Virtual City Maker And Virtual Navigator: A Modelling And Visualisation Solution For Mobile 3D Virtual Cities

Christos Gatzidis[1], Fotis Liarokapis[2], Vesna Brujic-Okretic[1], Stuart Baker[3]
giCentre, Department of Information Science, City University, London, EC1V 0HB, UK
{c.gatzidis, vesna}@soi.city.ac.uk[1]
Faculty of Engineering and Computing, Department of Creative Computing, Coventry, CV1 5FB, UK
F.Liarokapis@coventry.ac.uk[2]
Alcatel-Lucent Telecom Limited, Christchurch Way, Greenwich, London SE10 0AG
Stuart.e.Baker@alcatel-lucent.co.uk[3]

ABSTRACT
This paper presents a complete procedural 3D modelling solution for mobile devices, called Virtual City Maker, based on scripting algorithms allowing for both the automatic and also semi-automatic creation of photorealistic quality virtual urban content. The input data used include the combination of aerial images, GIS data, 2D ground maps and terrestrial photographs grouped with a user-friendly customized interface which permits for the automatic and interactive generation of large-scale, accurate, georeferenced and fully-textured 3D virtual city content. This content can be specially optimized for use with mobile devices but also with navigational tasks in mind. Moreover, a user-centred mobile virtual reality (VR) visualisation and interaction tool operating on PDAs (Personal Digital Assistants) for pedestrian navigation is also discussed. This engine supports the import and display of various navigational file formats (2D and 3D) and also includes a comprehensive front-end user-friendly graphical user interface providing immersive virtual 3D navigation to a wide range of users.

KEY WORDS
Applications, 3D urban modelling, mobile navigation

1. INTRODUCTION

Today there is a growing need for computer-based, photorealistic visualizations of 3D urban environments in many areas including environmental planning, engineering, telecommunications, architecture, gaming, 3D city information systems and even homeland security. Therefore, the procedural modelling of virtual cities in 3D is a topic not only computer graphics research but also other fields such as GIS or photogrammetry have focused on for a number of years. Different approaches have been favored with the two main ones being a fully-automatic or a semi-automatic one.

While the ideal 3D urban modelling system would be a totally automatic one, there are a number of obvious advantages to semi-automatic or user-assisted approaches where essentially an automatic system employs some limited operator input for guidance in a range of the tasks involved in order to offer better control on the resulting scene. Furthermore, the user-assisted (or interactive) tools can also be utilized as a more intelligent editing application for the preliminary model. For example, a popular approach is to offer suggestions to an automatic urban modelling system which then in turn concludes the modelling task at hand on its own. This can lead to a more efficient method for modeling complex structures. A more conventional approach to interactive modeling is to provide the user with a set of generic models that are then adjusted (using usually image data) altering the model and the viewing parameters [1]. In this approach, the system provides geometric computations but the drawback is that substantial time and effort are required from the user. Some other more recent approaches include the combination of user input with automatic processing introduced at various points with a different degree of control over the result. Offering an approximate building location to extract a building is one of the approaches suggested [2] that can lead to results although there is the important disadvantage of producing a final model very reliant on the automatic analysis. Other semi-automatic urban modelling tools have been described, some including methods for duplicating and/or cloning model meshes that are similar to others [3]. Yet another approach suggests an automatic system constructing topological relations amongst 3D roof points collected by a user for each roof. This method [4] results in a tool that can tackle easily several types of complex roofs. An additional notable interactive urban modelling approach involves the system handling complex building structures by means of constructive solid geometry [5]. This system also uses an image correlation method to fit a primitive to the image. It should be noted however that this last method can be very costly on processing power when modelling more complex urban sites. One of the areas where 3D urban models are also in need today, apart from the ones mentioned in the beginning of this section, is for urban navigation using mobile devices. This can range to a number of sub-applications including car and pedestrian navigation systems, location-based services and others. It appears that while the methods for automatic and semi-automatic creation of 3D urban models today do cover a lot of ground both in terms of concepts and also in delivery of content, they fail to take into account the considerable limitations and challenges mobile devices have to offer. Constraints in terms of graphics memory, processing power and memory cards render the output of most of these systems unusable on devices such as PDAs or smartphones because of excessive detail. At the same time, care needs to be taken in presenting this content on mobile devices since, for example, typical PDAs are only equipped with screens no larger than a 480x640 resolution display, thus presenting a further challenge. In this paper we propose both a modelling solution (*Virtual City Maker*) that can deliver urban content efficiently to mobile devices as well as a virtual navigation environment (*Virtual Navigator*) on a PDA to accompany it so that that the resulting meshes can be put to the test. It should be noted that while the content is specifically optimised for mobile use in mind, for example a varying low-polygon count and adjustable texture image sizes, the results need to be of the highest quality possible rivaling virtual city models seen on desktop machines but also accurately placed in space to accommodate real-world navigational needs.

