Abstract

Rapid procedural 3D urban modelling has been a constant research issue for a number of years now. We present a complete procedural 3D modelling solution for mobile device usage of the content, 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 content produced 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 supporting the import and display of various navigational file formats equipped with a comprehensive front-end user-friendly graphical user interface providing immersive virtual 3D navigation to a wide range of users.

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.

It has been accepted that fully-automatic approaches are categorized according to the input information used. The eldest and still an extremely popular data source for automatic virtual city generation are aerial images ([1], [2], [24]). Some disadvantages of aerial images include the fact that analyzing information of this low level in dense urban contexts can be complex but it has been accepted that they can be a very rich source of various kinds of information. Their main disadvantage listed above can be bypassed by using other special material in combination with them (additional height models, multiple overlap or colour images being some of this material).

Automatic city generation has also been achieved via the use of laser scanning data [3], [4]. This data, unlike aerial images, has the ability to represent the geometry of the surface directly. It has to be noted that the cost involved in acquiring this data is far greater than the one associated with aerial images, often almost prohibitive. However, technological advances since this type of data was first introduced have improved significantly with the cost decreasing progressively. DSMs (laser digital surface models) can offer very reliable information for segmentation and parametric reconstruction because of their nature. Unfortunately, they can also suffer from low density of measured points plus excessive automatic interpretation.

The last method for automatic urban content creation, and one that has been increasingly popular of late, is the use of 2D ground maps/plans [5], [6] from Geographical Information Systems (GIS). These can be very cost-effective in acquisition (often readily available) and are often used not just on their own but in conjunction with one of the previous data sources resulting in a more hybrid approach that can yield more accurate results. 2D ground maps can be an extremely high-level information source, much more so than laser scanning data or aerial images, however completeness and varying quality of the 2D ground maps can often pose inconsistency problems in the automatic reconstruction process.

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 [7]. 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 [8] 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 [9]. Yet another approach suggests an automatic system constructing topological relations amongst 3D roof points collected by a user for each roof. This method [10] 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 [11]. 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 the resulting meshes can be put to the test. It should be noted that while the content is specifically optimized 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. 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. Some of these attributes include three different heights for each building (from which our system derives the average value, again when the height cannot be derived from shadow detection, more on that on the next section) plus additional useful data.
such as an individual ID for each building, its age, shape, type and others.

To conclude the 3D urban creation process terrestrial photographs are also essential to construct a photorealistic 3D virtual city model. These terrestrial photographs are captured manually from different viewpoints with the use of a conventional digital camera photographic technique.

All tasks or methods of the solution are tackled by a unified system with one user-interface (automatic mode). They can also fall into different categories of work which means that they can also be resolved by individual modelling plug-ins (semi-automatic mode) that have been designed to also work as stand-alone applications in their own right.

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. Figure 1 illustrates the main processes involved in the building and roof detection system.

![Figure 1. Roof/building detection block diagram.](image)

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 [25]. 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 [25] suggests, a 90° degree corner projected to an angle, \( \phi \), given the angle \( \pi \) that one side of the projection makes with the horizontal and the viewing angles \( \psi \) and \( \xi \) is given by the following equation.

\[
\phi = \pi \tan(\lambda, \omega), \text{ where } \\
\lambda = \left[ \cos^2(\pi + \psi) \cos(\xi) \right] + \frac{\sin^2(\pi + \psi)}{\cos \xi} \\
\text{and} \\
\omega = \sin(\pi + \psi) \cos(\pi + \psi)[\cos(\xi) - \frac{1}{\cos(\xi)}]
\]

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 \( \psi \)). The following algorithm as introduced again by Lin [25]).

\[
W = \begin{cases} 
B \cdot X \cdot \tan(\gamma) & \text{when } \gamma = 0 \\
B \cdot X \cdot \sin(l + \xi) \cos(\xi) & \text{when } \xi \neq 0, \ \gamma = x = 270^\circ - \psi \\
B \cdot X \cdot \sin(l - \xi) \cos(\xi) & \text{when } \xi \neq 0, \ \gamma = x = 90^\circ - \psi \\
B \cdot X \cdot \sin(\xi - l) \cos(\xi) & \text{when } \xi \neq 0, \ \gamma = x = 180^\circ \\
B \cdot X \cdot \sin(\xi) \cos(y + \psi) & \text{else} \\
\end{cases}
\]

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 [23].

