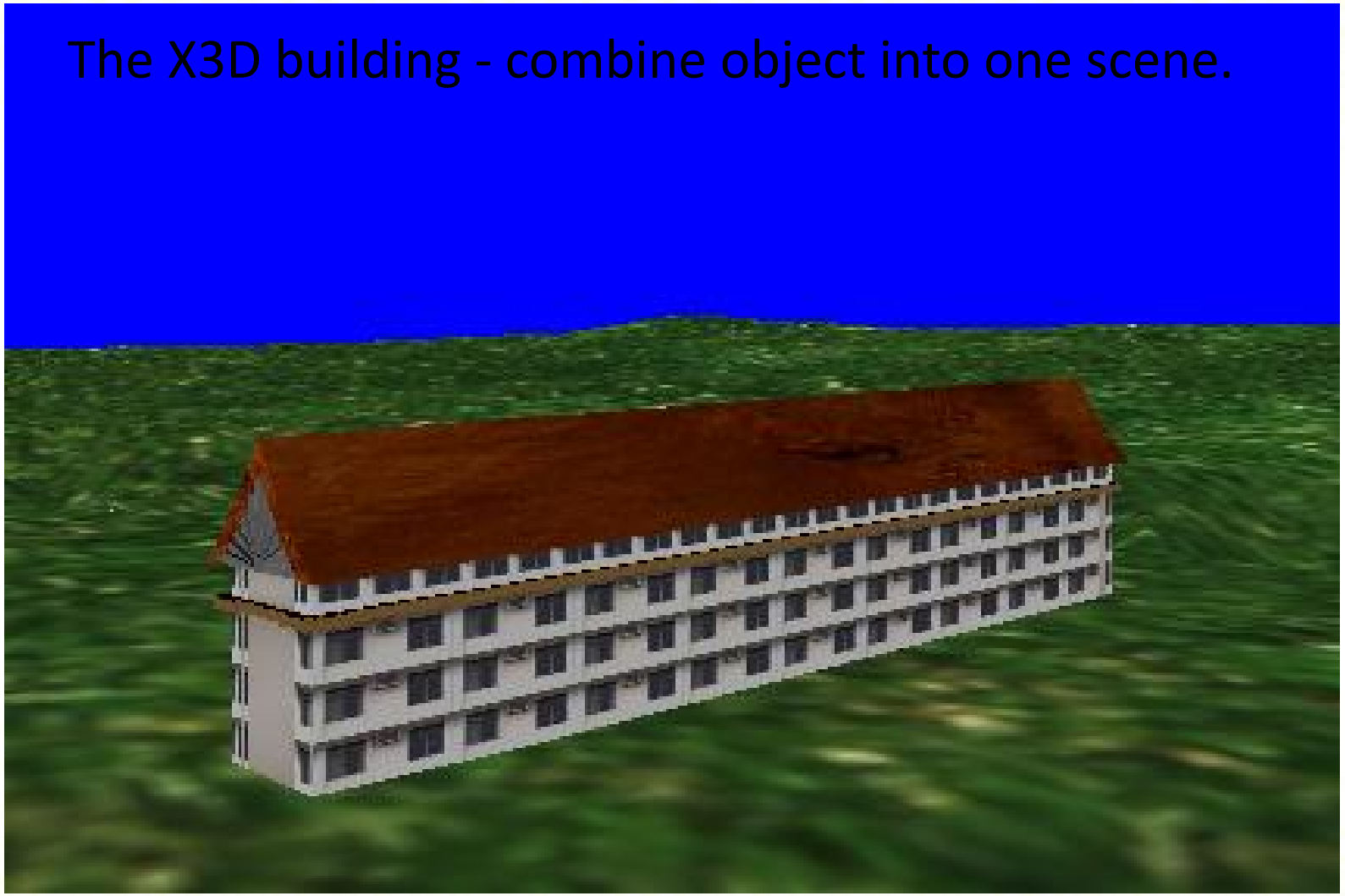
Scene Development

Scene Development

DTM with the real texture.

Scene Development

The X3D building - combine object into one scene.



GML

Data Exchange Formats

Industry defined formats

e.g.
- Shape, E00
- DXF
- MIF
- GeoMedia
- etc.

Standardization organizations

e.g.
- GDF
- ISO/IEC 8211:1994

National defined formats

e.g.
- NDCDB (Malaysia)
- EDBS for ALK/ATKIS (Germany)
- DSFL (Denmark)
- Interlis (Switzerland)
- KF85 (Sweden)
- NTF (GB)
- TIGER/Line (U.S. Census Bureau)

The Internet (XML based formats)

- GML

XML vs. GML

- What is XML?
- Extensible Markup Language
- A meta language created by W3C
- In contrast to human languages, computer languages need a well defined grammar
- XML is used to define (markup) languages:
  - XHTML
  - GML
  - SVG

Characteristics of XML

- Text based
- Tags to separate the different parts
- Separation of content and display
- DTDs (Document Type Definition) in XML 1.0
  - A set of rules that define an XML markup language (i.e. how the tags are arranged)
- Schema in XML 2.0
  - A set of rules plus a set of primitive data types and possibility to create own data-types

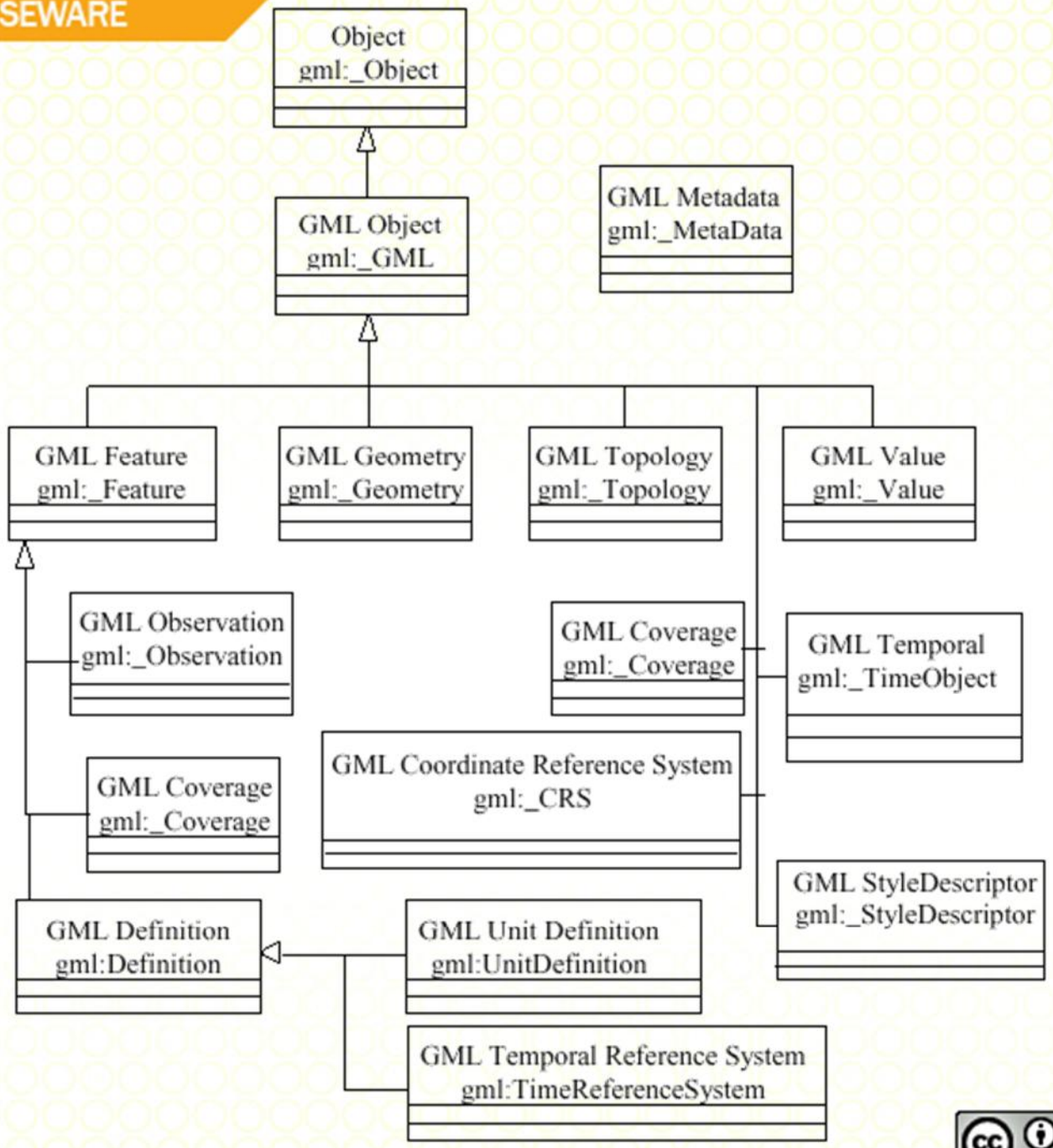
GML 3.0

- 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.

GML 3.0

- represent spatial and temporal reference systems, units of measure and standards information;
- use reference system, units and standards information in the representation of geospatial phenomena, observations, and values;
- represent default styles for feature and coverage visualization;
- conform with other standards, including
  - ISO DIS 19107 Geographic Information – Spatial Schema
  - ISO DIS 19108 Geographic Information – Temporal Schema
  - ISO DIS 19118 Geographic Information – Encoding
  - ISO DIS 19123 Geographic Information – Coverages

GML Data Model



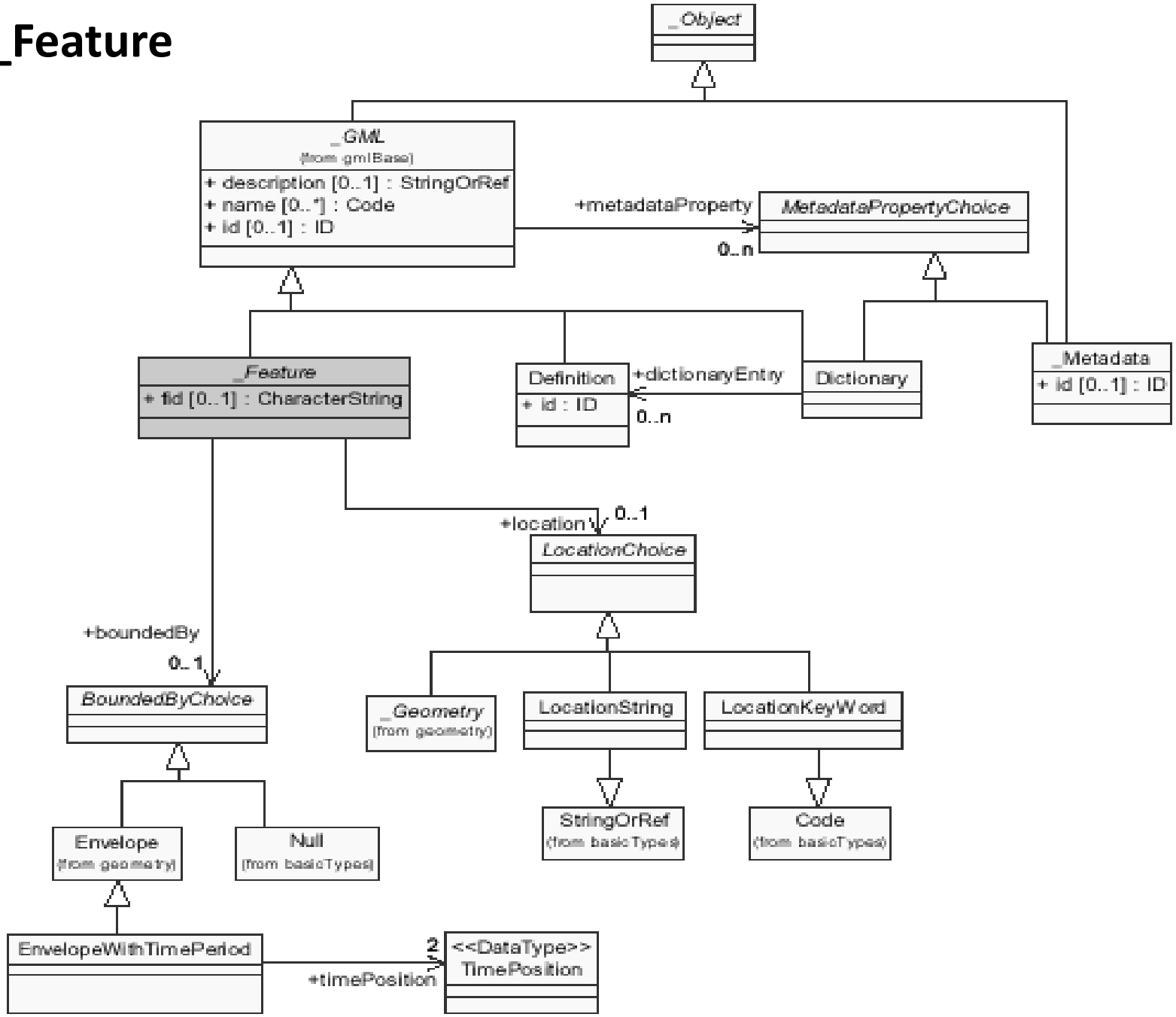
GML Schema

- Feature Schema (feature.xsd) with general rules
- Geometry Schema (geometry.xsd) with definitions of geometry components based on the Simple Feature Definition of OGC
- Xlink Schema (xlink.xsd) link attributes

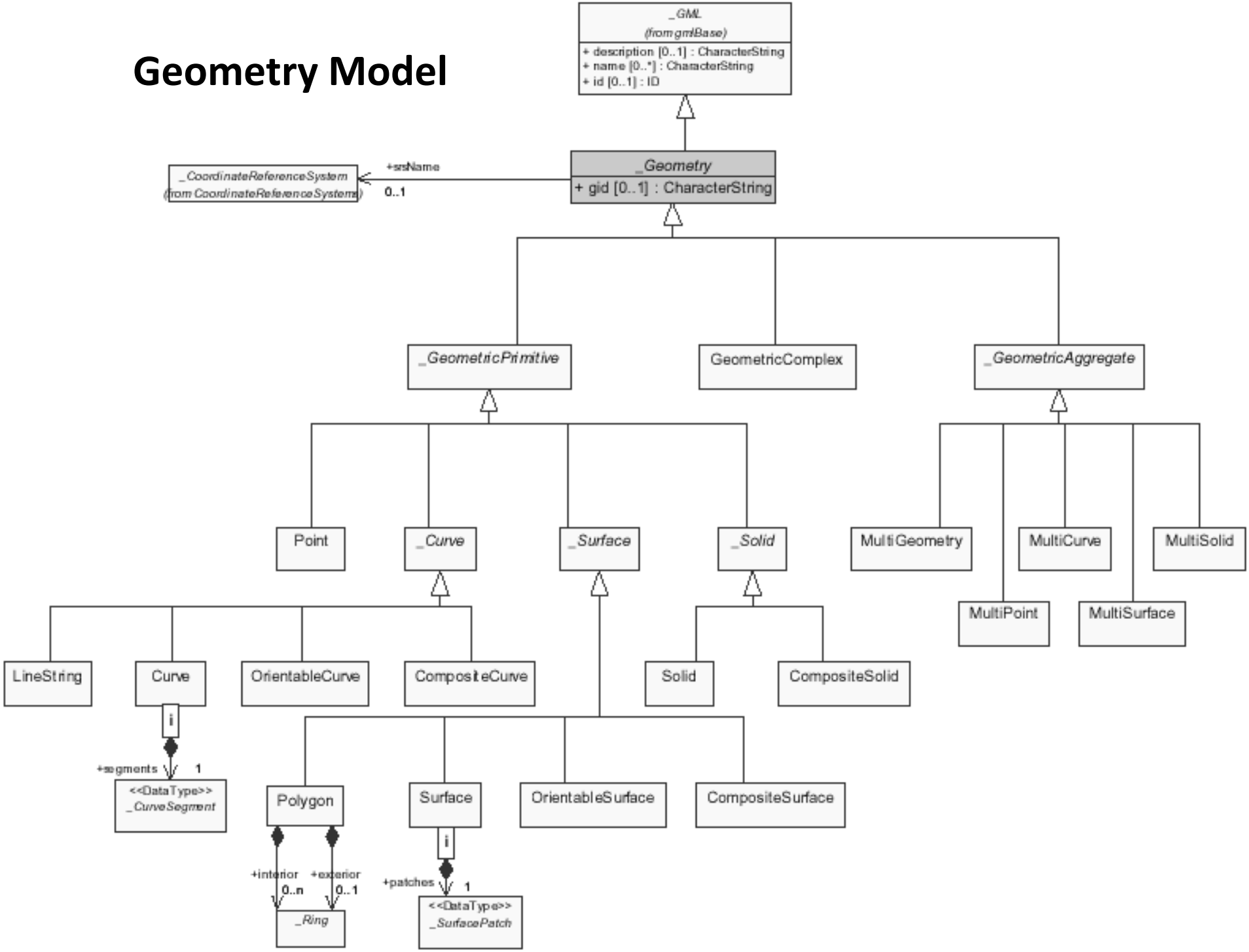
Example

```
...
<city>
  <name>Shah Alam</name>
  <population>43000</population>
  <gml:location>
    <gml:point
    srsName="http://www.opengis.net/gml/srs/epsg.xml#4326>
      <gml:coord> <X>346000.00</X><Y>565000.00</Y></gml:coord>
    </gml:point>
  </gml:location>
</city>
...
```

gml:_Feature


_Object
_GML
(from gmlBase)
+ description [0..1] : StringOrRef
+ name [0..*] : Code
+ id [0..1] : ID
+metadataProperty
0..n
MetadataPropertyChoice
_Feature
+ fid [0..1] : CharacterString
Definition
+ id : ID
+dictionaryEntry
0..n
Dictionary
_Metadata
+ id [0..1] : ID
+location
0..1
LocationChoice
+boundedBy
0..1
BoundedByChoice
_Geometry
(from geometry)
LocationString
LocationKeyWord
Envelope
(from geometry)
Null
(from basicTypes)
StringOrRef
(from basicTypes)
Code
(from basicTypes)
EnvelopeWithTimePeriod
+timePosition
2
<<DataType>>
TimePosition

Geometry Model



GML 3.0 Schemas

- gml.xsd
- gmlBase.xsd
- basicTypes.xsd
- dictionary.xsd
- units.xsd
- measures.xsd
- temporal.xsd
- geometryBasic0d1d.xsd
- valueObjects.xsd
- coverage.xsd
- defaultStyle.xsd
- geometryBasic2d.xsd
- geometryPrimitives.xsd
- geometryAggregates.xsd
- geometryComplexes.xsd
- grids.xsd
- topology.xsd
- direction.xsd
- feature.xsd
- dynamicFeature.xsd
- observation.xsd
- dataQuality.xsd
- referenceSystems.xsd
- datums.xsd coordinateSystems.xsd
- coordinateOperations.xsd
- coordinateReferenceSystems.xsd

KML

Introduction to KML

- KML is the Keyhole Markup Language.
- KML is an XML grammar used to encode and transport representations of geographical data for display in a geobrowser.
- KML uses a tag-based structure with nested elements and attributes.
- The basic building blocks of the language are called ***elements***, and a ***tag*** is the way an element is represented as KML code.

