Dynamic Worlds in Miniature

Ramón Trueba Carlos Andújar

MOVING Group, Universitat Politècnica de Catalunya
{rtrueba, andujar}@lsi.upc.edu

Abstract. The World in Miniature (WIM) metaphor allows users to interact and travel efficiently in virtual environments. In addition to the first-person perspective offered by typical VR applications, the WIM offers a second dynamic viewpoint through a hand-held miniature copy of the virtual environment. In the original WIM paper the miniature was a scaled down replica of the whole environment, thus limiting the technique to simple models being manipulated at a single level of scale. Several WIM extensions have been proposed where the replica shows only a part of the virtual environment. In this paper we present an improved visualization of WIM that supports arbitrarily-complex, densely-occluded scenes. In particular, we discuss algorithms for selecting the region of the virtual environment which will be covered by the miniature copy and efficient algorithms for handling 3D occlusion from an exocentric viewpoint.

1 Introduction

The World in Miniature (WIM) metaphor [16] complements the first-person perspective offered by typical VR applications with a second dynamic view of a miniature copy of the virtual world. This second exocentric view of the world helps users to understand the spatial relationships of the objects and themselves inside the virtual world. Furthermore, because the WIM is hand-held, it can be quickly explored from different viewpoints without destroying the point of view established in the larger, immersive point of view.

The WIM has been used as a unifying metaphor to accomplish many user tasks including object selection and manipulation, navigation and path planning. Object selection can be accomplished either by pointing directly at the object or by pointing at its proxy on the WIM. By rotating the hand-held replica, users can even view and pick objects that are obscured from the immersive camera viewpoint. Once selected, objects can be manipulated either at the scale offered by the WIM or at the one-to-one scale offered by the immersive environment. The WIM can also include an avatar of the user that can be moved to change the user location in the environment, providing camera control from an exocentric point of view. The WIM also supports path planning by selecting a sequence of 3D waypoints on the replica. Furthermore, since the WIM provides the user with an additional scale at which to operate, the user can choose the most appropriate scale for any given navigation or manipulation task.

Fig. 1. Two examples of our improved WIM visualization. The miniature replica provides a cut-away view of the part of the model where the user is located

Any WIM implementation has to address two key problems. On one hand, one must decide which region of the environment should be included in the miniature copy. Early implementations put a replica of the whole environment in the miniature, thus limiting its application to simple models like a single room. Some extensions of the WIM to handle more complex models have been proposed [3,13,18]. These approaches use a miniature copy which includes only those objects in the user's region of interest, which can be scaled and moved either manually or automatically. These techniques allow the accomplishment of user tasks at different levels of scale. However, current approaches either provide ad-hoc solutions valid for a limited class of models, or assume the input scene already provides some information about its logical structure, such as the subdivision of a building into floors.

On the other hand, WIM implementations have to provide some way to handle 3D occlusion. Regardless of the size and shape of the region included in the miniature copy, bounding geometry such as walls must be conveniently culled away to make interior objects visible. Early implementations relied solely on backface culling techniques [16], but these are suitable only for very simple models; general 3D models require the application of more sophisticated techniques for handling 3D occlusion such as cut-away views [5].

In this paper we present an improved visualization of WIM that supports arbitrarily-complex, densely-occluded scenes by addressing the two problems stated above. First, we propose to select the region to be included in the miniature copy using a semantic subdivision of the scene into logical structures such as rooms, floors and buildings computed automatically from the input scene during preprocessing. The rationale here is that matching the miniature copies with logical entities of the environment will help the user to accomplish interaction tasks on the WIM. A logical decomposition will provide additional cues to better understand the spatial relationships among parts of the scene, in addition to a more intuitive and clear view of the nearby environment suitable for precise object selection and manipulation. Second, we pro-

pose a simple algorithm to provide a cut-away view of the miniature copy, so that interior geometry is not obscured by the enclosing geometry (Figure 1).

The problem of computing a spatial decomposition of the scene into logical entities is closely related with the cell-and-portal graphs used for occlusion culling (see [4] for a recent survey). Cells correspond to regions with approximately constant visibility and thus often match with logical entities of the model such as rooms. Note that portal culling requires the computation of the geometry of each portal, which is a complex problem in general 3D scenes. Fortunately, our decomposition scheme only requires an approximate description of the cell's boundary. Our algorithm for building a cell-and-portal decomposition is based on a distance field computed on a grid [2].