2. VIRTUAL CITY MAKER SOLUTION

This section will describe in more detail the overall workflow for the Virtual City Maker modelling tool ranging from a description of the framework of the system to an overview of its *two modes*; the *automatic* and *semi-automatic* one with emphasis on some of the concepts, algorithms and interfaces behind them as well as some resulting 3D urban models.

2.1 OVERVIEW OF THE SYSTEM

First of all, regarding the input data used (a very important part of the process which affects the results in many ways) the authors of this paper have selected a combination of aerial photographs and also 2D ground maps, at least for the outline of the buildings modelled. This more hybrid approach combines the strengths of two different techniques for a more efficient result.

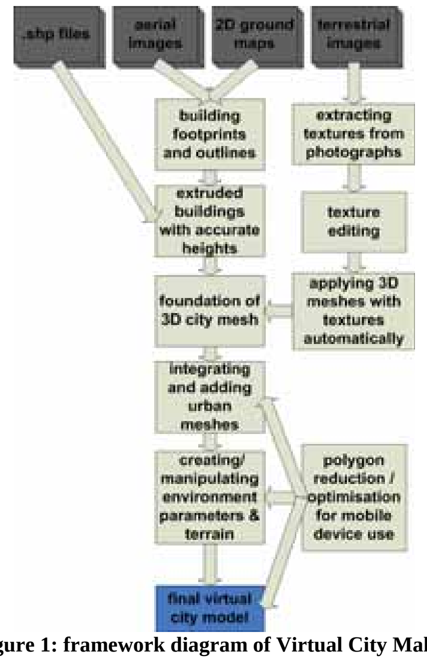

.shp files
aerial images
2D ground maps
terrestrial images
building footprints and outlines
extracting textures from photographs
extruded buildings with accurate heights
texture editing
foundation of 3D city mesh
applying 3D meshes with textures automatically
integrating and adding urban meshes
creating/ manipulating environment parameters & terrain
polygon reduction / optimisation for mobile device use
final virtual city model

Figure 1: framework diagram of Virtual City Maker

Aerial images ensure accurate results and the bypassing (to a certain extent) of a generalization process while ground maps provide positional and geographic information. Our work is not only focused on delivering content for mobile devices but also on navigation. Therefore, geo-referenced results and accuracy are key issues with errors in geometry placement only afforded to be marginal making this hybrid data use approach even more justified. Having placed building footprint outlines in a precise manner, these outlines have been extruded to real world heights using, if needed, GIS vector data in the form of the .shp file format. This particular file format operates as a combination of CAD data with added metadata attributes.

2.2 THE AUTOMATED MODE

The key method in the automated reconstruction mode is the detection of building outlines and roofs from aerial images and/or 2D ground maps. The second other important step is the extrusion to real-world heights and that is handled, when needed, by a parser developed to read the GIS .shp files. The philosophy behind this has been to make only those decisions in the 3D analysis stage of the detection process that can be made in as much assurance at each level illustrated as possible. Therefore the hypothesis generation process creates hypotheses that may have somewhat weak evidence, which is where information from additional input data comes into use. Following that, a selection process then picks these based on more global evidence, filtering out the unwanted ones. The hypothesis verification process uses the most of this global information and can therefore make better decisions, resulting in a more accurate output. To begin with, initial hypotheses are generated for 2D projections of building roofs. The reason behind that is that it is generally accepted that roofs are usually the most distinct features in aerial images from a nadir point of view (such as the ones our system uses) and that focusing on the roofs helps reduce the number of hypotheses to be examined later on [18]. The generation of all roof hypotheses is gathered by the restriction of the building shapes to rectangular form or similar compositions and also of roofs to be flat. Roofs of buildings of this shape should project into a parallelogram-like shape. To showcase a more detailed example of the process we followed here, as Lin [18] suggests, a $90°$ degree corner projected to an angle, o, given the angle π that one side of the projection makes with the horizontal and the viewing angles ψ and ξ is given by the following equation.