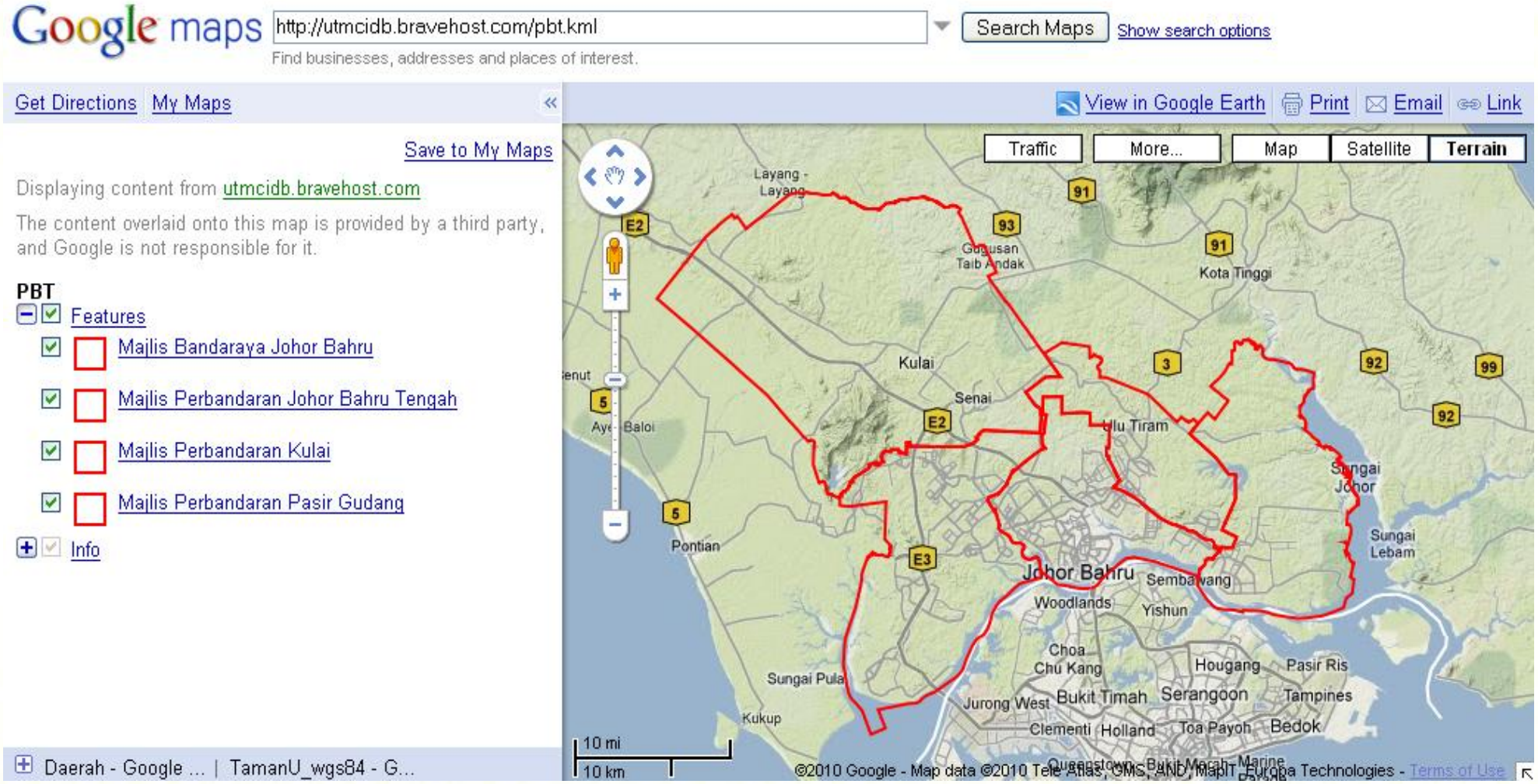
Introduction to KML

- KML is focused on visualization of geographic features on map. The XML language also includes controls of the user's navigation in the sense of where to go and where to look.
- KML was originally created as a file format for Keyhole's Earth Viewer, which later emerged as the Google Earth application allowing users to overlay their own content on top of the base maps and satellite imagery.
- In 2007, Google submitted KML to the OGC. Later in 2008 KML was adopted as an OpenGIS standard and the OGC has now the responsibility for maintaining and extending the standard.

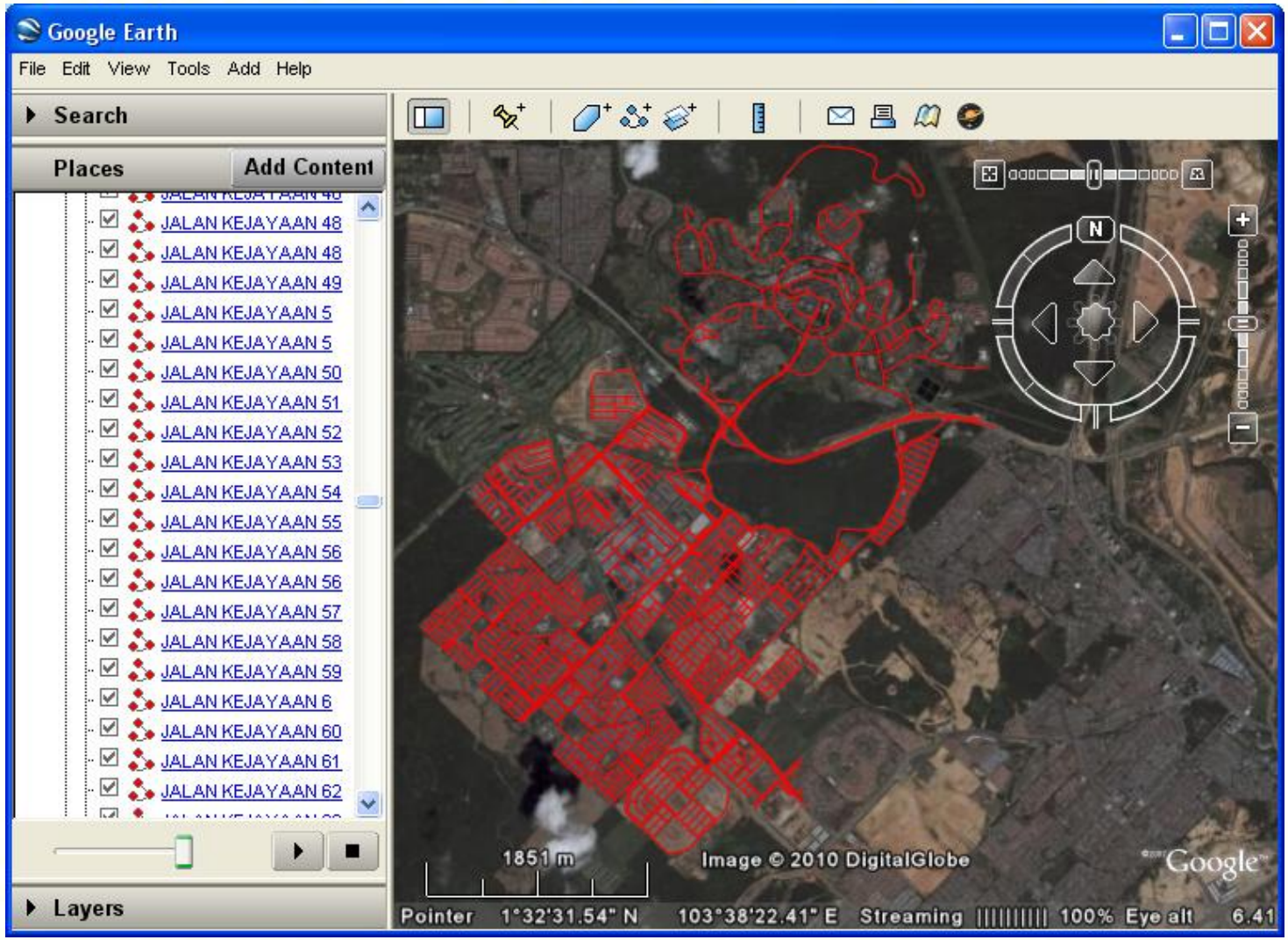
Introduction to KML

- KMZ is a compressed (.zip) KML plus any images.
  - KML documents and their related images and 3-D objects (if any) may be compressed using *ZIP encoding* into KMZ files.
  - This greatly reduces the file size and makes data transfer more efficient.
- You can view KMLs in Google Earth application or in Google Maps by simply pointing at the URL from the map search box.

Introduction to KML



Introduction to KML



Alternatives to KML

- **KML:**
  - It is popular and the structure is simple.
  - the standard is supported by many of the popular geobrowsers.
- **GML:**
  - Complex format.
  - XML grammar which helps in the storage, exchange and modelling of geographical information containing both spatial and non-spatial attributes.
  - The encoding is comprehensive in the way in which it can represent features with complex 3-D geometry, features with 2-D topology, dynamic features and coverages.
  - GML is not a visualisation language; it does not provide any information regarding how the data is to be displayed.

Alternatives to KML

- ***Styled Layer Descriptor*** **(SLD) and** ***Symbol Encoding*** **(SE):**
  - two related XML languages for styling information.
  - Widely used in OpenStreetMap application (www.openstreetmap.org).
  - SLD/SE is capable of describing the rendering of vector and raster data.
  - Only lines, polygons, points, text and raster images are integrated in the description language, which makes it impossible to visualise multiple data values, e.g. pie charts or bar charts.
  - Various proposals exist for an OGC Styled Layer Descriptor / Symbology Encoding extension for thematic cartography.

Alternatives to KML

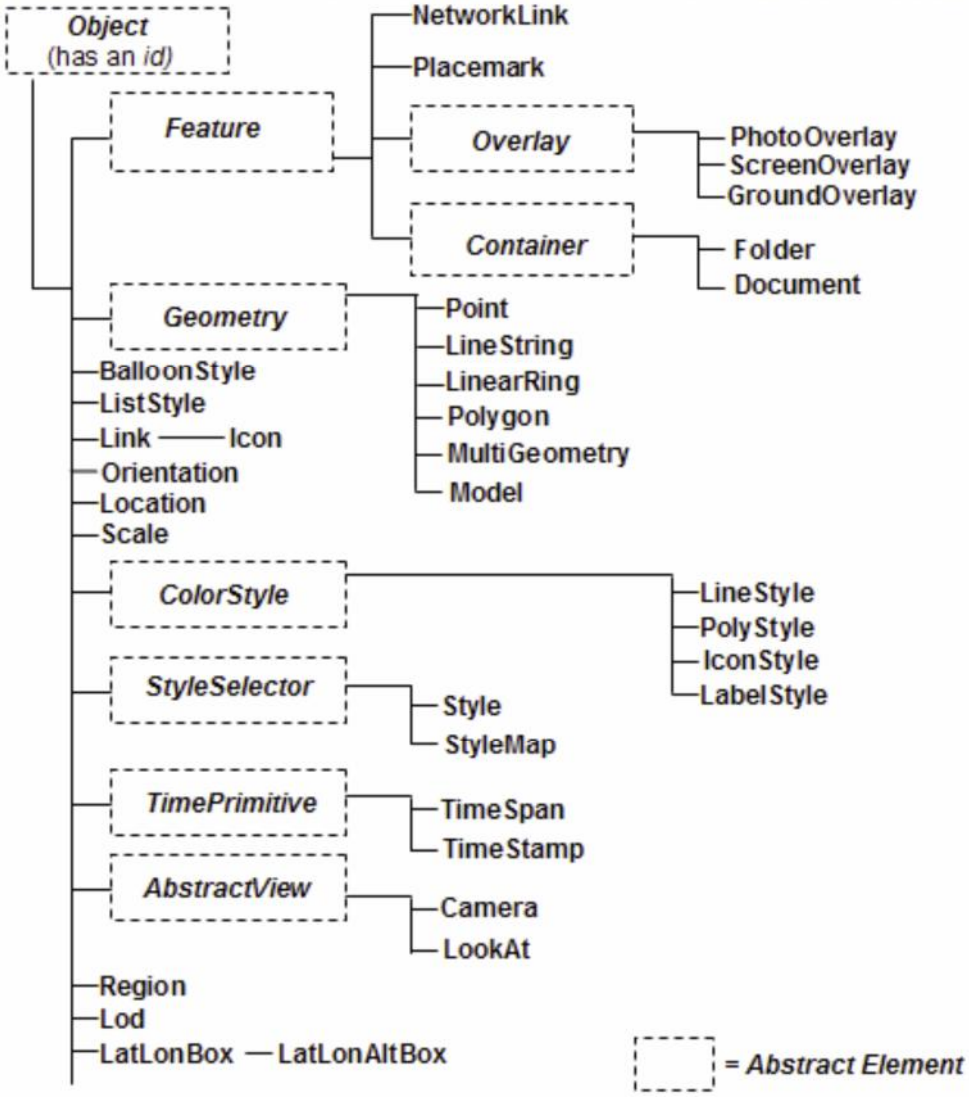
- ***Scalable Vector Graphics* (SVG):**
  - XML based web standard for 2-D vector graphics from W3C.
  - suitable format for GIS and mapping applications.
  - While GML provides a means of storing and transporting geographical features, SVG makes it possible to display these features as vector maps.
  - The graphics description capabilities of SVG are much stronger than those of KML, but SVG is only 2-D and does not incorporate concepts for navigation.

Alternatives to KML

As a summary:

- KML does a bit of everything. It defines geographic objects, their styling and their graphical representation.
- There are some overlaps between GML and KML in the way in which the basic geometrical objects are represented.
  - KML contains styles, but is not a styling language in the manner of SLD/SE.
- In SVG, the canvas is the 2-D surface of a computer screen, whereas KML provides the mechanisms for visualising geographical features on a map or a globe.
- GML and KML are also logical partners, like GML and SVG.
- SLD/SE provides styling rules to transform data encoded in GML into a target visualisation language (e.g. KML or SVG).

Specifications of KML



Hierarchy of KML elements

Specifications of KML

- A Placemark is one of the most commonly used features in Google Maps. It marks a position on the Earth's surface.