Figure 4. Detecting shadow evidence on image.

Provided the direction of lighting is known, an attempt is made to establish these relationships between shadow casting lines and the actual shadow lines and subsequently between shadow casting vertices or junctions and shadow vertices (illustrated in Figure 4). The ambiguous / non-visible vertical edges of buildings cast visible shadows in a direction parallel to the projection of the illumination on the ground. These shadow lines start at the edge of a building and can be detected fairly easily. There are occasions when these lines are cast in part on the ground and in part on nearby buildings or structures but even then the projection constraint can be justified and they can be detected.

2.3. The Interactive / Semi-Automated Mode

The interactive mode or user-assisted mode of Virtual City Maker has been developed to take advantage of the fact that operator input can not only enhance the automatically-produced results and improve them but also diversify the appeal of the system itself. 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 also cater for that too. In this section, some of the key individual plug-ins of the system that undertake these tasks will be presented.

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.

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 hipped, pyramidal and shed, all of these illustrated in Figure 6), 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.

Some other features included are preview windows within the script's interface both for the texture and the model to assist the user with their modelling progress. Also linkage with another plug-in developed, called the Polygon Reductor and described below, is provided for the decreasing of mesh geometrical detail. It is worth-mentioning that a help function is also included for first time users.

2.3.2. Optimising Building Structures. Following that, another very important component script for the overall tool is the previously mentioned Polygon Reductor. This script offers the ability of reducing the polygon or face count for one or multiple buildings already generated while retaining its texture coordinates and also georeferenced positioning (the interface of the plug-in is shown in Figure 9). The algorithm which forms the basis of the Polygon Reductor script is based on a concept called edge decimation [21], [22]. Some concepts known as vertex decimation [20] and triangle decimation [19] 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 (Figure 8) 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 10 illustrates the algorithm we have opted to use which is the fundamental edge decimation one (as described by Ronfard [31], 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 \( \{ B^0, B^1, \ldots, B^n \} \) 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 \( B^1 \) 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:**

```plaintext
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:**