$$o = \pi \tan(\lambda, \omega) \text{ , where}$$

$$\lambda = [\cos^2(\pi+\psi)\cos(\xi)] + \frac{\sin^2(\pi+\psi)}{\cos\xi}$$

and

$$\omega = \sin(\pi+\psi)\cos(\pi+\psi)[\cos(\xi) - \frac{1}{\cos(\xi)}]$$

Figure 2: parallelogram projection algorithm used

Any parallelogram found must satisfy the relationships above. Following that, a selection process eliminates or keeps hypotheses based on what detectable evidence there is for them in the aerial image and on the global relationships amongst them all. The basis for the verification of 3D outlines in our work are the shadows buildings cast on the ground in aerial images. It is assumed that the direction of illumination in the image is known and also that the ground surface in the vicinity of the structure is flat. These two assurances allow for the computation of accurate height information. To offer more insight in the algorithms involved in this area, the building height is related to several parameters that can be measured from the image given. With X (pixels/meter) the image resolution in the neighborhood of the building location, H the projected wall height, can be computed (in pixels again) from the building height, B (in meters), and also the viewing angles by the following algorithm, $H = (B \cdot X \cdot \sin \xi)$. The projected shadow width, W, can be calculated from the building height, the viewing angles and the sun angles (the direction of illumination, x, the direction of shadow cast by a vertical line, y, and the sun incidence angle l using the following algorithm as introduced again by Lin [18]).

$$W = \begin{cases} B \cdot X \cdot \tan l \quad when\ \gamma = 0 \\ \dfrac{B \cdot X \cdot \sin(l+\xi)}{\cos i} \quad when \begin{cases} \xi \neq 0 \\ y = x = 270° - \psi \end{cases} \\ \dfrac{B \cdot X \cdot \sin(l-\xi)}{\cos i} \quad when \begin{cases} \xi \neq 0 \\ y = x = 90° - \psi \end{cases} \\ \dfrac{B \cdot X \cdot \sin(\xi - l)}{\cos i} \quad when \begin{cases} \xi \neq 0 \\ y = x + 180° \end{cases} \\ \dfrac{B \cdot X \cdot \sin \xi \cdot |\cos(y+\psi)|}{|\cos(y-x)|} \quad else \end{cases}$$

Figure 3: shadow width computation algorithm

However, when surfaces are not flat in the aerial image or are cast on surfaces of nearby buildings, the quality of the 3D information is low. While still usable for other purposes it was decided to have an alternative. Nevertheless then, in this eventuality, the average height value from the GIS .shp data is used. This illustrates in detail why our choice of using a number of different input data can be so useful. The concept behind shadow analysis used in this work derives from the establishment of relationships between shadow casting elements and shadows cast [17].

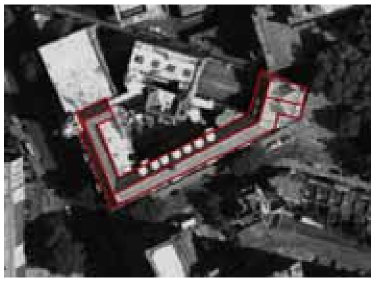


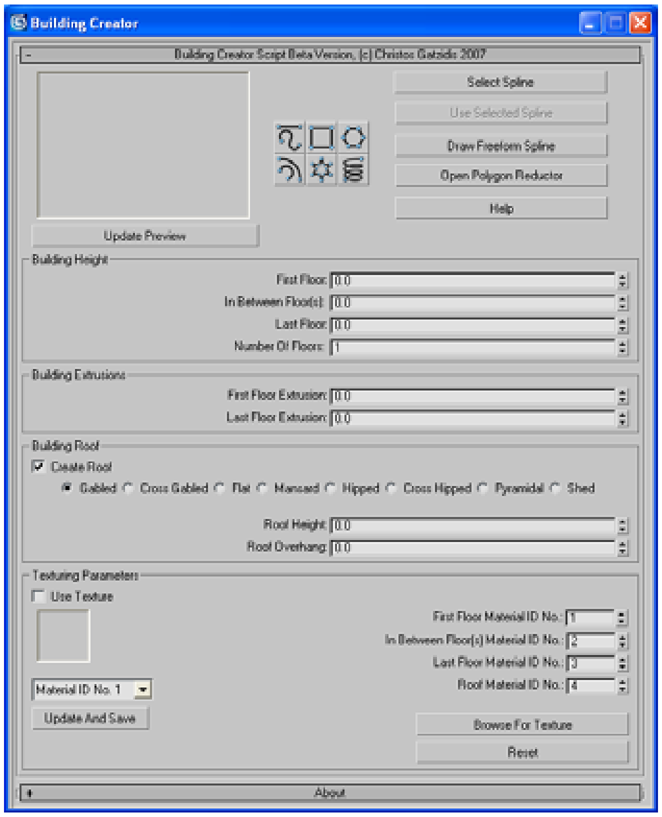
Figure 4: roof edge detection final results using method/algorithms

2.3 THE SEMI-AUTOMATED MODE

Often there are cases where users require urban content that is not derived from real-world GIS coordinates but can be fictitious (mobile gaming applications for example). The semi-automatic mode of the application presented here can cater for that via individual plug-ins.