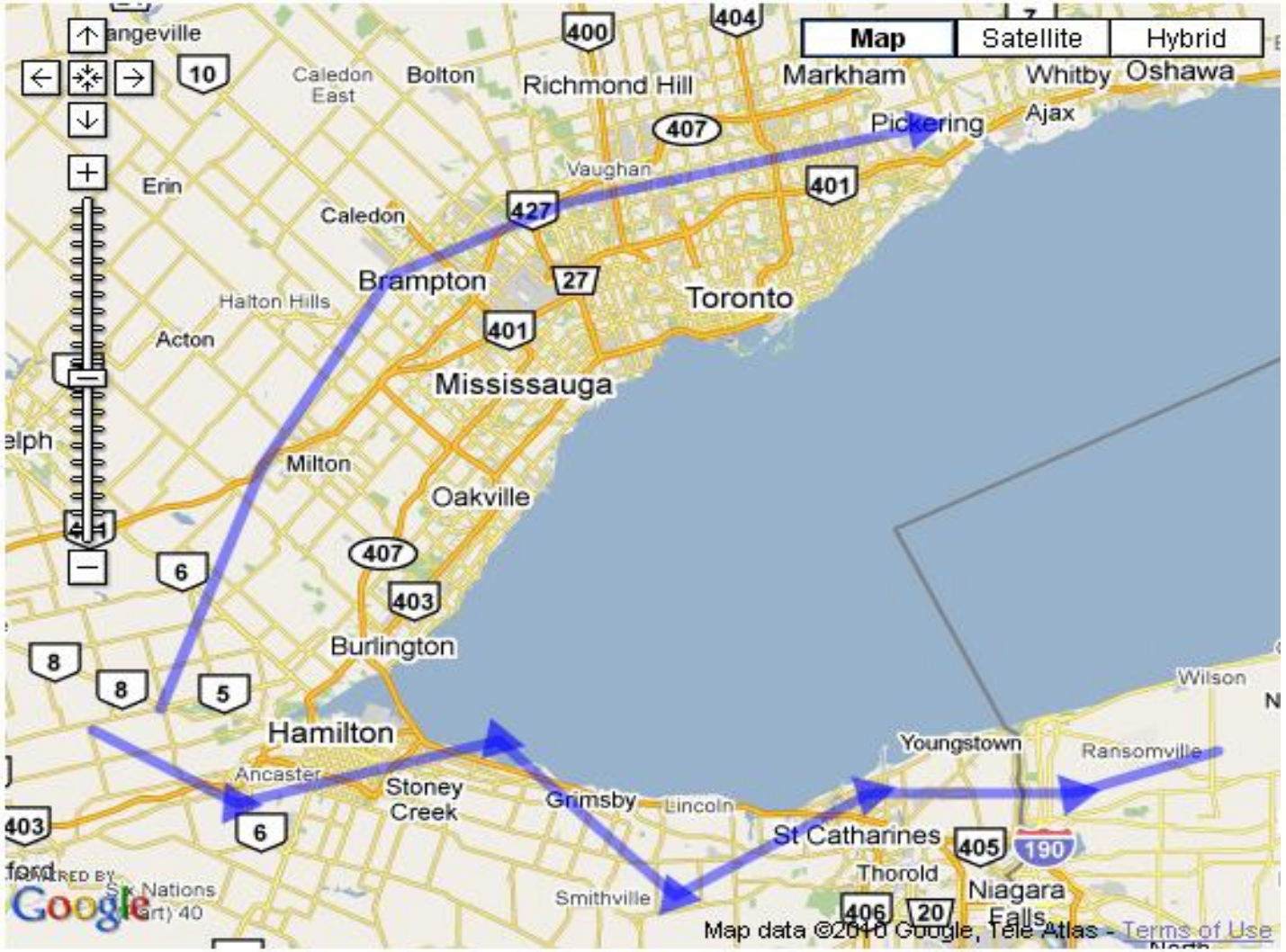
```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <Placemark>
    <name> 3.4 </name>
    <Point>
      <coordinates>-110.46,44.47,0</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

Specifications of KML

Specifications of KML

Polyline code.

```
var points = [ new GLatLng(43.28,-80.07),
               new GLatLng(43.51,-79.95),
               new GLatLng(43.69,-79.80),
               new GLatLng(43.76,-79.59),
               new GLatLng(43.83,-79.17),
               new GLatLng(43.26,-80.15),
               new GLatLng(43.19,-79.98),
               new GLatLng(43.25,-79.67),
               new GLatLng(43.10,-79.46),
               new GLatLng(43.20,-79.23),
               new GLatLng(43.20,-78.99),
               new GLatLng(43.24,-78.82)];

map.addOverlay(new GPolyline(points));
```


Map
Satellite
Hybrid
Caledon East
Bolton
Richmond Hill
Markham
Whitby
Oshawa
Pickering
Ajax
Vaughan
Erin
Caledon
Brampton
Halton Hills
Acton
Toronto
Mississauga
Milton
Oakville
Burlington
Hamilton
Ancaster
Stoney Creek
Grimsby
Lincoln
St Catharines
Thorold
Niagara Falls
Youngstown
Ransomville
Wilson
Smithville
Google
Map data ©2010 Google, Tele Atlas - Terms of Use

Specifications of KML

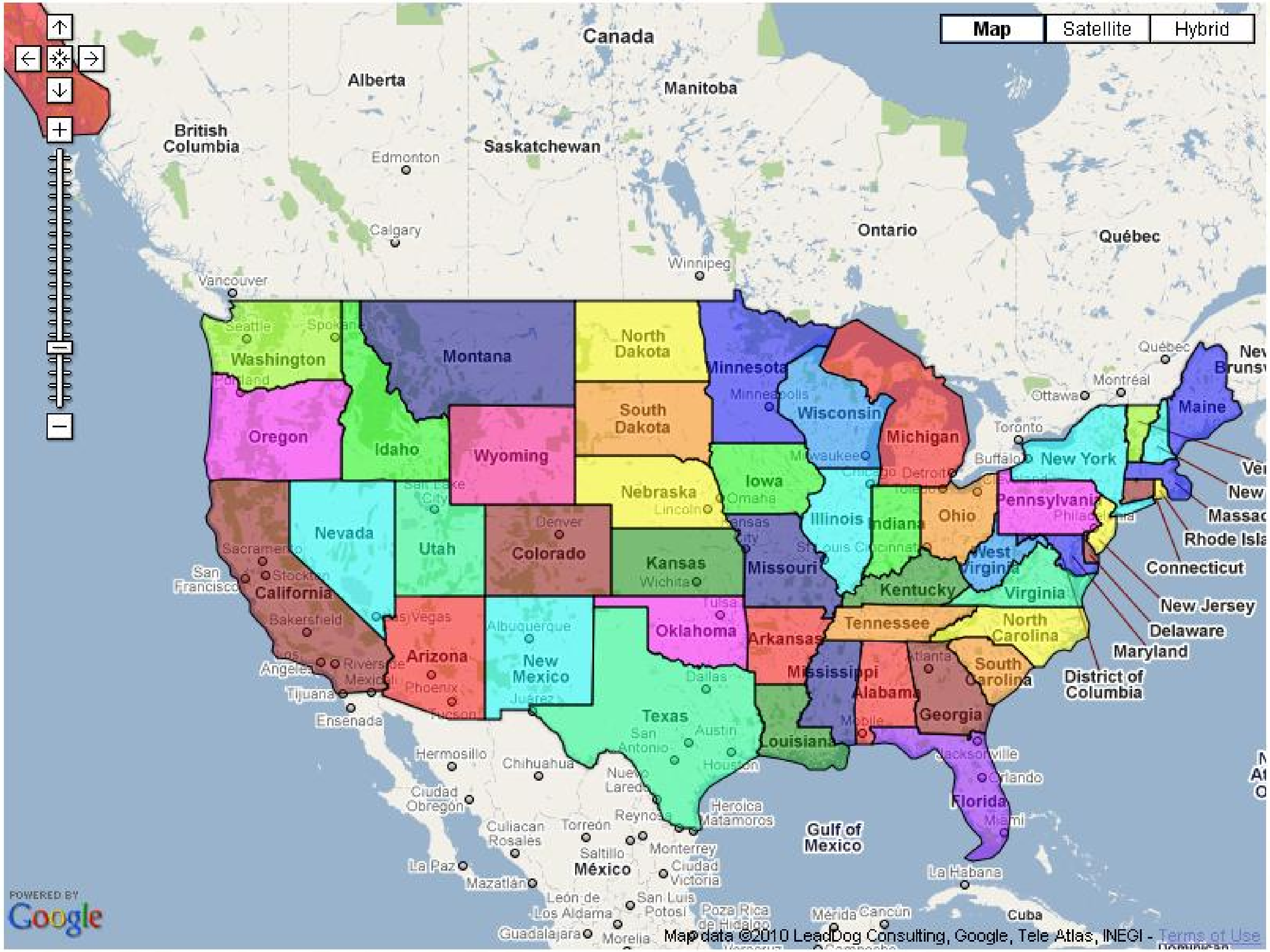
- A polygon feature is generated using a series of points data.

```
- <states>
  + <state name="Alaska" colour="#ff0000">
  + <state name="Alabama" colour="#ff0000">
  + <state name="Arkansas" colour="#ff0000">
  + <state name="Arizona" colour="#ff0000">
  + <state name="California" colour="#880000">
  - <state name="Colorado" colour="#880000">
      <point lat="37.0004" lng="-109.0448" />
      <point lat="36.9949" lng="-102.0424" />
      <point lat="41.0006" lng="-102.0534" />
      <point lat="40.9996" lng="-109.0489" />
      <point lat="37.0004" lng="-109.0448" />
    </state>
  + <state name="Connecticut" colour="#880000">
  + <state name="Delaware" colour="#880000">
  + <state name="Florida" colour="#8800ff">
  + <state name="Georgia" colour="#880000">
  + <state name="Hawaii" colour="#00ff00">
  + <state name="Iowa" colour="#00ff00">
```

Specifications of KML

Polygon code.

```
var states = xmlDoc.documentElement.getElementsByTagName("state");

var a = 0; a < states.length; a++) {
var label = states[a].getAttribute("name");
var colour = states[a].getAttribute("colour");
var points = states[a].getElementsByTagName("point");
var pts = [];

for (var i = 0; i < points.length; i++)
{
  pts[i] = new GLatLng(parseFloat(points[i].getAttribute("lat")),
  parseFloat(points[i].getAttribute("lng")));
}