The main contributions of the paper to the WIM metaphor are (a) an algorithm to automatically compute which region of the scene must be included in the miniature copy which takes into account a logical decomposition of the scene, and (b) an algorithm for providing a cut-away view of the selected region so that interior geometry is revealed. These improvements expand the application of WIM to arbitrarily-complex, densely-occluded scenes, and allows for the accomplishment of interaction tasks at different levels of scales.

The rest of this paper is organized as follows. Section 2 reviews related work on WIM extensions, cell decomposition and 3D occlusion. Section 3 provides an overview of the three main components of our extension to the WIM, which are detailed in Sections 4-6. Some preliminary results are discussed in Section 7. Finally, we provide concluding remarks and future work in Section 8.

2 Previous Work

In this section we briefly review previous work related with the problem being addressed (enhancements to the WIM metaphor) and to our adopted solutions (cell decompositions and 3D occlusion management).

World in Miniature. The WIM was proposed originally by Stoakley, Conway and Pausch [16] who discussed their application to object selection, manipulation and travelling. They found that users easily understood the mapping between virtual world objects and the proxy WIM objects. User orientation during WIM navigation was reduced by flying into the WIM [14]. Unfortunately, using a replica of the whole environment in the miniature limits its application to simple models like a single room, due to occlusion and level-of-scale problems. Several WIM extensions have been proposed to overcome this limitations. The STEP WIM proposed by La Viola et al. [13], puts the miniature replica on the floor screen so that it can be interacted with the feet. The main advantage is the freeing of the hands for other tasks. The method provides several methods for panning and scaling the part of the scene covered by the miniature. This technique has been used in multiple projects although its effectiveness has not been evaluated formally. The Scaled and Scrolling WIM (SSWIM) [18] supports interaction at multiple levels of scale trough scaling and scrolling functions. SSWIM adds functionally and hence complexity because the user has to scale the model manually. However, the scrolling is automatic when the user moves to a position out-

side of a dead zone. This dead zone is a box centered at the SSWIM, but no information is given about how this box should be computed. Chittaro *et al*. [3] propose an extension of the WIM that supports user navigation in virtual buildings as a navigation aid, but also provides a means of examining any floor of a virtual building without having necessary to navigate to it. The aid provides information about both the internal and external structure of a building, but it is necessary to identify manually all the polygons on the different floors, which can be a time-consuming task on complex models. Unlike previous WIM extensions, we select the region covered by the miniature using an automatically-computed decomposition of the model into cells.

Cell and portal decompositions. A cell-and-portal graph (CPG) is a structure that encodes the visibility of the scene, where nodes are cells, usually rooms, and edges are portals which represent the openings (doors or windows) that connect the cells. There are few papers that refer to the automatic determination of CPGs, and most of them work under important restrictions [4]. Teller and Séquin [17] have developed a visibility preprocessing suitable for axis-aligned architectural models. Hong et al. [10] take advantage of the tubular nature of the colon to automatically build a cell graph by using a simple subdivision method based on the center-line (or skeleton) of the colon. To determine the center-line, they use the distance field from the colonic surface. Very few works provide a solution for general, arbitrarily-oriented models with complex, non-planar walls. Notable exceptions are those approaches based on a distance-field representation of the scene [2,9]. Our approach for cell detection also relies on a distance field and it is based on the scene decomposition presented in [2].

Management of 3D occlusion. Complex geometric models composed of many distinct parts and structures arise in many different applications such as architecture, engineering and industrial manufacturing. Three-dimensional occlusion management techniques are often essential for helping viewers understand the spatial relationships between the constituent parts that make up these datasets. Elmqvist and Tsigas [6] analyze a broad range of techniques for occlusion management and identify five main patterns: *Multiple Viewports* (using two or more separate views of the scene), *Virtual X-Ray*, *Tour Planners* (a precomputed viewpoint path reveals the otherwise occluded geometry), *Interactive Exploders* (adopted for the WIM in [3] through a floor sliding mechanism) and *Projection Distorter* (where nonlinear projections are used to integrate two or more views into a single view).