2.3.1 CREATING BUILDINGS

Every virtual city consists of buildings, with these structures being the most important ones for the overall scene. One of the most important plug-ins for interactive or semi-automatic editing and creation of virtual cities, and thus buildings in particular, with the Virtual City Maker tool is the Building Creator script. Its interface is illustrated in Figure 5.



Building Creator Script Beta Version, (c) Christos Gatzidis 2007

Building Height
First Floor: 0.0
In Between Floor(s): 0.0
Last Floor: 0.0
Number Of Floors: 1

Building Extrusions
First Floor Extrusion: 0.0
Last Floor Extrusion: 0.0

Building Roof
[✓] Create Roof
[✓ Gabled] Cross Gabled Flat Mansard Hipped Cross Hipped Pyramidal Shed
Roof Height: 0.0
Roof Overhang: 0.0

Texturing Parameters
[ ] Use Texture
Material ID No. 1
First Floor Material ID No.: 1
In Between Floor(s) Material ID No.: 2
Last Floor Material ID No.: 3
Roof Material ID No.: 4

Figure 5: Building Creator script interface

The version presented here supports the creation of 3D buildings from splines which can either be imported from CAD files such as 2D ground plans or can be freeform ones that the user can draw. Multiple floor segmentation for the building is offered as well as roof modelling with controllable height and overhang based on the user's selection (out of a range of roof shapes such as gabled, cross-gabled, flat, mansard, hipped, cross hiped, pyramidal and shed), the ability to create insets for additional geometric detail (such as for example segmentation between floors or creating a ground surface surrounding the building or even a terrace-like shape before the roof) and also allowance for texturing. This final process has been implemented by means of assigning different numerical material IDs to each floor and roof so that when the user selects an image to map onto a surface it will be saved according to this ID selection. This speeds up and simplifies the otherwise tedious process of texturing while at the same time giving the user full control on it.

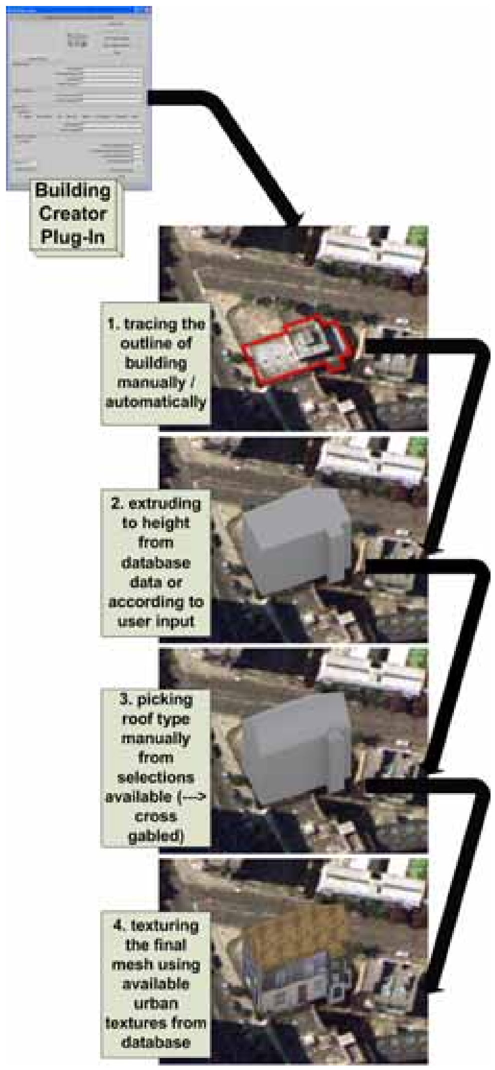

Building Creator Plug-In
1. tracing the outline of building manually / automatically
2. extruding to height from database data or according to user input
3. picking roof type manually from selections available (---> cross gabled)
4. texturing the final mesh using available urban textures from database

Figure 6: step-by-step operation of the Building Creator script, creating a random building from a real-world map