var poly = new
GPolygon(pts,"#000000",1,1,colour,0.5,{clickable:false});
polys.push(poly);
labels.push(label);
map.addOverlay(poly);
```

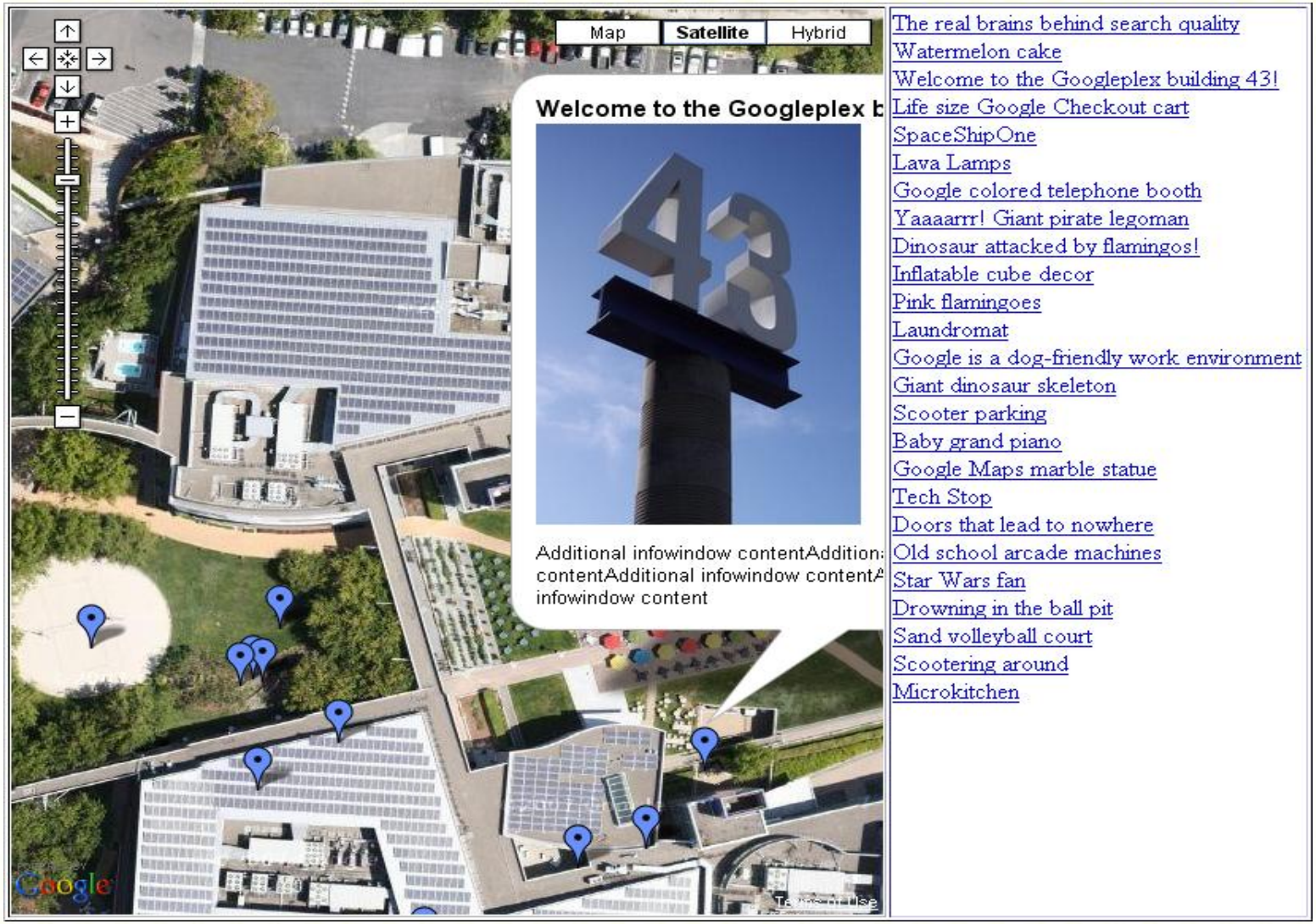

Map
Satellite
Hybrid
Canada
Alberta
Manitoba
British Columbia
Saskatchewan
Edmonton
Calgary
Ontario
Québec
Winnipeg
Vancouver
Washington
Montana
North Dakota
Minnesota
Oregon
Idaho
Wyoming
South Dakota
Wisconsin
Michigan
Maine
New York
Iowa
Nebraska
Illinois
Indiana
Ohio
Pennsylvania
Nevada
Utah
Colorado
Kansas
Missouri
West Virginia
Kentucky
Virginia
California
Arizona
New Mexico
Oklahoma
Arkansas
Tennessee
North Carolina
South Carolina
Mississippi
Alabama
Georgia
Texas
Louisiana
Florida
Connecticut
New Jersey
Delaware
Maryland
District of Columbia
Gulf of Mexico
Cuba
México
Map data ©2010 LeadDog Consulting, Google, Tele Atlas, INEGI - Terms of Use

KML Applications

As **GIS presentation**:

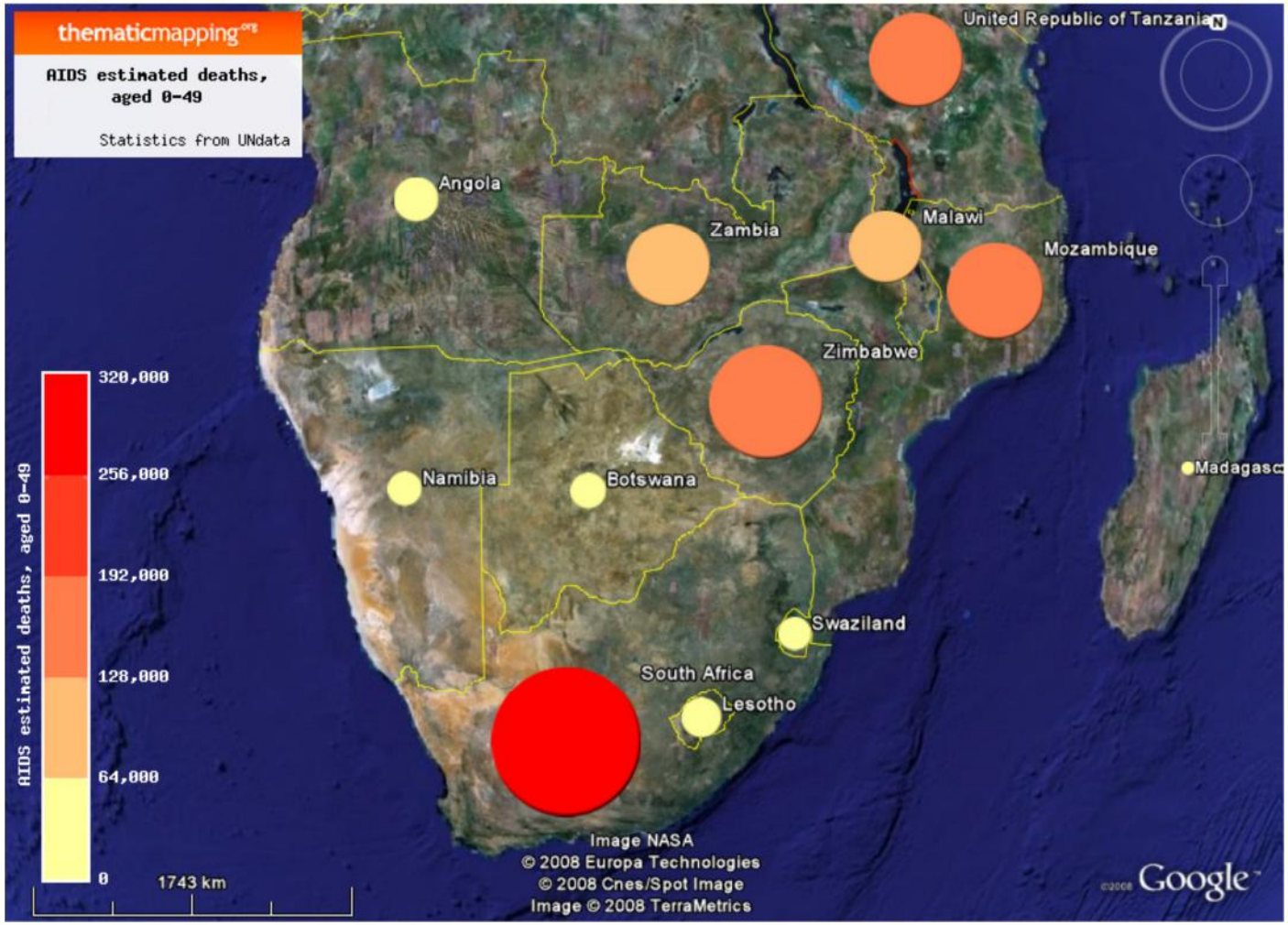
- Displaying thematic maps,
- Overlaying GIS layers onto satellite images,
- GPS track log viewer
- Etc..

KML Applications



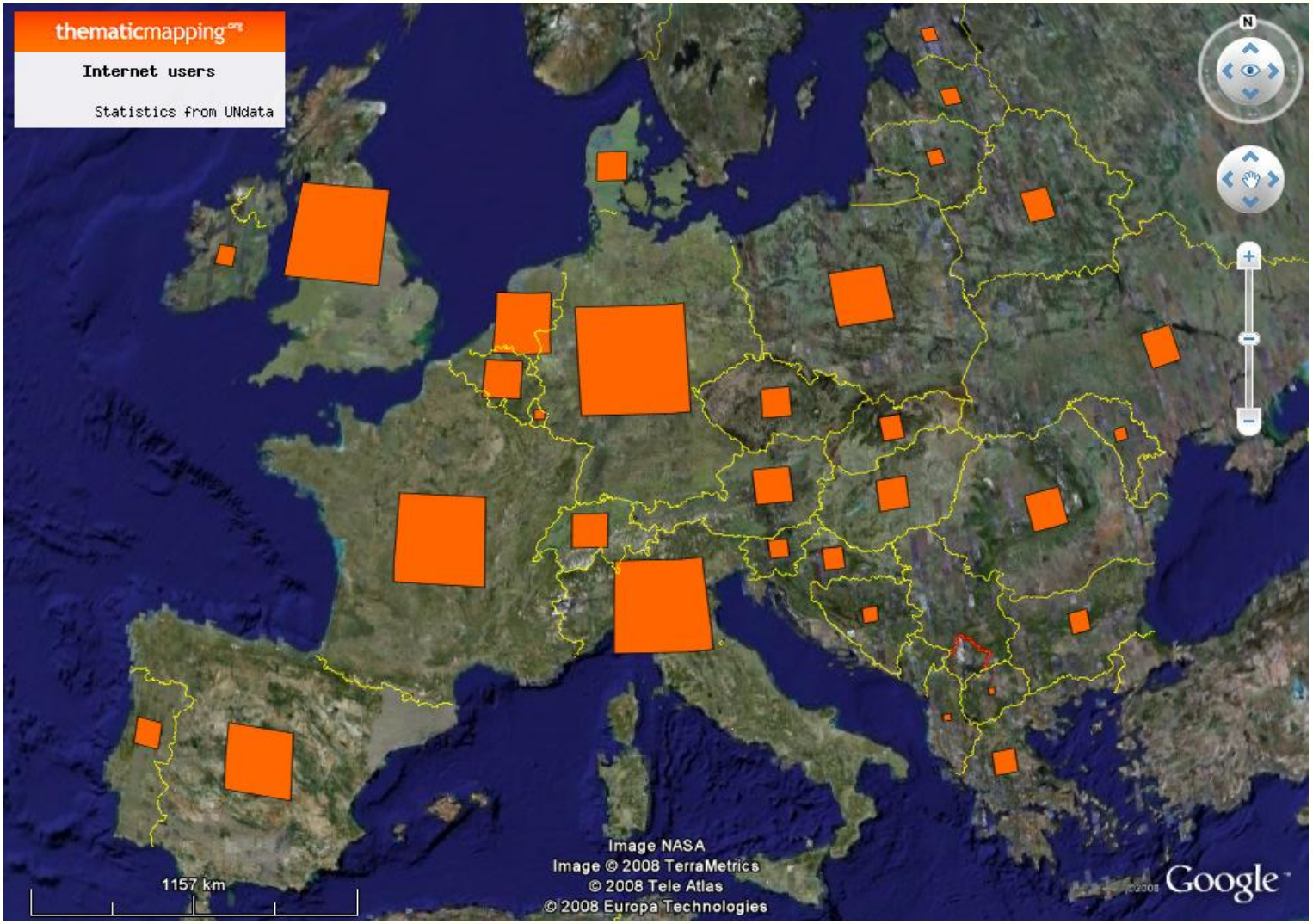
GIS layer overlay onto Satellite image.

KML Applications



AIDS estimated deaths (aged 0-49) in southern Africa in 2005.

KML Applications

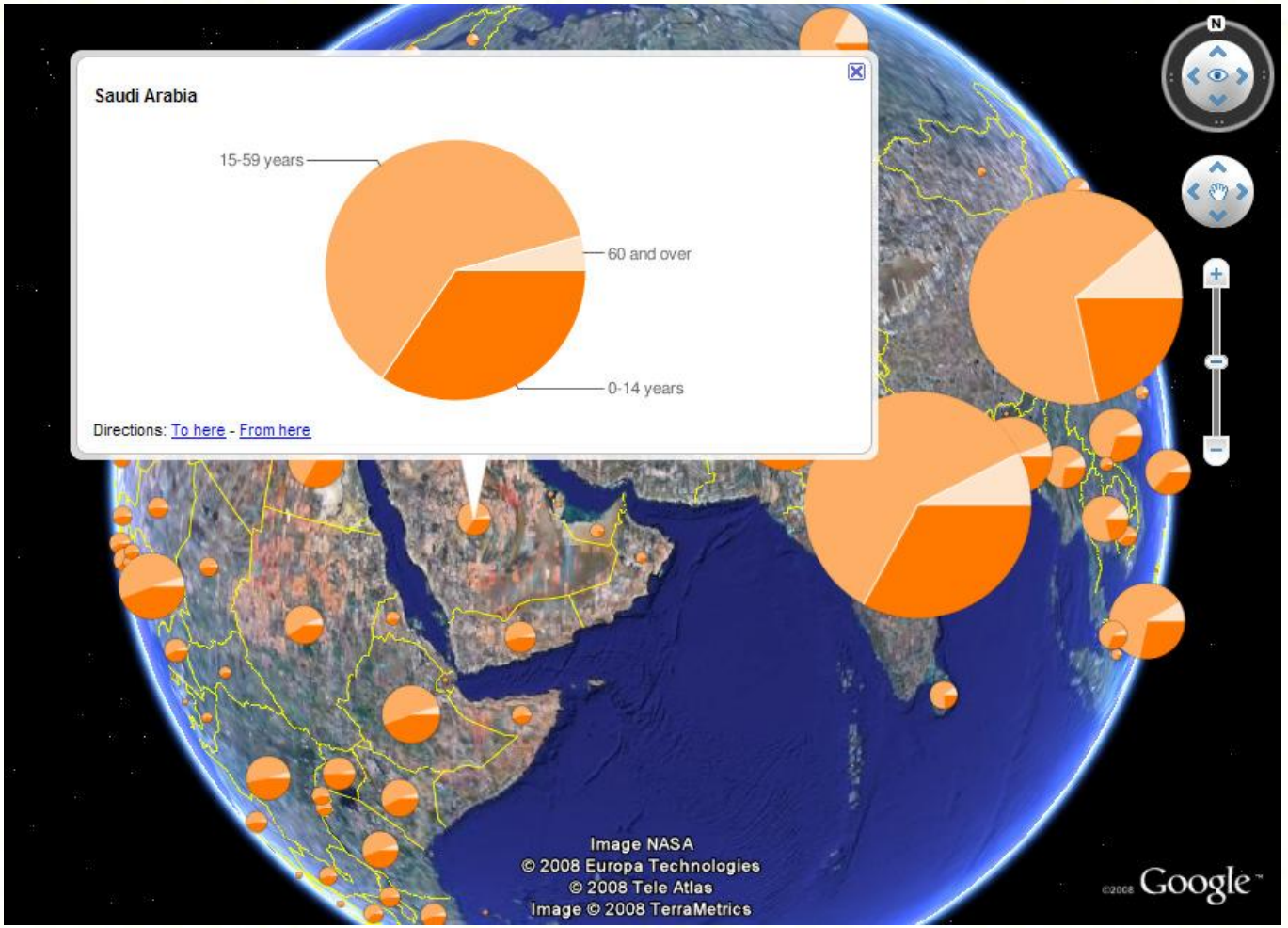


Number of internet users in Europe, 2008.

KML Applications

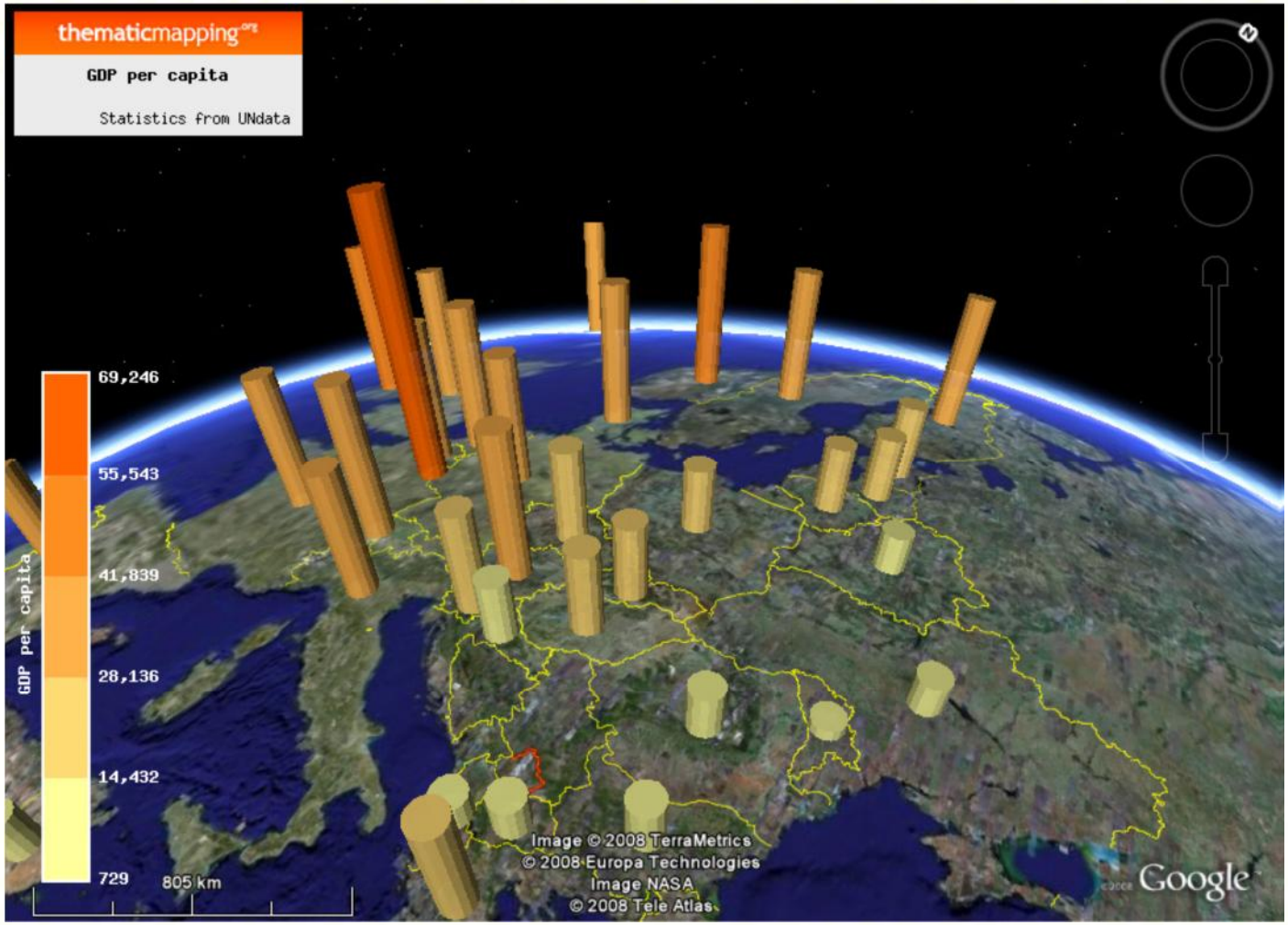
Mobile phone subscribers in South East Asia in 2004.