```plaintext
using the base mesh perform sequence of vertex split operations using the stored collapsed record;
produce a continuous family of meshes \( \{ B^0, B^1, \ldots, B^n \} \);
```

Finally, to select the target edges for the edge collapse function introduced by Hoppe [29] which has yielded the best results for our urban meshes compared to other error metrics:

\[
E(B) = E_{\text{disc}}(B) + E_{\text{spring}}(B) + E_{\text{scalar}}(B) + E_{\text{disc}}(B)
\]

where \( E_{\text{disc}}(B) \) is the distance energy equal to the sum of squared distances from the points \( X = \{ x_1, \ldots, x_n \} \) to the mesh, \( E_{\text{spring}}(B) \) places on each edge of the mesh a spring of rest length zero and spring constant \( k \), \( E_{\text{scalar}}(B) \) measures the accuracy of its scalar attributes and \( E_{\text{disc}}(B) \) 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.

![Figure 10. Progressive polygon red. algorithm used.](image)

![Figure 11. Reducing poly count with Polygon Reductor.](image)

As an example of how crucial it is to reduce the polygon count of a building and also the difference it can make in achieving an acceptable frame rate in real-time visualization, a London building (illustrated in Figure 11) has been generated automatically with our method and is 492 polygons in total. Despite the fact that it is not prohibitively large, trying to visualize it in real-time with tens of similar geometrical structures on a mobile device with a graphics card barely capable of 3D acceleration is impossible. After using the plug-in at hand the face count of this model has been reduced to 244 and finally to 56 polygons. It should be noted here...
that while polygon count has decreased to almost a tenth of the original amount, the building’s main defining vertices have not been altered. This results in a model with not only the same boundaries as the original but also the same geographical shape.

2.3.3. Different Types Of Shading. Literature suggests [17], [18] 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 some of the results below is a variant of the one presented by Lake [30] 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. The pseudo-code for this process is presented in Figure 13.

Pre-processing & visualizing a cartoon-shaded mesh
compute the illuminated diffuse color for each material using
\[
S_i = (c_g \times c_m) + [(c_m \times \ell_g) + (d_m \times d_m)]
\]
where \(S_i\) is the vertex color, \(c_g\) is the coefficient of global ambient light, \(d_m\) 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
\[
S_d = (c_g \times c_m) + (\ell_a \times c_m)
\]
where \(\ell_a\) 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 \(i=1\) end of the texture with \(S_d\);}
compute the one-dimensional texture coordinate at each vertex of the model using \(\text{Max} \{\bar{\mathbf{P}} \cdot \hat{\mathbf{n}}, 0\}\), where \(\bar{\mathbf{P}}\) is the normalized vector from the vertex to the light source location, \(\hat{\mathbf{n}}\) is the unit normal to the surface at the vertex and \(\bar{\mathbf{P}} \cdot \hat{\mathbf{n}}\) is their vector inner product;
return the rendered mesh with lighting disabled and one-dimensional texture maps enabled;
Figure 13. Pseudo-code for cartoon-shading algorithm.

2.3.4. Enhancing The Urban Environment. The impact of a virtual city both on a navigational and also on an aesthetical level can be enhanced by surrounding world objects, largely demonstrated by the research work in this area ([26],[27],[28]).

The main drawback of using these is that it can be very time consuming to create and place them on a fairly large scale urban model.

A solution to this, tackling foliage meshes, is the Tree Maker plug-in (user-interface illustrated in Figure 14). The Tree Maker script is in many ways representative of all the other work conducted on generation of models other than buildings for the Virtual City Maker solution. It targets performance and also automation.

All tree models generated consist of planes with adjustable height and width. These planes have opacity based bitmaps that create the illusion of a 3D mesh via transparency. They can have 2, 4, 6 or 8 sides, the more sides the better the illusion of a 3D mesh is, with a cost to real-time visualization performance.

An extensive texture library offers the user selection of plants in order to cover a number of different tree types. The user can also select any plane from the virtual city scene as a ground the models will rest upon.

Figure 12. Some expressive urban renders using City Shader.
and then generate rows (or columns of trees) in both the x and y axis (with controllable distances).

Figure 14 illustrates some types of trees generated by the Tree Maker script and also how they can be placed into an urban scene, all developed by the rest of our solution.

2.3.5. Other Plug-Ins. Other plug-ins developed, include tools for the generation of terrain meshes (Terrain Generator) encompassing the urban areas and also tools for producing an environment for the scene (Environment Maker) with a minimum amount of effort from the side of the user and with fully-controllable parameters. The terrain can be created with a different number of segments to provide different LODs (levels of detail) while mountain and hill-like shapes can be created and instantiated with appropriate user input (positioning, size and noise).

As for generating the environment for the urban mesh some of the parameters supported include adjusting camera position and sunlight parameters such as lighting according to the time of the day, the shadows as well as the sun system orientation. Effects such as self-illumination, clouds (with user-adjusted size, quality and density) and finally a fog system can also be added.

3. Virtual Navigator 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 [12]. 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 [13], [14].

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 the following features [15]: have a clear start and end; allowing for interaction at any time from the user; simultaneous activities that operate concurrently; time characterisation of pedestrians and device and associative models of information. To address these issues, a VR pedestrian mobile environment (which was originally developed at City University and now is under continuous development at Coventry University) called Virtual Navigator has been implemented.