2.3.2 OPTIMISING BUILDING STRUCTURES

Following that, another very important component script for the overall tool is the previously mentioned Polygon Reductor. The algorithm which forms the basis of the Polygon Reductor script is based on a concept called *edge decimation* [15], [16]. Some concepts known as *vertex decimation* [14] and *triangle decimation* [13] are popular alternative approaches to polygon reduction / mesh simplification processes. In this case, i.e. with edge decimation, polygons are removed from the mesh by collapsing or contracting edges. That effectively means removing two triangles from an area of the mesh's surface thus simplifying it. The two vertices of the collapsed, *decimated*, edge are merged into one endpoint and the triangle adjacency lists of the two original vertices are concatenated for the newly formed vertex. It should be noted that this contraction process has been made both progressive (for performance reasons, more on that below) and also reversible. In other words, every time an edge is collapsed and then merged there needs to be enough data kept to split the concatenated list back into the lists for each of the two vertices in question. To achieve that goal it has to be guaranteed that collapsing and then rebuilding an edge is a process that needs to be general enough to work on any random area of each given mesh. There are meshes where not all edges have two adjacent triangles, essentially, meaning with edges that are not continuous in nature. Therefore, to cater for this eventuality, the algorithm used makes a check on all degenerated-bound edges for the two adjacent triangles needed. This tackles one of the most important drawbacks of the edge decimation technique; uncontrollable edge contraction. Edge merging algorithms can often implicitly alter the topology of a model by closing holes in the surface if the importance of all edges is not evaluated. In more detail, Figure 7 illustrates the algorithm we have opted to use which is the fundamental edge decimation one (as described by Ronfard [20], amongst others) both for preprocessing and visualizing the optimized mesh. This approach has the following advantages: a) progressive representations of the original building model B^n with the continuous family of meshes ($B^0, B^1, ..., B^{n-1}, B^n$) is very space-efficient plus has a smaller storage than the results of standard triangle/vertex approaches b) level-of-detail can be supported via the transformation from the B^i mesh to B^{i+1} by just applying the i-th vertex split/edge collapse operation c) the model can offer view-dependent or selective refinement.

```
Reconstruction pre-processing:

for (reach approximation index)
{
find and select edge whose decimation displays the smallest error measure;
collapse selected edge into one vertex;
store the collapse record also including one collapsed edge and two collapsed triangles;
}
return (base mesh & collapse records);

Visualization / reconstruction:

using the base mesh $B^0$ perform sequence of vertex split operations using the stored collapsed record;

produce a continuous family of meshes ( $B^0, B^1, ..., B^{n-1}, B^n$ );
```

Figure 7: progressive/reversible polygon reduction algorithm used

Finally, to select the target edges for the edge collapse we have opted to use an energy-derived error-based function introduced by Hoppe [21] which has yielded the best results for our urban meshes compared to other error metrics: $E(B) = E_{dist}(B) + E_{spring}(B) + E_{scalar}(B) + E_{disc}(B)$, where $E_{dist}(B)$ is the distance energy equal to the sum of squared distances from the points $X = \{x_1, ..., x_n\}$ to the mesh,

$E_{spring}(B)$ places on each edge of the mesh a spring of rest length zero and spring constant k, $E_{scalar}(B)$ measures the accuracy of its scalar attributes and $E_{disc}(B)$ measures the geometric accuracy of its discontinuity curves. There are two ways to decrease the polygon count on a building mesh using this plug-in. The first is automatically, i.e. by selecting one of the four different levels of detail presented with each progressively decreasing polygon count by 25%. The second is semi-automatically, by decreasing either the vertex or the polygon count/percentage. The resulting output can be previewed within the script, have its settings reset and also exported to the VRML file format for use with our visualization engine.

2.3.3 DIFFERENT TYPES OF SHADING

Literature suggests [11], [12] that other ways of rendering 3D urban content can also be beneficial for communicating elements of numerous other principles such as cognition, cartography and non-photorealism. Thus an additional plug-in has been incorporated to the Virtual City Maker solution called City Shader. This facilitates the production of cartoon-shaded (via stylistic shaded rendering), clay-rendered and wireframe visualizations with various parameters. Most importantly, these different types of shading can be exported to suitable file formats for mobile use which involves a number of issues considering mobile devices are not ideal to handle expressive visualizations. For example the clay-rendered model, before exported to VRML file format, had to have all its textures “baked” ensuring lighting and shadows information were kept because this particular file format does not support features such as advanced lighting or radiosity. The algorithm used for the results above is a variant of the one presented by Lake [19] which rather than smoothly interpolating shading across a model as in Gouraud shading (another popular approach), finds a transition boundary and shades each side of the boundary with a solid color.