KML Applications



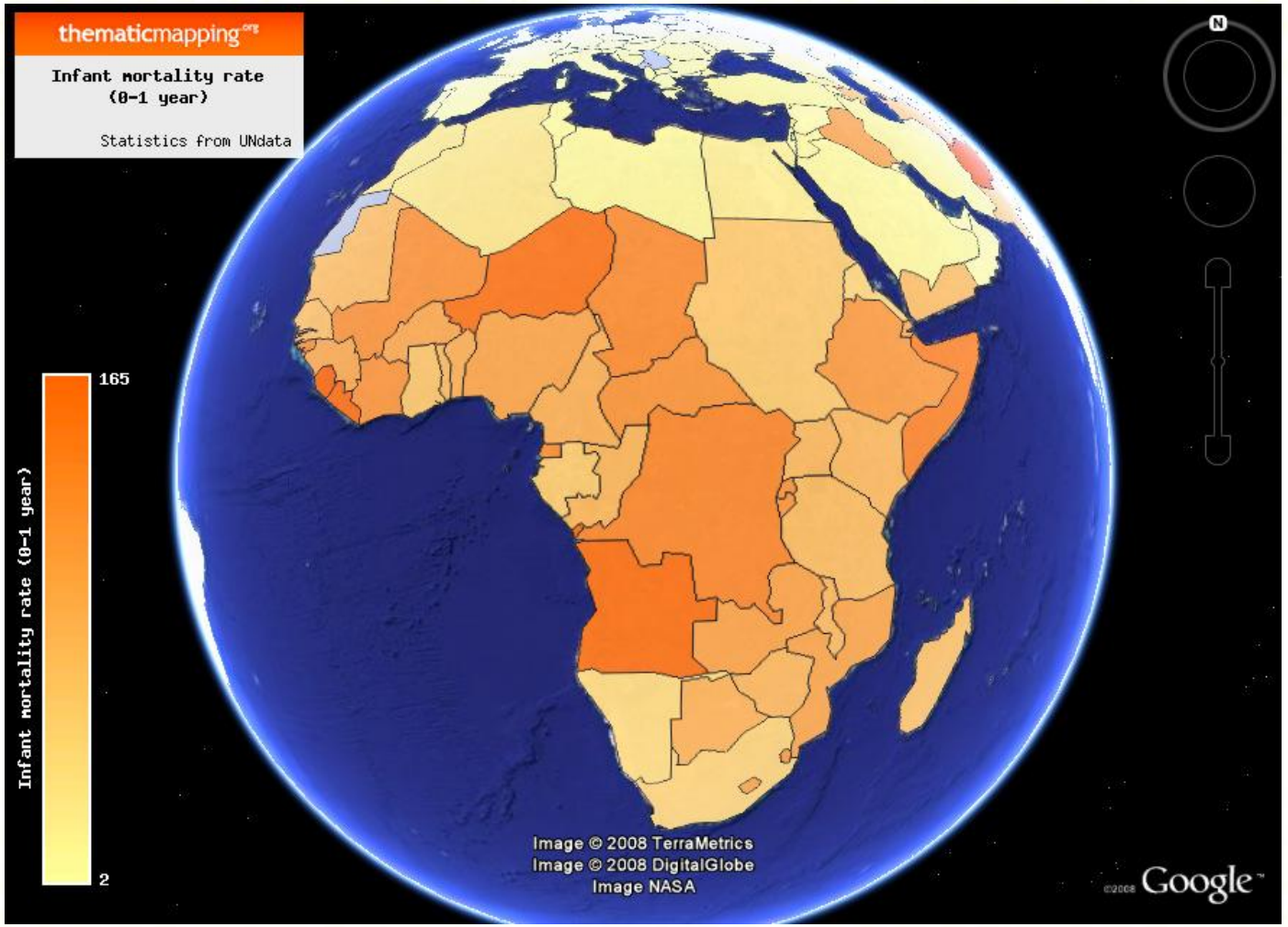
The charts are scaled according to total population and the pie shows the age distribution for each country.

KML Applications



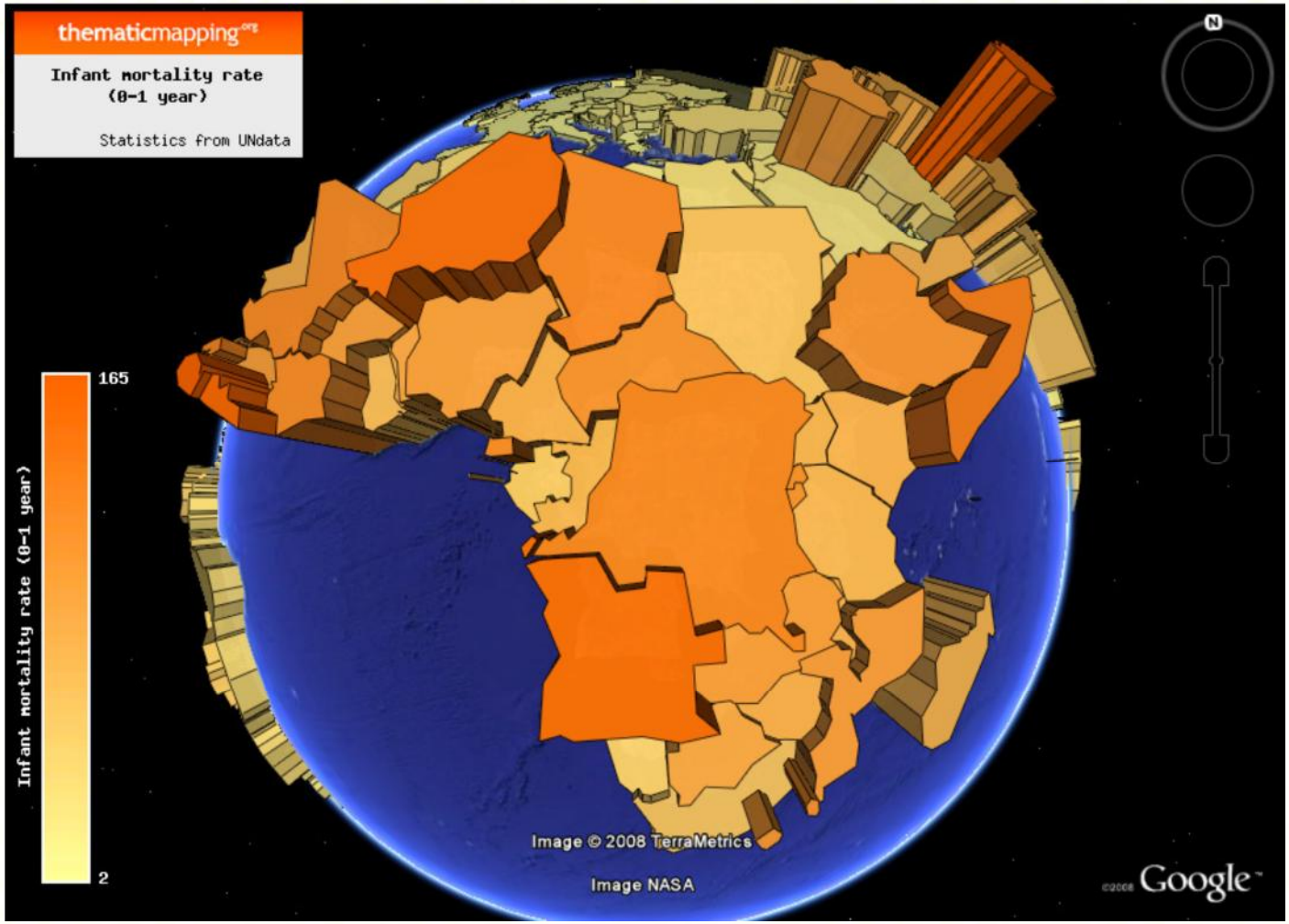
GDP per capital in European countries in 2006.

KML Applications



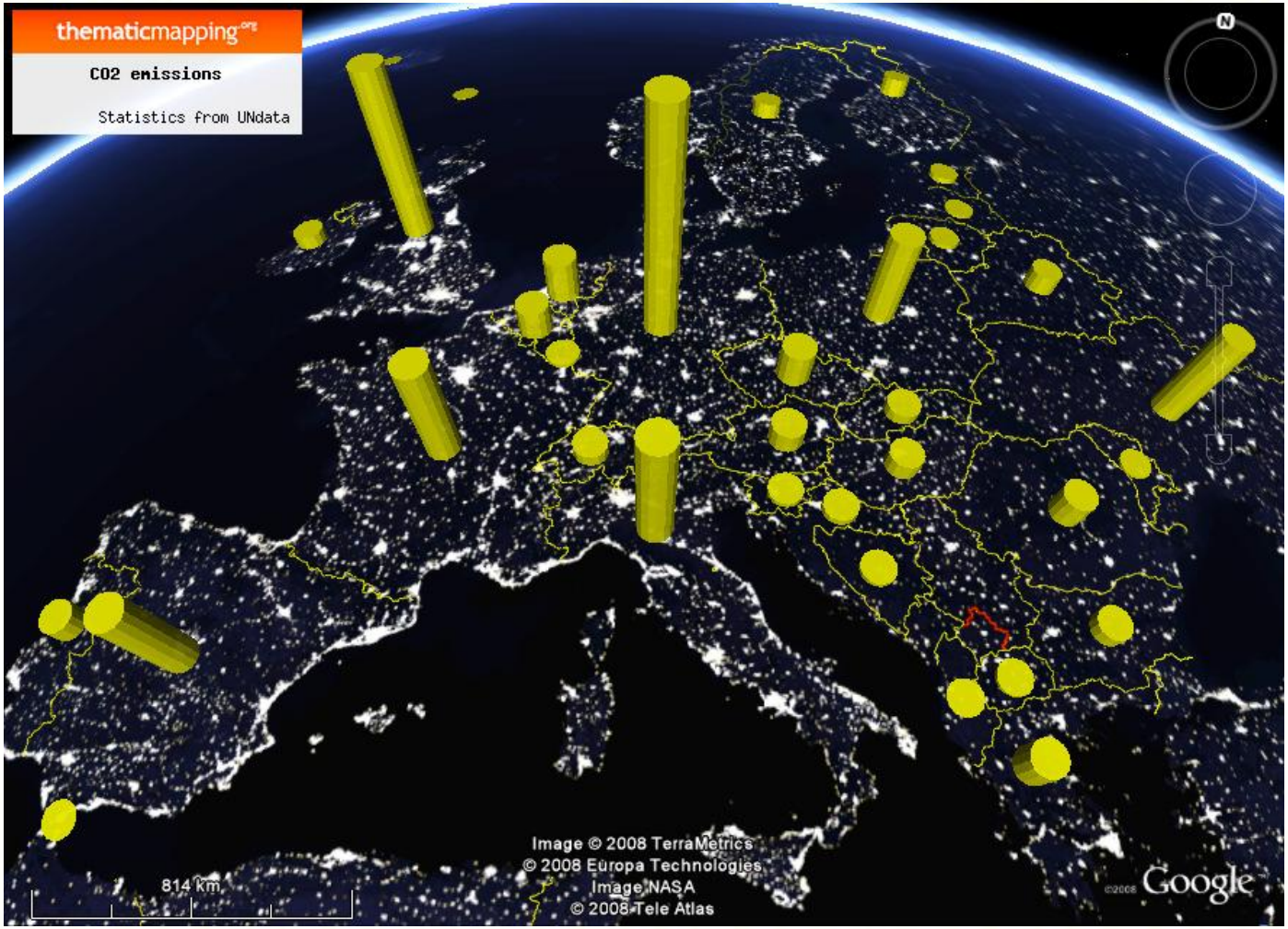
Choropleth map showing infant mortality rate in Africa, 2008.

KML Applications



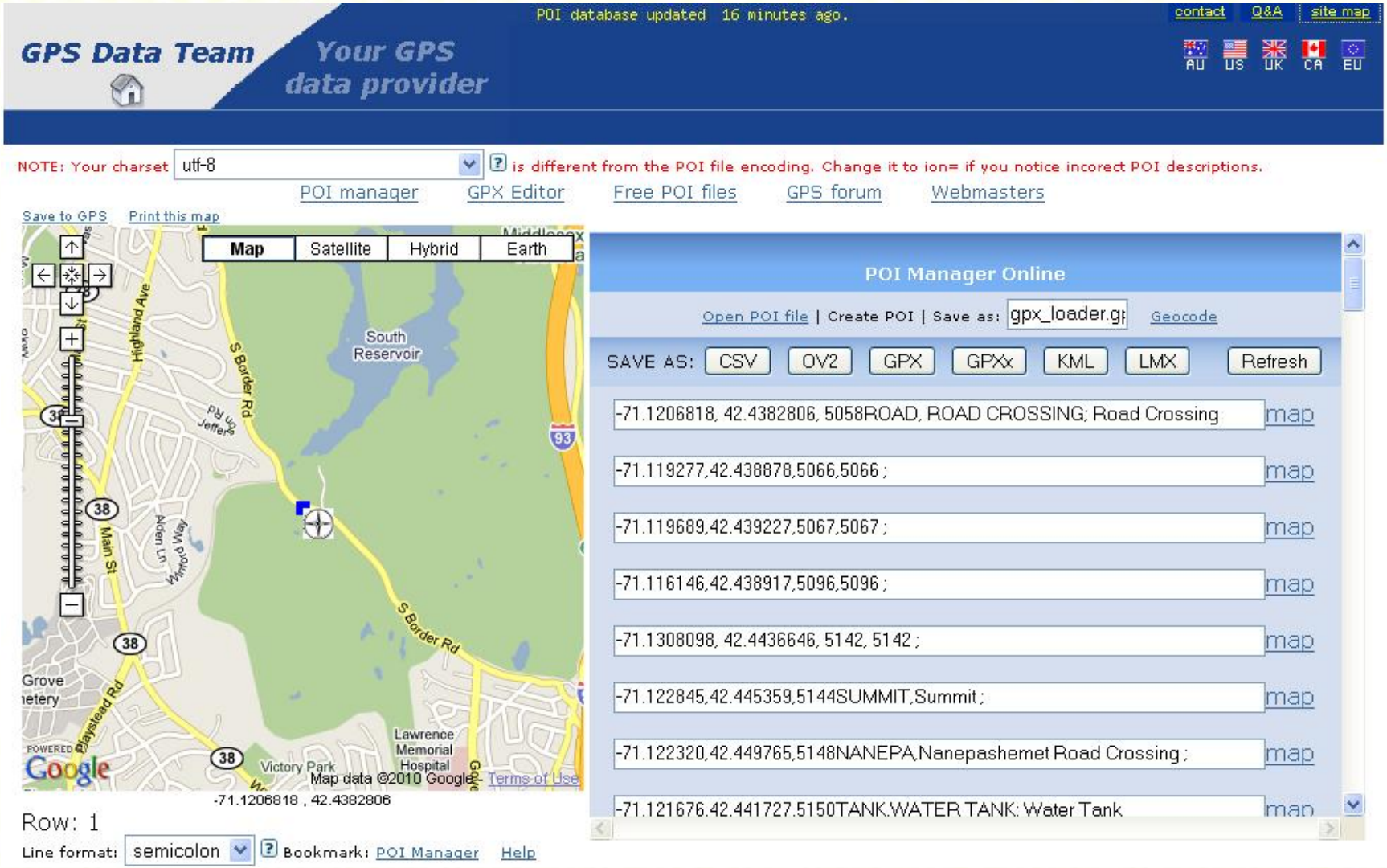
Infant mortality rate visualized as a 3-D prism map.

KML Applications



CO2 emissions from each country in Europe, 2008.

KML Applications

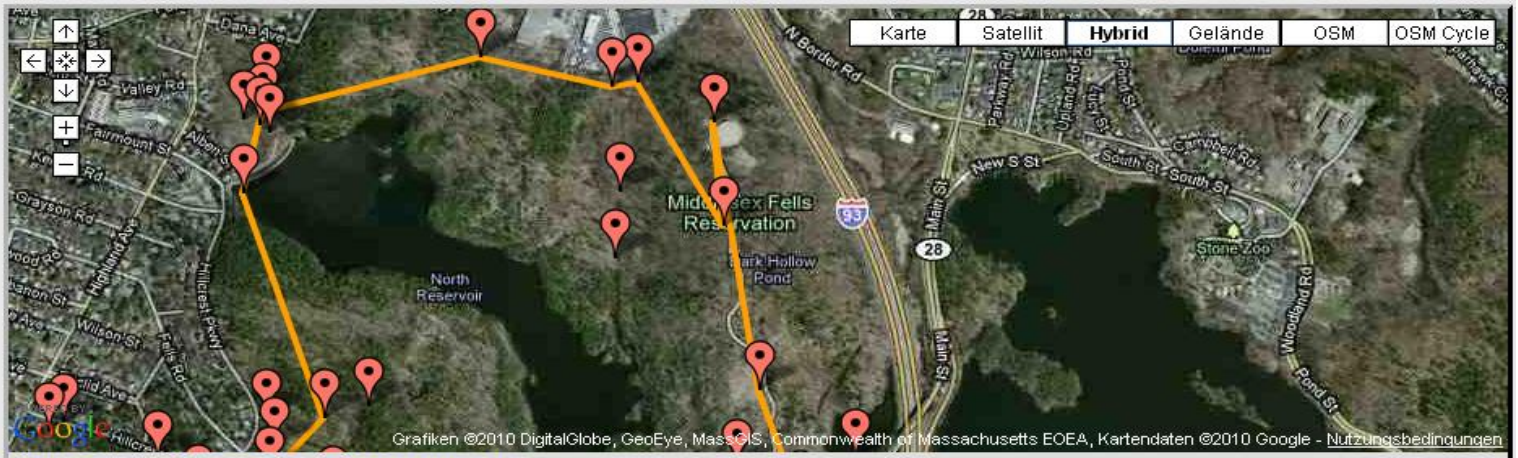


Online GPS waypoint editor.

Wandern mit GPS:

GPS Track Viewer

Anzeige von GPS Tracks, Routen und Waypoints im Kartenfenster

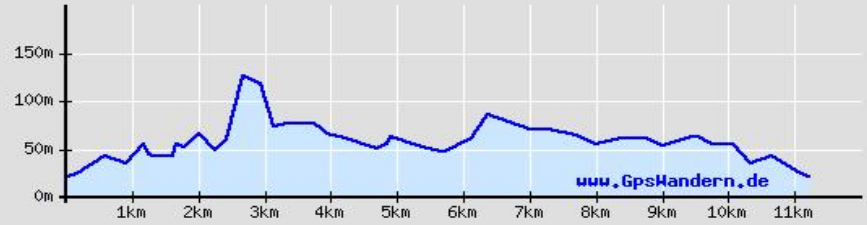


Fertig, Sie können eine weitere GPX-Datei auswählen.

Datei: D:\JabPendGoogle\gpx_loader.gpx | Browse... | Senden | Alles löschen

URL: http://muenchen-surf.de/karger/beispiel.gpx | WEB-Datei abrufen

Karte: S



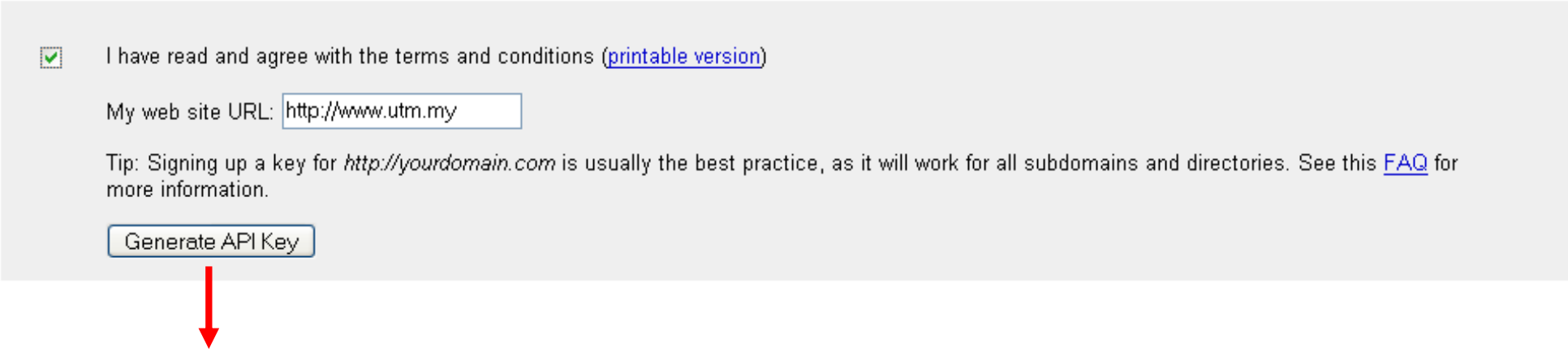
Höhendiagramm

Tourlänge:	1km
Höhe Start:	0m
Höhe Ziel:	0m
Höchster Punkt:	0m
Tiefster Punkt:	0m
Höhendifferenz:	0m
Aufstieg:	0m
Abstieg:	0m

KML Applications

As a **developer**:

- Sign Up for the Google Maps API (http://code.google.com/apis/maps/signup.html)

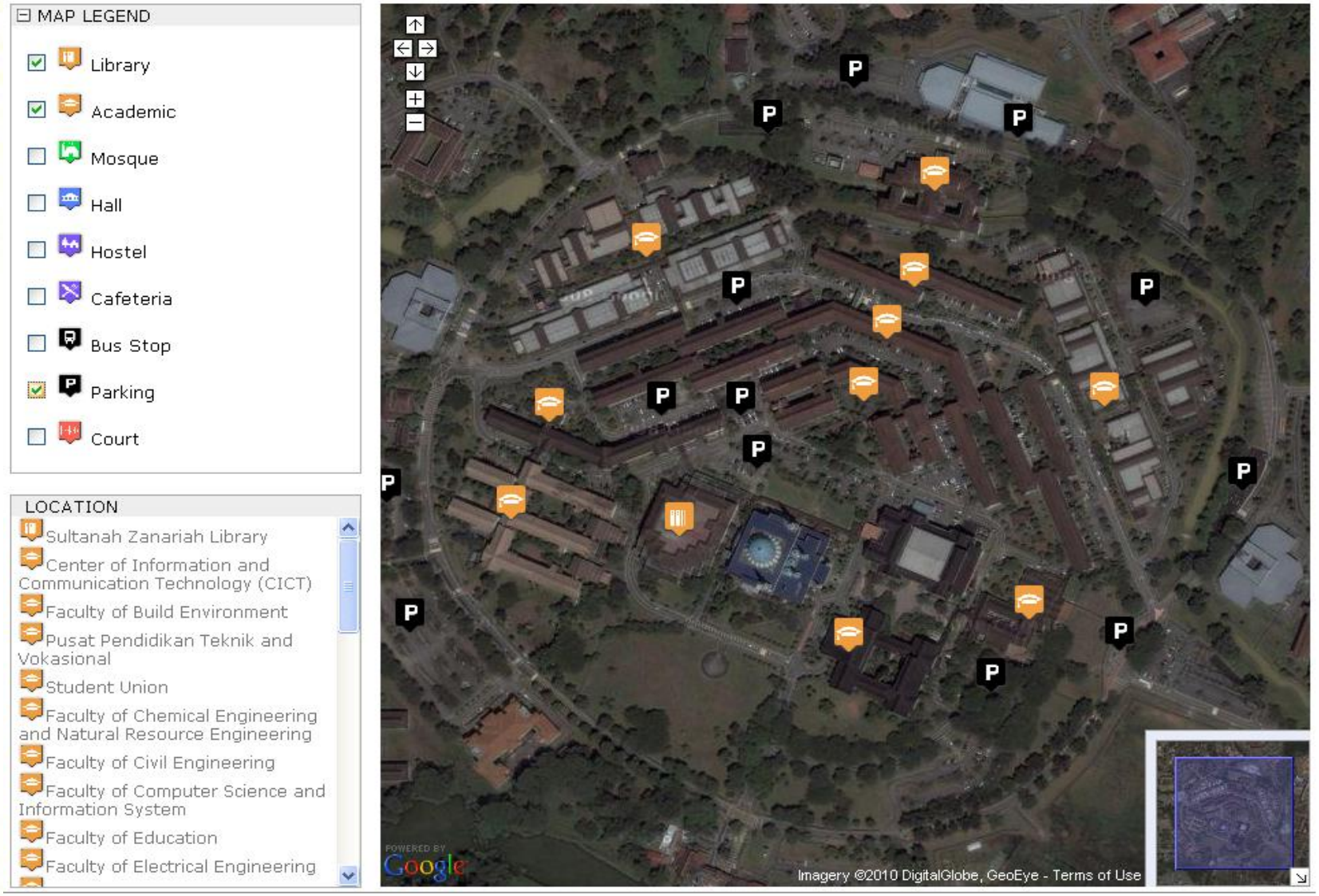