Previous user-studies [13] 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 [14] 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 15) for user-centred navigation and wayfinding was redesigned 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 wayfinding depends on the number and quality of wayfinding cues provided to pedestrians [12]. Earlier user studies [14] 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 [13],[14] 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 [14] 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. In our work 'guides' are virtual representation of humans, also known as avatars. Their main objective within Virtual Navigator is to guide pedestrians towards the requested destination. Currently, we have designed only one avatar, with male characteristics, which is activated as soon as the user clicks on him. Subsequently, the avatar starts walking with a slow pace, guiding this way the user, from one of the entrances of City University, London (St John Street) to the main entrance (Northampton Square). 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 [16].
Finally, we have performed a brief experiment to test the Virtual Navigator's real-time performance on a number of PDAs. Results (Figure 16) show that only the PDA (Dell Axim X51v) with a 3D accelerated graphics card can yield relatively high results in the FPS (frames per second) area (37.01 FPS being the highest value). The other three devices performed on the same level with negligible variations since our application switches to software rendering whenever it does not detect a hardware solution (19.60 to 19.62 FPS). In the future, a database with a management system will be designed to hold and maintain all the necessary georeferenced 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. This hybrid method both in terms of input used and in modes results in virtual models of the highest photorealistic standards (fully-textured) and optimized (reduced number of polygons and texture image resolutions and with a number of different LODs). Apart from an accurate recreation and placement of building (including roof structures), other features include the semi-automatic control of supplementary parameters such as terrain, lighting (time of day) and environment (ground, fog, sunlit weather etc.) and also foliage. Furthermore, different types of shading are supported (wireframe, cartoon-rendered, clay-rendered etc.). The 3D urban areas produced have been imported in a mobile VR navigation environment, called Virtual Navigator, specifically created for the purposes of pedestrian navigation, wayfinding and exploration. This engine, operating on a PDA, is capable of reading different navigational file formats and includes a front-end user-centered interface for fully immersive virtual 3D navigation by manipulating 6 degrees of freedom such as observer location in three dimensions (x, y, z) plus orientation (yaw, roll and pitch). Both on the modelling and VR techniques presented above work is continuous and ongoing. Future work includes: a) the introduction of a plug-in, which via access to a library of different meshes can interactively enrich the virtual city scene with a number of objects (such as benches, pavements, street lamps etc.) further enhancing the scene considerably with user-assisted input b) the introduction of a new plug-in that can intelligently distinguish between rural areas and more complex urban areas etc. and offer the user with a more “in-between” stage of creating a collection of building meshes c) the introduction of a new plug-in that can semi-automatically model landmarks according to their type (statues, cathedrals etc.) d) integrating semantic details to buildings so that interior details can be modelled too, such as inner walls to provide for indoor navigation e) conduct experiments with the resulting models, the engine and a number of human subjects in order to gain insight on the issues of efficient modelling for mobile navigation and assess some of the usability issues of the interface.

Figure 16. Virtual Navigator real-time performance.

4. Conclusions And Future Work

While extensive work has been carried out in the methods and approaches behind automatic and semi-automatic creation of 3D virtual cities, to this day no comprehensive solution exists that can provide a complete pipeline from creating photorealistic georeferenced urban content to offering user-assisted enhancement and producing output specifically targeting mobile devices with all the challenges they have to offer. In this paper a novel 3D modelling solution has been presented consisting of a collection of plug-ins, called Virtual City Maker. The approach operates in two modes (automatic and manual) and is reliant upon using high-definition aerial and terrestrial photographic input as well as 2D ground maps and GIS data combined with a user-friendly customized interface. This hybrid method both in terms of input used and in modes results in virtual models of the highest photorealistic standards (fully-textured) and optimized (reduced number of polygons and texture image resolutions and with a number of different LODs). Apart from an accurate recreation and placement of building (including roof structures), other features include the semi-automatic control of supplementary parameters such as terrain, lighting (time of day) and environment (ground, fog, sunlit weather etc.) and also foliage. Furthermore, different types of shading are supported (wireframe, cartoon-rendered, clay-rendered etc.). The 3D urban areas produced have been imported in a mobile VR navigation environment, called Virtual Navigator, specifically created for the purposes of pedestrian navigation, wayfinding and exploration. This engine, operating on a PDA, is capable of reading different navigational file formats and includes a front-end user-centered interface for fully immersive virtual 3D navigation by manipulating 6 degrees of freedom such as observer location in three dimensions (x, y, z) plus orientation (yaw, roll and pitch). Both on the modelling and VR techniques presented above work is continuous and ongoing. Future work includes: a) the introduction of a plug-in, which via access to a library of different meshes can interactively enrich the virtual city scene with a number of objects (such as benches, pavements, street lamps etc.) further enhancing the scene considerably with user-assisted input b) the introduction of a new plug-in that can intelligently distinguish between rural areas and more complex urban areas etc. and offer the user with a more “in-between” stage of creating a collection of building meshes c) the introduction of a new plug-in that can semi-automatically model landmarks according to their type (statues, cathedrals etc.) d) integrating semantic details to buildings so that interior details can be modelled too, such as inner walls to provide for indoor navigation e) conduct experiments with the resulting models, the engine and a number of human subjects in order to gain insight on the issues of efficient modelling for mobile navigation and assess some of the usability issues of the interface.
5. Acknowledgements
Work presented in this paper was funded by ALCATEL Lucent Telecom UK Limited.

6. References