<u>Pre-processing & visualizing a cartoon-shaded mesh</u>

```
compute the illuminated diffuse color for each material using $S_i = (c_g \times c_m) + [(c_m \times i_a) + (d_s \times d_m)]$ where $S_i$ is the vertex color, $c_g$ is the coefficient of global ambient light, $d_s$ is the diffuse coefficient of the light source and $c_m$ and $d_m$ are the ambient and diffuse coefficients of the object's material;
compute the shadowed diffuse color using $S_d = (c_g \times c_m) + (i_a \times c_m)$ where $i_a$ is the ambient coefficient of the light source ;
for (each material)
{
    create a one-dimensional texture map with two texels using the texture functionality
    store this one-dimensional texture map
    fill the texel (texture pixel) at the $i = 1$ end of the texture with $S_i$ and the texel at the $i = 0$ end of the texture with $S_d$ ;
}
compute the one-dimensional texture coordinate at each vertex of the model using $Max\{\overline{V} \cdot \overline{u}, 0\}$, where $\overline{V}$ is the normalized vector from the vertex to the light source location, $\overline{u}$ is the unit normal to the surface at the vertex and $\overline{V} \cdot \overline{u}$ is their vector inner product;
return the rendered mesh with lighting disabled and one-dimensional texture maps enabled;
```