```
ABQIAAAAqkUwGG0FmCnU50FrUUsgqBRJAlemGgWXUKHZMRzyYmbBIYRtQBR
qMQuNxoAqctDofpp3JmhouxuMgQ
```

KML Applications

```
…
<script
src=http://maps.google.com/maps?file=api&amp;
v=2&amp; sensor=false&amp;
key=ABQIAAAAqkUwGG0FmCnU50FrUUsgqBRJAlemGgWXUK
HZMRzyYmbBIYRtQBRqMQuNxoAqctDofpp3JmhouxuMgQ
type="text/javascript">
</script>
…
```

Embedded in HTML / PHP coding as Javascript.

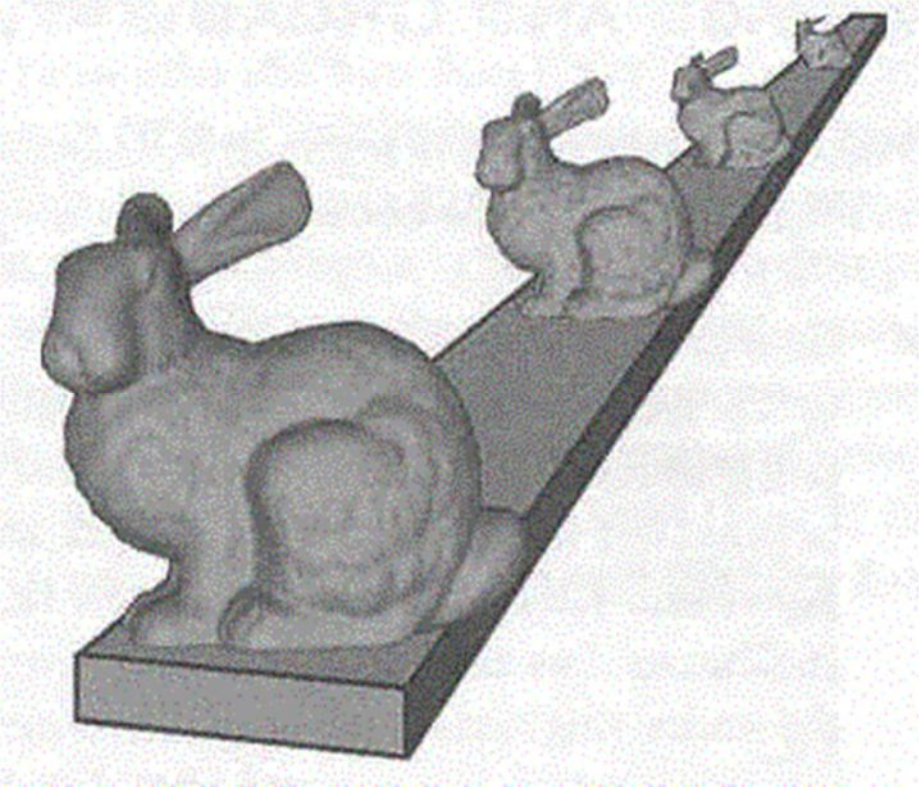


UTM's campus map

Level of Detail

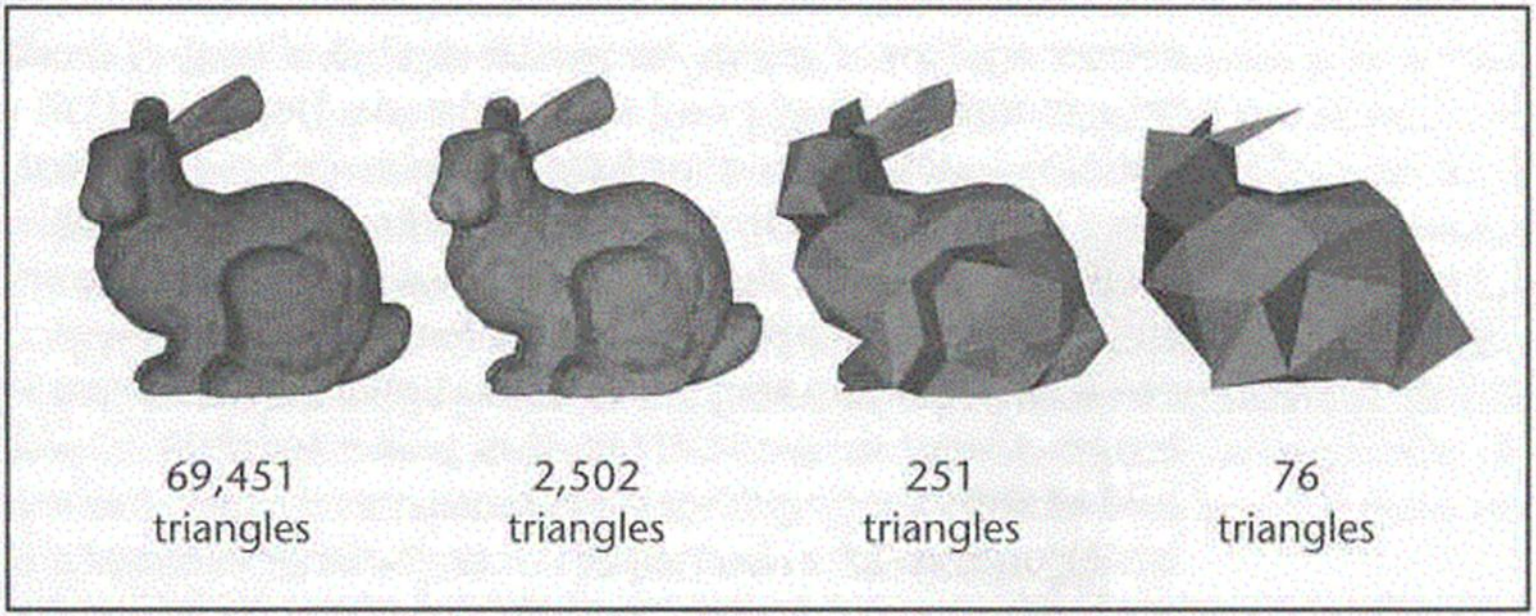
Level of Detail

- Discrete, Continuous & View-Dependent LOD
- Simplification operators
- Terrain LOD



Fundamental concept of LOD

- Simplify complex object.
- Create LOD to reduce the rendering cost of small distant or unimportant geometry.

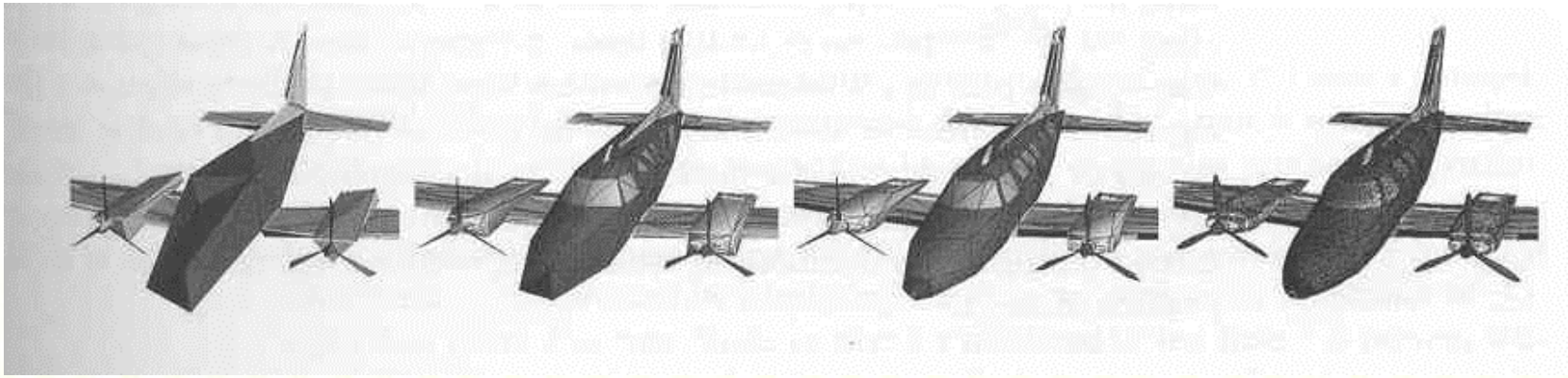


Discrete LOD

- Discrete LOD
  - Create multiple versions of every object during an offline process
  - At run-time chose the appropriate LOD
- LOD Node in VRML
- Disadvantages:
  - View independent
  - Popping effect

