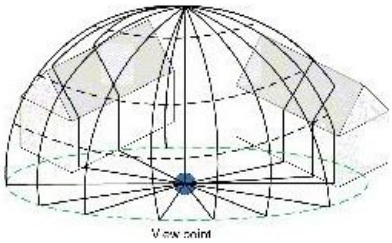
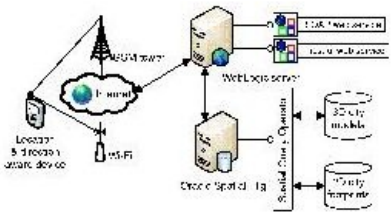
THREAT DOME VISIBILITY FOR SMARTPHONES

Three Dimensional Query





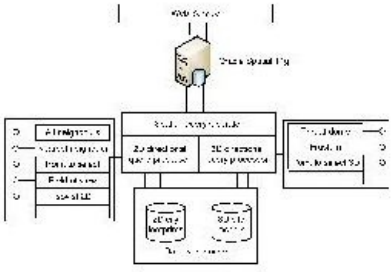


Figure 1, Threat-dome query shape interacting with 3D geo-datasets.

GPS receivers, digital compasses (magnetometers) and tilt sensors (accelerometers) are increasingly implemented in state-of-the-art smartphones such as the iPhone and various Android Phones. The resulting data on location, azimuth, and declination angles, together with images from inbuilt cameras, enable creation of a photorealistic and geometrically accurate city model. In turn, the newly created 3D scene can be queried wirelessly while moving through an urban environment. The authors developed a prototype - Three Dimensional Query (3DQ) - which tailors 3D visibility analysis to mobile devices based on a military-style threat dome computation.

The 'threat dome' approach is operational in dedicated military applications. Armed forces driving, for example, an armoured personnel carrier through an urban battlefield, want to know the sightlines to/from targets in real time. A virtual view-sphere (3D threat dome) is generated of a certain radius, aimed at identifying buildings or other objects around the viewpoint (Figure 1). Geo-information on objects that touch, overlap or otherwise interact with the dome can be retrieved via wireless communication from 2D and 3D datasets stored and indexed in an advanced Database Management System (DBMS).

Information Overload

Basing it on the threat dome approach, we developed Three Dimensional Query (3DQ) as a mobile spatial interaction (MSI) prototype linking users to the physical environment by retrieving and displaying spatial and non-spatial information using smartphones pre-loaded with web-based map services such as Google Maps and Yahoo Maps. Such devices may help real-estate appraisers, facility managers, tourists and others while walking through a city to obtain answers to questions such as: what are these buildings around me, or are there any points-of-interest (POIs) in my field-of-view? The volume of information on restaurants, hotels, ATMs, and other POIs available for querying and display is often overwhelming, both for devices and their users. Display clutter may confuse and disorientate the user, resulting in annoyance and apathy towards the usefulness of any Location Based Service (LBS).

Egocentric Querying

The threat dome approach facilitates an egocentric visibility query process, which enables the introduction of Hidden Query Removal (HQR) functionality in conjunction with other map personalisation or semantic filtering mechanisms. HQR ensures that a query returns information on only those objects that a user can actually see from their current position, and vice versa, thus supporting de-cluttering and reducing information overload. For example, new students can explore their university campus by pointing a smartphone at a building, or even at specific floors or rooms, to retrieve information indicating whether these are labs, offices or classrooms. Depending on the contents of the database, questions could be answered concerning whose office window is being pointed at, or what classes are

timetabled to take place there, or, perhaps more interestingly, "Can I see any of my Facebook/Twitter Friends from where I'm sitting?" or indeed "Can they see me?"

Flexible Database

A specialised DBMS is required for storing, indexing and managing such large amounts of dense geodata. Performing simple directional queries requires a database that supports 2D spatial objects. In addition, efficient topological operators for determining interactions between objects should be available. Most commercial databases, such as IBM DB2 Spatial Extender, Oracle Spatial, and PostGIS, have these capabilities. However, performing 3D directional queries is more complex and requires storage and indexing of 3D objects. For instance, a building is stored as a polygon in a 2D database and as a solid in a 3D database. Moreover, to support 3D queries, functionality should be available to determine which building, or part of it, a direction vector interacts with. These advanced requirements limit database options. We opted for Oracle 11g, as this database supports many coordinate systems and identification of 3D topological relationships. It also sustains storage, indexing and modelling of large sets of geo-objects, such as buildings in cities, for further query processing. 3D objects are modelled as composite solid, solid, surface, and polygon; corresponding respectively to building, house, roof, and window. The architecture of 3DQ is split into server-side and client-side, where data exchange is based on either a SOAP or RESTful style web-service (Figure 2). A WebLogic server provides the interfaces; request and response are wrapped in an XML document, thus allowing portability across mobile device platforms.

Figure 2, Overview of 3DQ architecture.

Present Prototype

Our prototype consists of a Nokia Navigator 6210 smartphone installed with 3DQ and equipped with an integrated GPS receiver, magnetometer, and accelerometer (Figure 3). Data is accessed using Python for Symbian Series 60 (PyS60) Operating System. The database covers Dublin City. A SOAP library port for PyS60 (SOAPpy) gives access to the web-service on the 3DQ server. The entire query/retrieval process is performed on the server-side, with as primary input location, direction and tilt provided by the smartphone. 2D queries include (see Figure 4, left part of schematic overview):

Figure 3, Mapping interface on Nokia Navigator 6210; green arrow indicates query direction.

- standard range queries to find all neighbours and nearest neighbours

- single ray directional queries (Point-to-Select 2D)

- directionally constrained (Field-of-View) queries with HQR functionality.

- full 360º queries (Isovist).

The 3D query processor facilitates three types of query. Depending on the level of detail of the dataset, point-to-select 3D identifies which building, floor and even window the device is pointing at, functionality which is not delivered by Point-to-Select 2D. In 3D, the elevation of the user is taken into account along with the real-world dimensions of the buildings and other database objects.

Pointing over a lower building to a higher one behind it, the 3DQ processor can recognise this difference. Detailed information about individual objects in a 3D Field-of-View (Figures 5 and 6) can be generated with a frustum view query, which can be thought of as a ‘squared' flashlight beam scanning a wall to retrieve information about areas ‘illuminated'. Finally, a 360º Isovist view in three dimensions out to a specified radius (threat dome) can be generated. The above queries are all implemented by generating the respective query shape as a 3D object (i.e. 3D query ‘window') in a spatial database, and then utilising inherent 3D query operators to identify topological relationships. For example, the Oracle SDO_anyinteract query operator will retrieve all database objects that intersect in any way with the query shape, while the SDO_intersection operator will retrieve the coordinates of the intersection point(s). From these data the query process is refined to include only those objects actually visible to the user, thus reducing information overload on a mobile device.

Figure 5, 3D frustum query in real-world environment.

Concluding Remarks

Further tests will be carried out on a highly detailed campus model constructed from Lidar data. We are investigating making available the 3DQ technology to a larger audience through various online app stores such as iPhone App Store, Android Market and Nokia OVI Store, for user testing and possible commercialisation opportunities.

Acknowledgements

Research presented here was funded by a Strategic Research Cluster Grant (07/SRC/I1168) by Science Foundation Ireland (SFI) under the National Development Plan.

Further Reading

Gardiner K., Yin J., Carswell J.D., 2009, EgoViz - A Mobile Based Spatial Interaction System; 9th International Symposium on Web & Wireless GIS (W2GIS), Maynooth, Ireland: Springer LNCS Vol. 5886, pp135-152.