Figure 8: pseudo-code of the cartoon-shading algorithm used

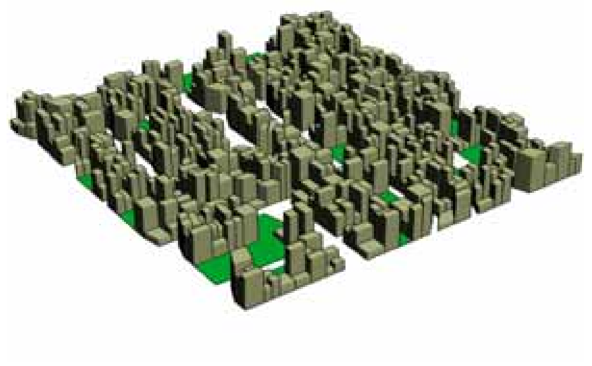


Figure 9: cartoon shaded urban mesh using City Shader

3. VIRTUAL NAVIGATION SOLUTION

Travel is the major component of navigation and usually involves unconscious cognition whereas wayfinding is the cognitive process of defining a path via an environment using and acquiring spatial knowledge based on both natural and virtual cues [6]. Although a number of mobile navigation systems have been designed, from industry and universities, they do not seem to have the appeal that was expected apart from the GPS in-car navigation systems. However, even these, have selected as the main visualisation environment to be the digital map which provides limited capabilities and they have now started to move into a rough 3D mesh visualisation which does not include photorealism. On the other hand, a few experimental mobile guides have been mainly developed by universities and are mostly based on abstract representations of the real environment without providing a robust solution for both intuitive navigation and wayfinding. In addition, they usually provide standard multimedia interfaces like audio or video operations without taking advantage of the capabilities of VR and user interface (UI) technologies. A brief overview of the most significant mobile VR applications has been recently documented [7], [8]. Part of the problem lies on the provision of meaningful spatial information in a realistic manner, so that inexperienced users do not have to put a great amount of cognitive effort to ‘decode’ the retrieved information (i.e. 3D map or wayfinding operations). Another significant concern relates to the lack of user-friendly graphical interfaces that will allow for the customisation of the information provided as well as the provision of different navigation tools in real-time. The emphasis on designing for continuous mobile interaction requires addressing a number of features [9]. To address these issues, a VR pedestrian mobile environment called Virtual Navigator has been implemented. Previous user-studies [7] indicated that the use of photorealism in 3D urban maps used for pedestrian navigation is helpful and more attractive than the standard 2D digital map representation. However, a few users reported some problems with the interaction techniques used, mainly because of the limited functionality the interface provided. Another interesting point [8] relates to the provision of choice to the user to accommodate sudden, external factors that may allow them to detour from a default path. In addition, it was positively suggested that the route line should be more distinct, minimising the probability of missing it while moving. To overcome these issues the graphical user interface (a prototype shown in Figure 10) for user-centred navigation and wayfinding was re-designed from scratch and is now divided into four categories including *file*, *routes*, *directions* and *tools*. The 'file' category contains the necessary functionality to open and close geo-referenced spatial maps represented in 3D. The virtual maps are currently stored in the device by using the wireless connectivity (GPRS or WiFi) so the 3D maps can be downloaded from a web-server. This allows Virtual Navigator to meet one of the most significant requirements of modern navigation systems which is to be operational anywhere and anytime. The 'exit' option permits to exit Virtual Navigation without the need of 'killing' the process from the memory control panel of the PDA. Next, the 'routes' category allows mobile users to select the type of representation they require for navigation and wayfinding operations. The effectiveness of the wayfinding depends on the number and quality of wayfinding cues provided to pedestrians [6]. Earlier user studies [7] indicated that users prefer different types of wayfinding aids like lines and arrows. In our work, this includes 'arrows', 'lines', 'hotspots' and 'guides'. Arrows, lines and hotspots have been used individually in previous mobile prototypes [7], [8] but not the guides. The purpose of using different categories of routes is to provide a meaningful aid that has a clear start and end, assisting pedestrians using Virtual Navigator, to find their way and not to get lost. The size and colours of the arrows and lines can be customised allowing for personalisation of the cues used. One of the points that were mentioned in previous user studies [7] is that the addition of recognisable landmarks would provide a clearer cognitive link between the VR environment and the real world scene. To address this effectively, first the 3D map was modelled using more detail such as including trees and lamps which are considered as landmarks from a large number of users. Besides, a number of hotspots that contain different types of functionality were added to the virtual 3D map making it interactive. In the simplest case, these include hyperlinks that link the 3D map with relevant web pages about the environment, but they can also provide links to other multimedia information such as digital pictures, audio and other 3D navigational information. The 'directions' category refers to the type of audio-visual information that can be provided to the pedestrians. The simultaneous presentation of audio-visual information meets one of the requirements (simultaneous activities that operate concurrently) of design issues in mobile interfaces. In the simplest case-scenario, textual directions provide information on how to navigate from one position to another. Additionally, textual annotations can be used effectively for presenting information about a place or a building (i.e. '…this is the main entrance of City University campus…'). Similarly, auditory directions perform the same action using pre-recorded wav files, but are prone to external noise. Finally, the 'tools' category, allows users to change some of the graphical and navigational properties of Virtual Navigator. These include the speed of navigation, the interaction type as well as the lighting of the scene. The speed of navigation can be customised according to the user needs accordingly (it was found that more experienced users prefer faster mode whereas inexperienced users like a slow pace). By increasing the altitude of the user location, through the 'interaction' menu, and altering the pitch to look directly down, it is possible to switch the view from a ground view into a bird's eye view of the surroundings. This is analogous to the standard map view and can be used for personal orientation and navigation [10].

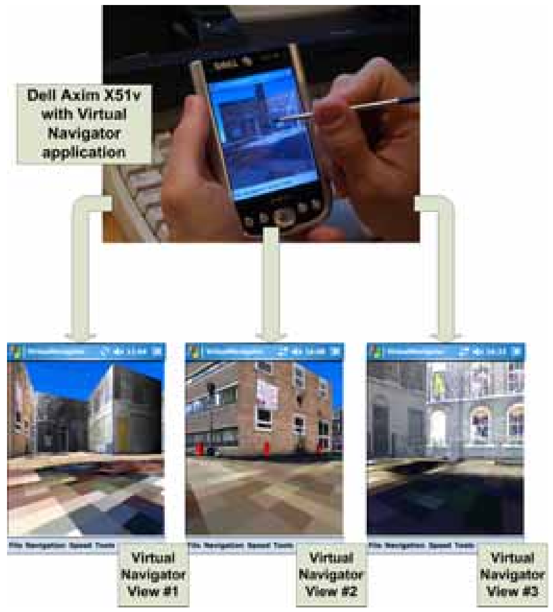

Dell Axim X51v with Virtual Navigator application
Virtual Navigator View #1
Virtual Navigator View #2
Virtual Navigator View #3

Figure 10: prototype interface design Virtual Navigator (manual mode)

4. CONCLUSION AND FUTURE WORK

Both on the modelling and VR techniques presented above work is continuous and on-going. Future work on the Virtual City Maker and the Virtual Navigator tools include: a) the introduction of a new plug-in that can intelligently distinguish between rural areas and more complex urban areas and offer the user with a more "in-between" stage of creating a collection of building meshes b) in the future, a database with a management system will be designed to hold and maintain all the necessary geo-referenced spatial data into a secure server which will be used in urban navigation. To serve the demands of current user-needs, our server should feed the client, through an optimised network with minimum latency in real-time performance and finally c) more advanced routing services will be also developed to provide advanced navigational assistance to mobile users by making searches to the remote database as well as considering the behaviour for different modes of transport.