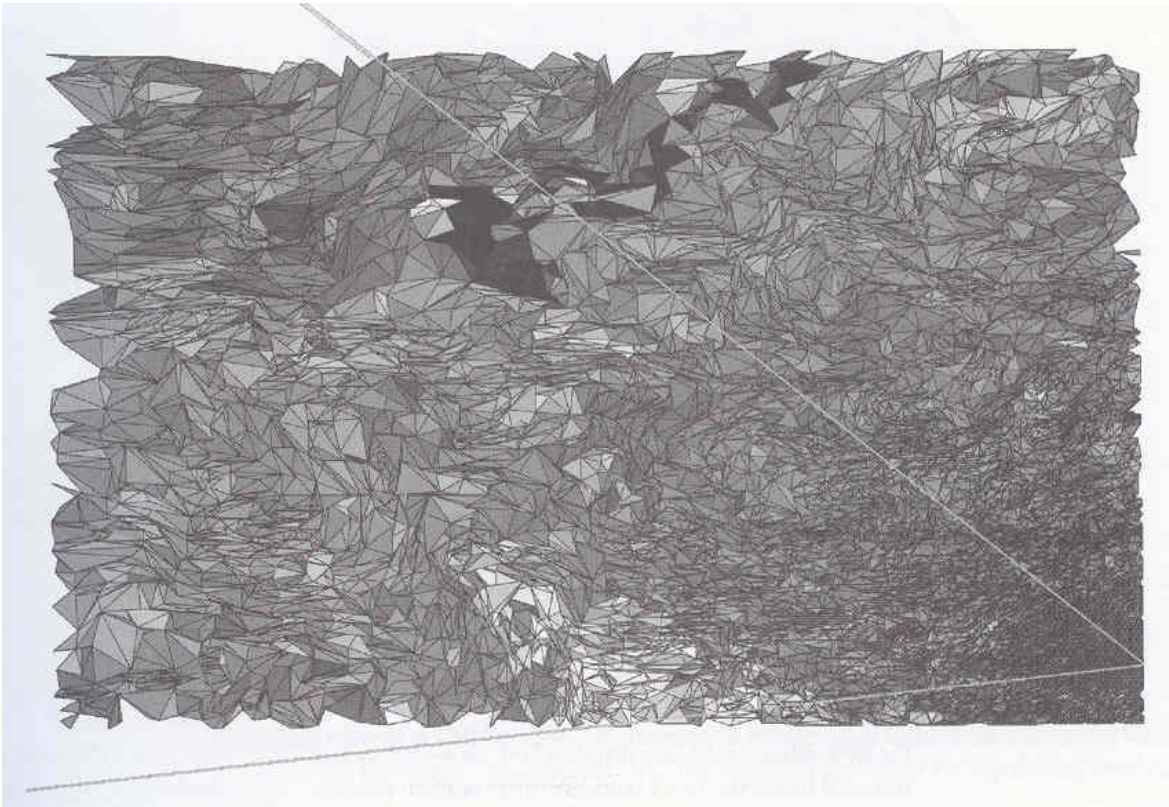
Continuous LOD

- Simplification process creates a data structure encoding a continuous spectrum of continuous LOD
  - Progressive Mesh (Hoppe 1996).
- Desired LOD is extracted from this data structure at run-time.
- Progressive Mesh LOD containing 150, 500, 1000 and 13546 triangles.

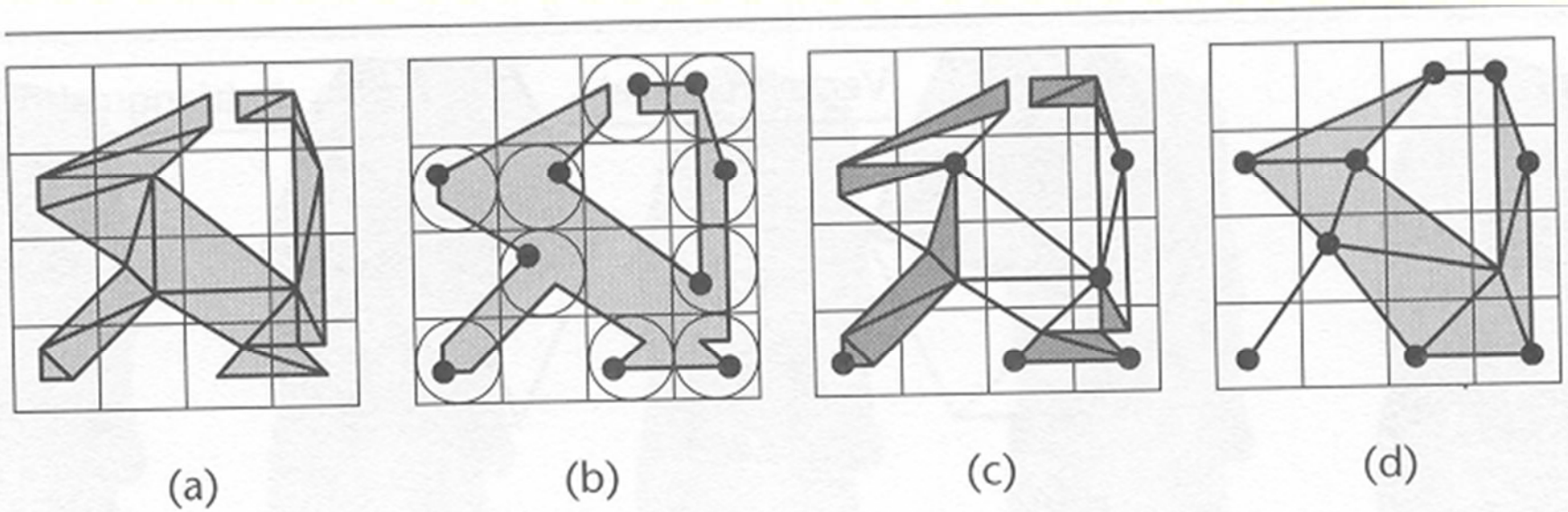


View-Dependent LOD

- Extends continuous LOD using view-dependent criteria to dynamically select the most appropriate LOD for the current view.

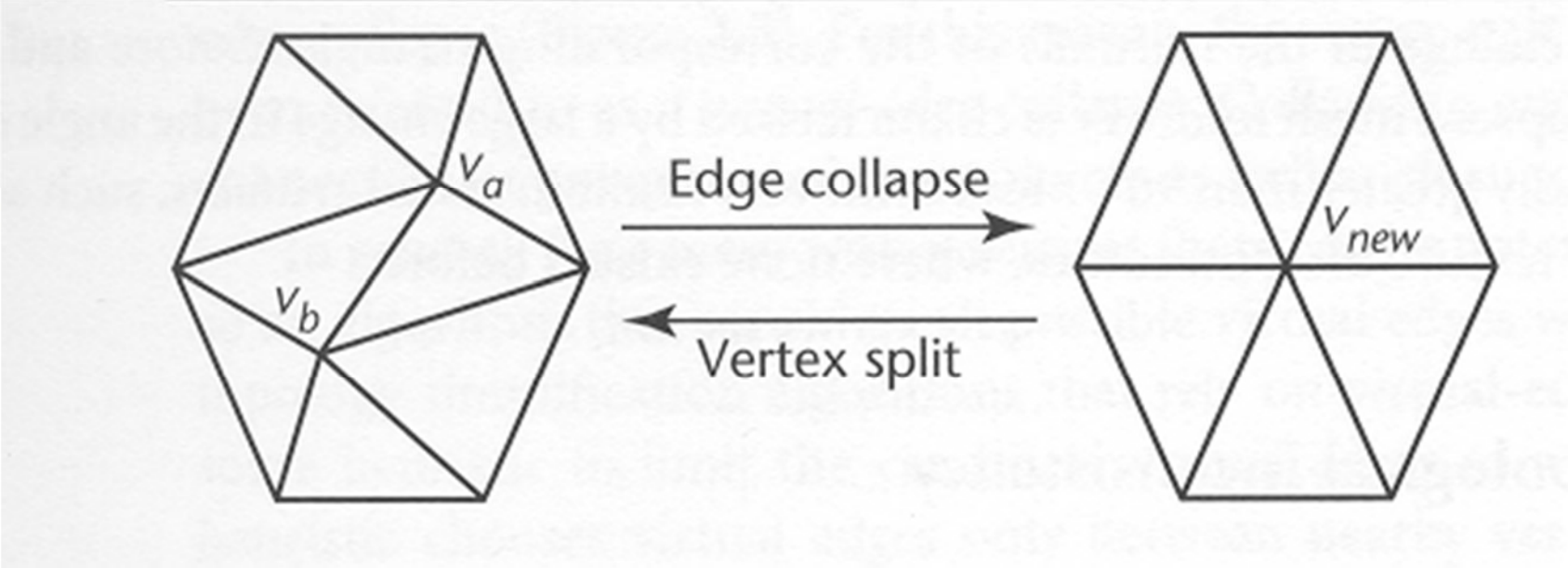


Simplification: Vertex Clustering



A cell collapse. (a) A regular grid classifies the vertices. (b) A single vertex is selected to represent all vertices within each cell. (c) Triangles with 2 or 3 corner vertices in the same cell simplify to a single edge or vertex, respectively. (d) The final simplification [Rossignac 92].

Simplification: Edge Collapse



Progressive Mesh stores a Mesh by a sequence of vertex splits!

Topology simplification

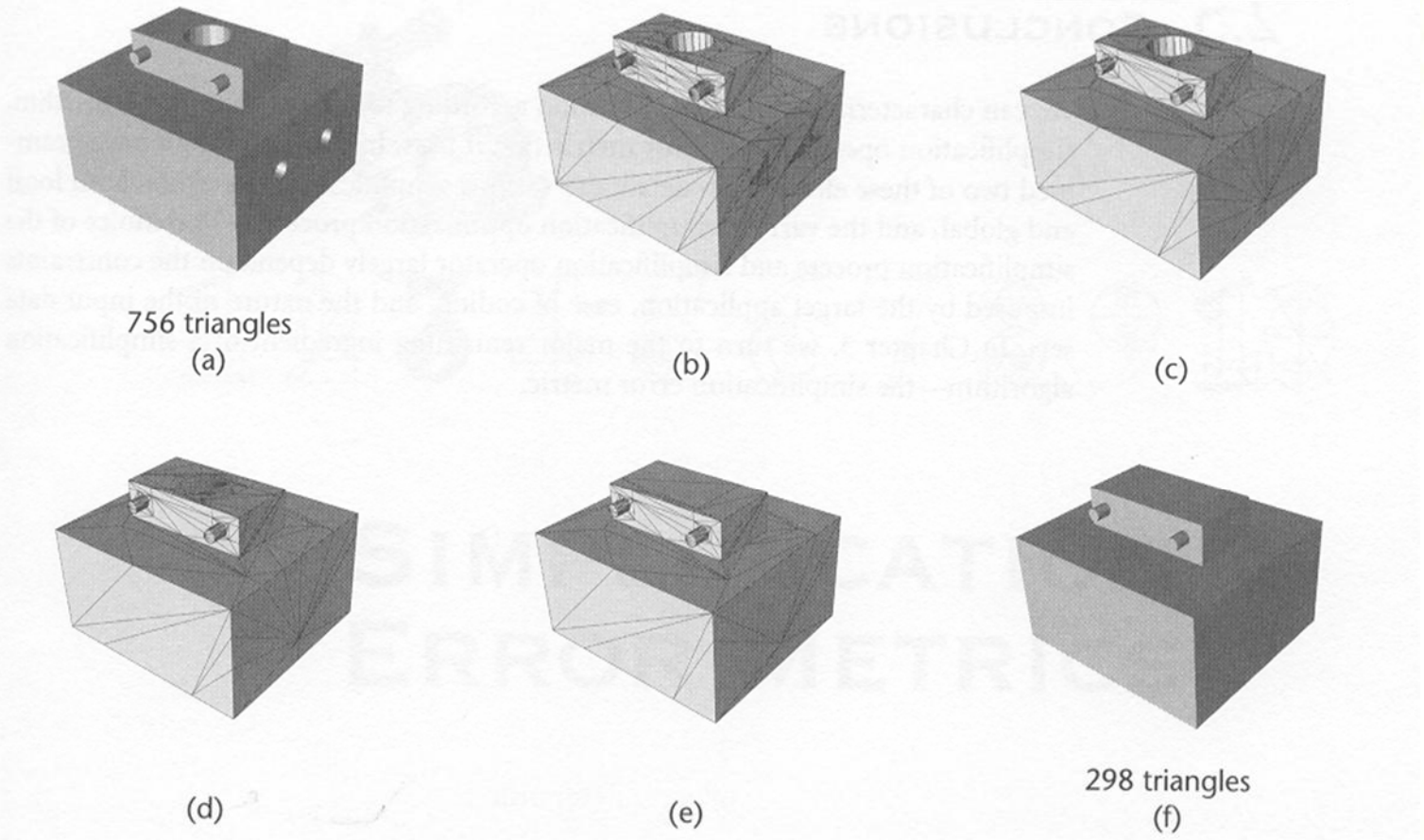
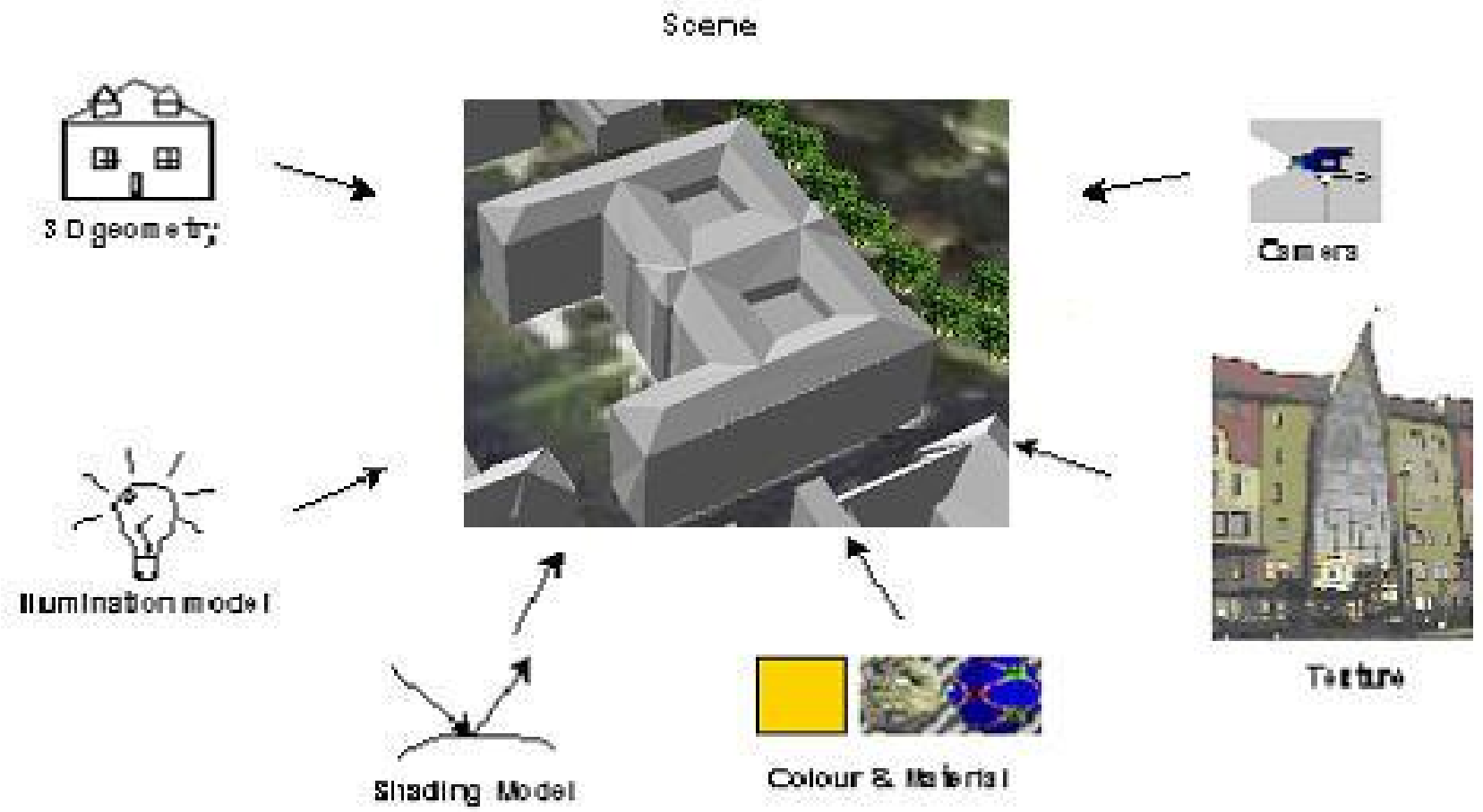
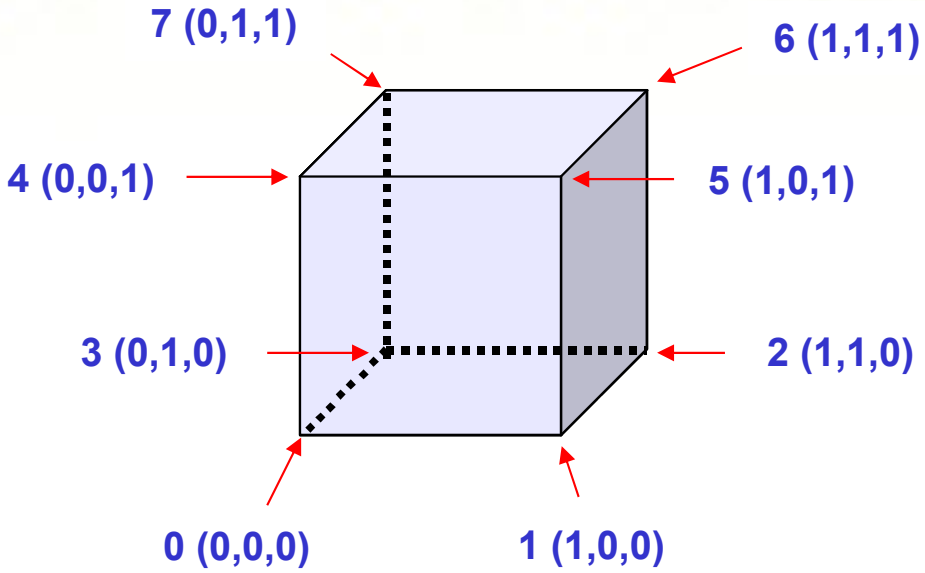


Figure 2.26 (a–f) Alternating topology and geometry simplifications [El-Sana 98]. Copyright © 1998 IEEE.

Components of a 3D Scene



Indexed Face Set



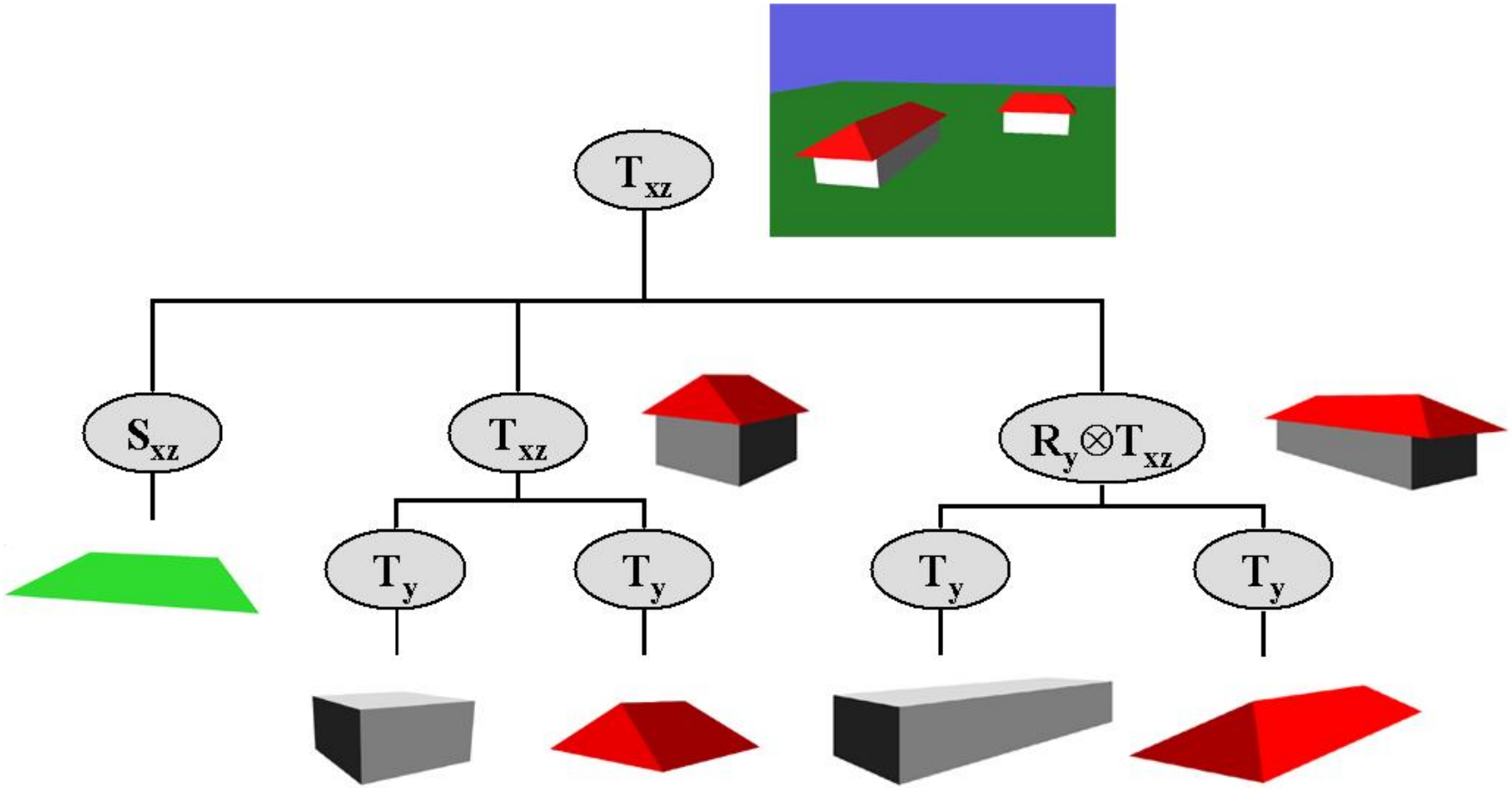
Points:

0 0,0 0,0 0,0
1 1,0 0,0 0,0
2 1,0 1,0 0,0
3 0,0 1,0 0,0
4 0,0 0,0 1,0
5 1,0 0,0 1,0
6 1,0 1,0 1,0
7 0,0 1,0 1,0

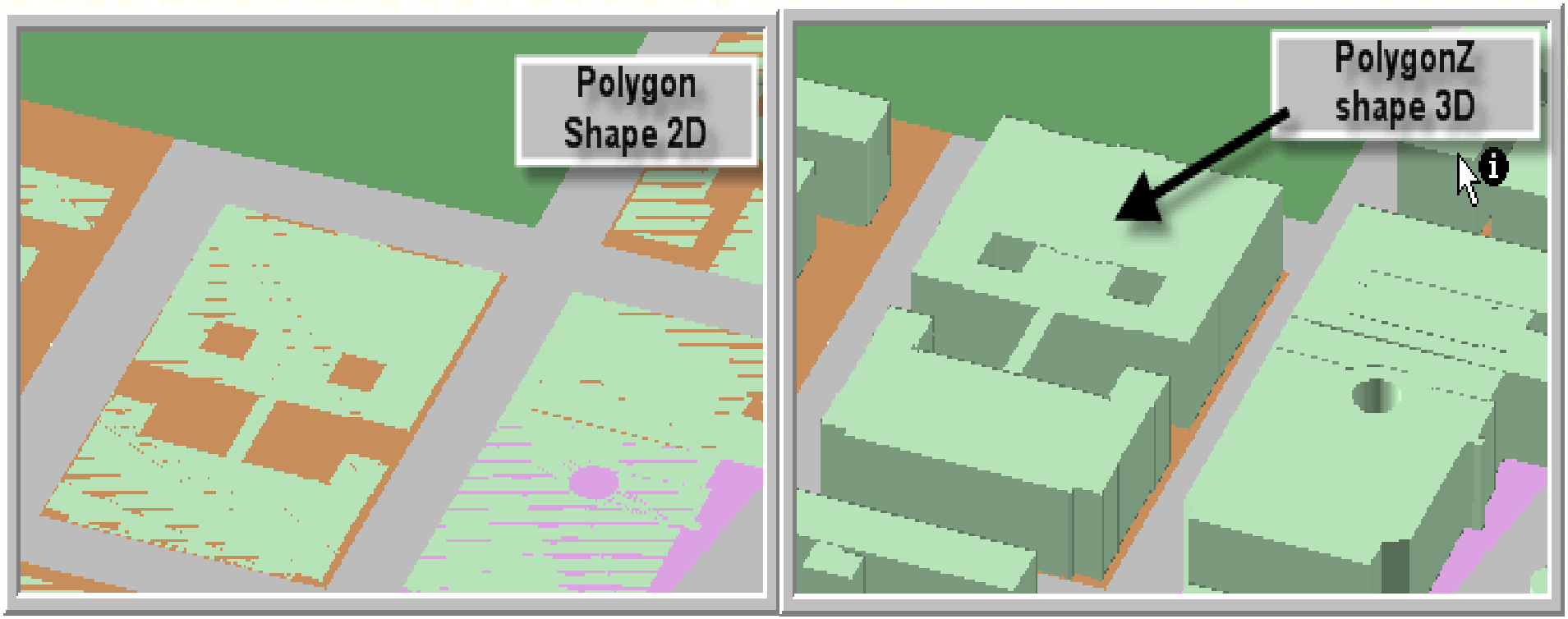
Faces:

0 0 1 2 3
1 0 1 5 4
2 1 2 6 5
3 2 3 7 6
4 3 0 4 7
5 4 5 6 7

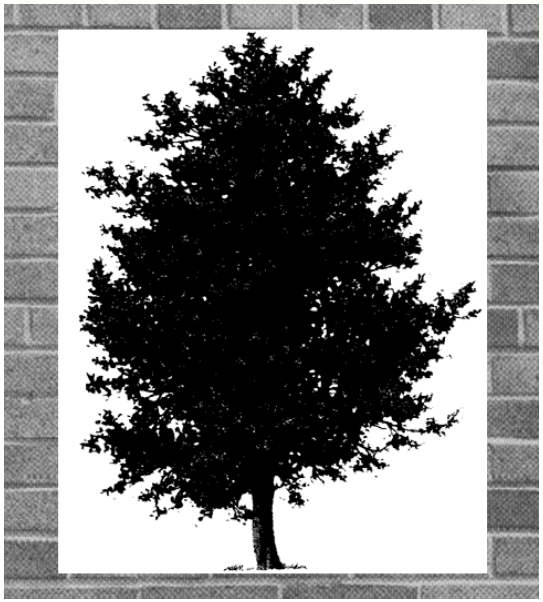
Scene Graph



Extrusion: from 2D to 2.5D

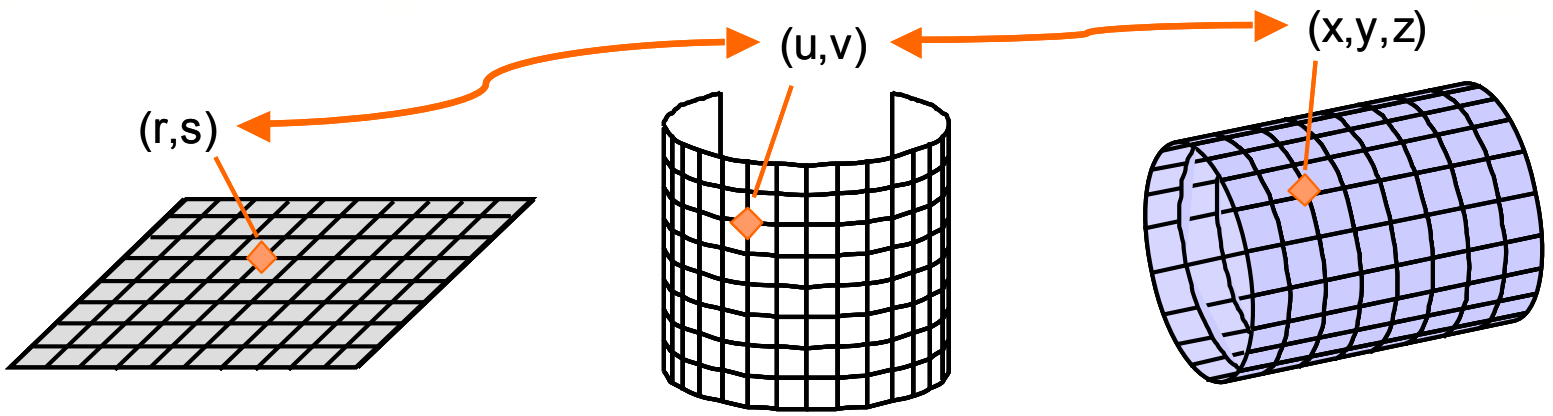


Texture (Alpha-channel)

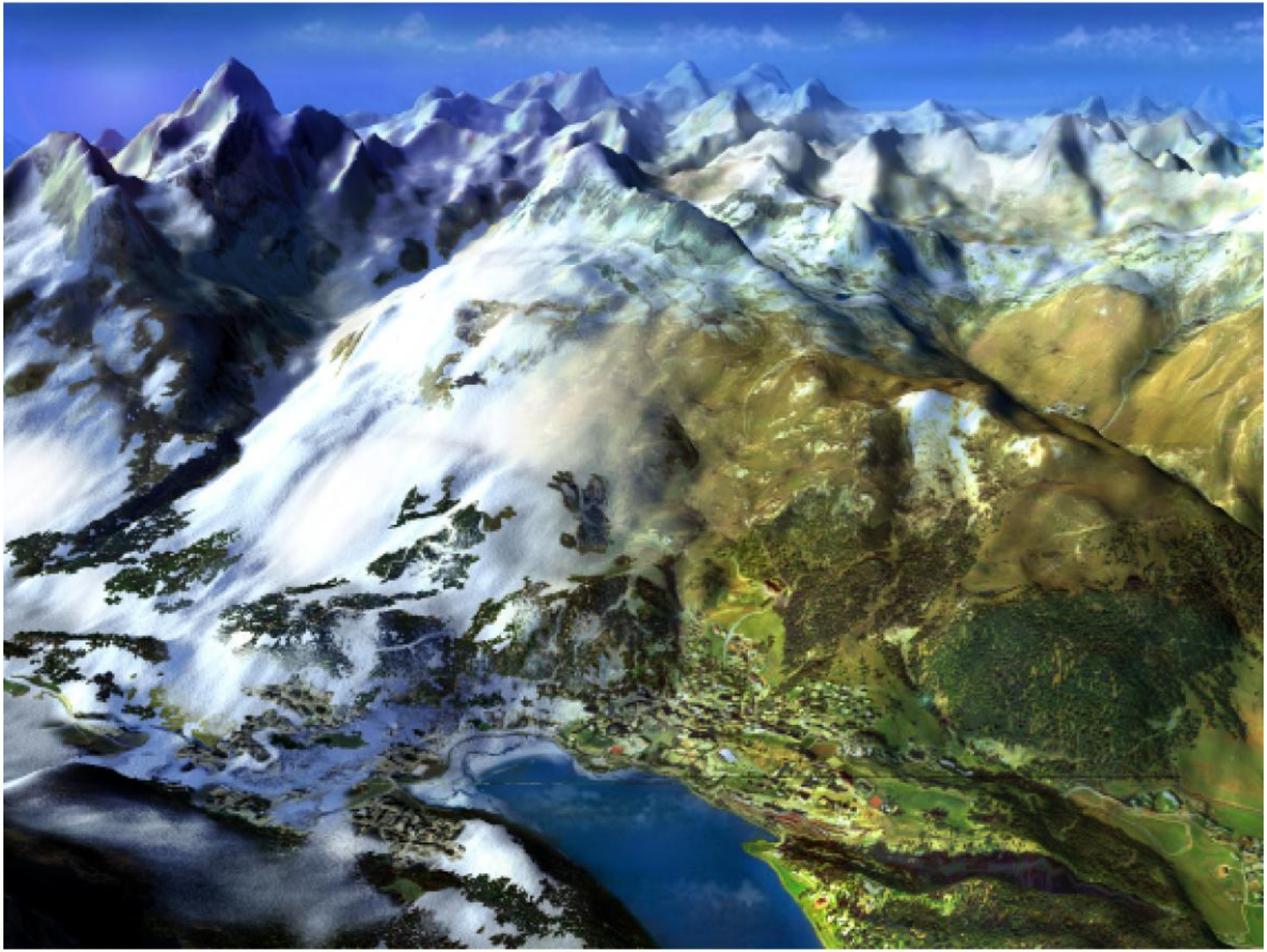


Texture coordinates

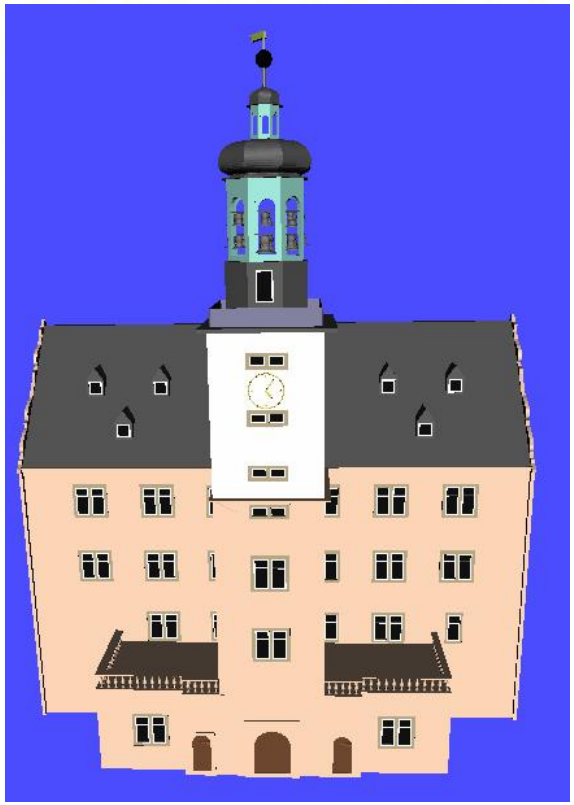
- Indexed Face Set
  - Coordinates (3D) are mapped to texture coordinates (2D)



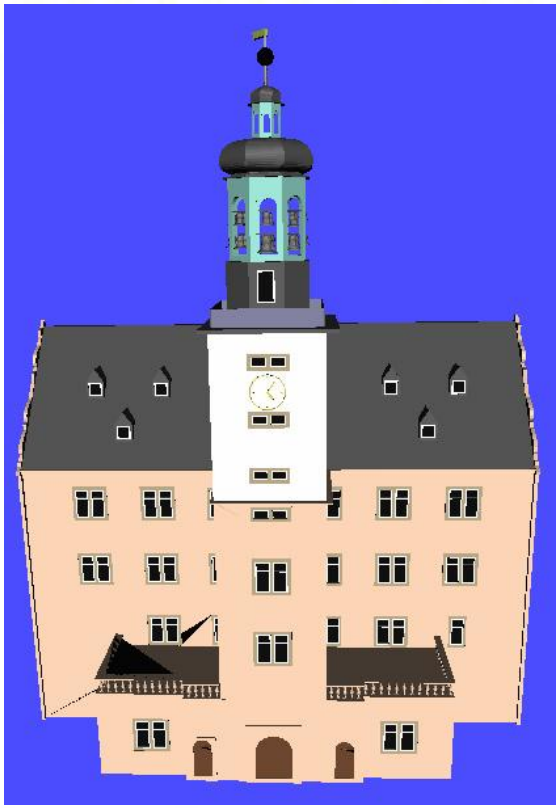
Texture



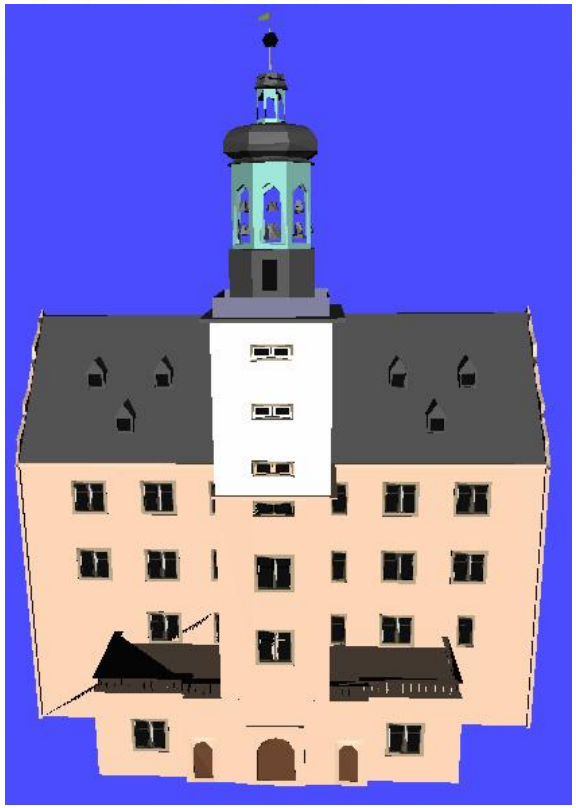
Level of Detail: Building



100K triangles



20K triangles



5K triangles