ACKNOWLEDGEMENTS

Part of the work presented in this paper was conducted within the LOCUS project, funded by EPSRC, through the Location and Timing (KTN) Network. ALCATEL Lucent Telecom Limited is also thanked for their input and insight.

REFERENCES

[1] T. Strat, L. Quam, J. Mundy, R. Welty, W. Bremner, M. Horwedel, D. Hackett, A. Hoogs, The RADIUS Common Development Environment, *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, 1992, 215-226.

[2] J. Li, R. Nevatia, S. Noronha, User Assisted Modeling of Buildings from Aerial Images, *Proceedings of IEEE CVPR*, 1999, II:274-279.

[3] Y. Hsieh, SiteCity: A Semi-Automated Site Modeling System, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, 1996, 499-506.

[4] A. Gruen, H. Dan, A Topology Builder for Automated Building Model Generation, *Proceedings of Automatic Extraction of Man Made Objects from Aerial and Space Images (II)*, Birkhauser, Basel, 1997, 149-160.

[5] E. Gülch, H. Müller, T. Läbe, Integration of Automatic Processes Into Semi-Automatic Building Extraction, *Proceedings of ISPRS Conference Automatic Extraction Of GIS Objects From Digital Imagery*, Munich, Germany, 1999.

[6] D. Downman, E. Kruijff, J. LaViola, I. Poupyrev, *3D user interfaces: theory and practice* (Boston, Addison Wesley, 183-254, 2005).

[7] C. Gatzidis, F. Liarokapis, V. Brujic-Okretic, Automatic modelling, generation and visualisation of realistic 3D virtual cities for mobile navigation, *Proceedings of the 9th International Conference on Virtual Reality*, Laval, France, 2007, 225-234.

[8] F. Liarokapis, V. Brujic-Okretic, S. Papakonstantinou, Exploring urban environments using virtual and augmented reality, *Journal of Virtual Reality and Broadcasting*, Digital Peer Publishing, *3*(5), 2006, 1-13.

[9] A. Dix, J. Finlay, G. Abowd, R. Beale, *Human-computer interaction* (Harlow, Prentice Hall, 2004).

[10] D. Mountain, F. Liarokapis, Interacting with virtual reality scenes on mobile devices, *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices & Services*, Salzburg, Austria, 2005, 331-332.

[11] J. Schumann, T. Strothotte, S. Laser, A. Raab, Assessing the effect of non-photorealistic rendered images in CAD, *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, Vancouver, British Columbia, Canada, 1996, 35-41.

[12] J. Dollner, M. Walther, Real-Time Expressive Rendering of City Models, *Proceedings of the Seventh International Conference on Information Visualization (IV'03)*, Seattle, WA, USA, 2003, 245-250.

[13] W. J. Schroeder, J. A. Zarge, W. E. Lorensen, Decimation of triangle meshes, *Proceedings of SIGGRAPH '92: annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1992, 65-70.

[14] B. Koh, T. Chen, Progressive VRML Browser, *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, Copenhagen, Denmark, 1999, 71-76.

[15] M. Garland, P. Heckbert, Surface Simplification Using Quadric Error Metrics, *Proceedings of SIGGRAPH '97: annual conference on Computer graphics and interactive techniques*, Los Angeles, USA, 1997, 209-216.

[16] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Mesh optimization, *Proceedings of SIGGRAPH '93: annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1993, 19-26.

[17] C. Lin, A. Huertas, R. Nevatia, Detection of Buildings using Perceptual Grouping and Shadows, *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, Seattle, WA, USA, 1994, 62-69.

[18] C. Lin, R. Nevatia, Building Detection and Description from a Single Intensity Image, *Computer Vision And Image Understanding Journal*, *72*(2), 1998, 101-121.

[19] A. Lake, C. Marshall, M. Harris, M. Blackstein, Stylized rendering techniques for scalable real-time 3d animation, *Proceedings of NPAR 2000: First International Symposium on Non Photorealistic Animation and Rendering*, Annecy, France, 2000, 13-20.

[20] R. Ronfard, J. Rossignac, Full-range approximation of triangulated polyhedra, *Computer Graphics Forum Journal*, *15*(3), 1996, 67-76.

[21] H. Hoppe, Progressive Meshes, *Proceedings of SIGGRAPH '96: annual conference on Computer graphics and interactive techniques*, New Orleans, Lousiana, USA, 1996, 99-108.