Virtual X-Ray techniques are particularly relevant to our approach. These techniques make the targets visible through intervening distractors (occluders) by turning occluding surfaces invisible or semi-transparent. Several methods for distractor removal have been proposed. Some techniques are view-dependent (break-away views) whereas others are static (cut-away views); some eliminate distractors (or parts of distractors) while others merely make distractors semi-transparent.

We have adopted the cut-away approach to reveal the interior cell objects. Diepstraten et al. [5] describe a number of algorithms for generating cut-away views. The proposed algorithms are both efficient and easy to implement, although two important assumptions are made: they assume that the classification of objects as interior or exterior is provided by an outside mechanism, and that the cut-out geometry is convex. Fortunately, our decomposition of the scene into cells with approximately constant visibility allows the application of very simple algorithms to generate automatically cut-away views.

3 Overview

We aim at improving the WIM metaphor by addressing three problems:

WIM selection: given the current viewpoint location, we compute which region of the scene must be included in the miniature copy and thus presented to the user as a hand-held miniature. We use a polyhedron to represent the region of the space to be included in the miniature.

WIM clipping: once the region to be included has been computed, we must render only the scene geometry inside this region.

WIM revealing: we provide a cut-away view of the selected region so that interior geometry is revealed.

The algorithm for *WIM selection* proceeds through two main preprocessing steps (described in detail in Section 4):

- Cell detection: the scene is discretized into a voxelization and then decomposed into a collection of cells corresponding to logical entities; each voxel is assigned a cell ID, adjacent cells are detected and their connectivity is stored in a cell-and-portal graph.
- Region extraction: for each cell, we extract a polygonal surface which approximately encloses the cell.

Since cells are detected during preprocessing, *WIM clipping* can be done either during preprocessing or at runtime. In the former case, we could clip the scene geometry to each cell and store the resulting geometry. However, this approach would add complexity to the integration with already existing applications using their own scene graph. Therefore, we have adopted a runtime approach for clipping the scene geometry to a polyhedral region. Our solution (described in Section 5) can be seen as a particular case of CSG rendering (we must render the intersection of the scene with the polygonal region) which can be implemented easily on a GPU [12].

Finally, WIM revealing is based on cut-away views. An important benefit of using cells with approximately constant visibility is that they greatly simplify the WIM revealing problem: removing the enclosing geometry will be enough to reveal interior objects. Note that the presence of interior walls inside the miniature would complicate significantly the algorithm for providing cut-away views. We describe a simple solution (Section 6) with no extra cost which gives good results on the class of cells produced by our decomposition algorithm.

4. WIM Selection

4.1 Automatic cell detection

Our approach for computing the cell-and-portal graph consists of first building a binary grid separating empty voxels from non-empty ones, next approximating the distance field using a matrix-based distance transform. Then we start an iterative conquering process starting from the voxel having the maximum distance among the remaining voxels. During this process, all conquered voxels are assigned to the same cell ID. Lastly, a final merge step eliminates small cells produced by geometric noise.

4.1.1 Distance field computation

The first step converts the input model into a voxel representation encoded as a 3D array of real values. Voxels traversed by the boundary of the scene objects are assigned a zero value whereas empty voxels are assigned a +1 value. This conversion can be achieved either by a GPU-based rasterization of the input model or by a simultaneous space subdivision and clipping process supported by an intermediate octree. The next step involves the computation of a distance field (Figure 2-middle). Distance fields have been used in generating cell-and-portal graph decompositions [2,9]. Distance fields can be computed in a variety of ways (for a survey see [11]).

4.1.2 Cell generation

The cell decomposition algorithm visits each voxel of the distance field and replaces its unsigned distance value by a cell ID. We use negative values for cell ID's to distinguish visited voxels from unvisited ones. The order in which voxels are visited is key as it completely determines the shape and location of the resulting cells. The order we propose for labeling cells relies on a conquering process starting from the voxel having the maximum distance among the remaining unvisited voxels. This local maximum initiates a new cell whose ID is propagated using a breadth-first traversal according to the propagation rule described in [2] (see Figure 2-right).

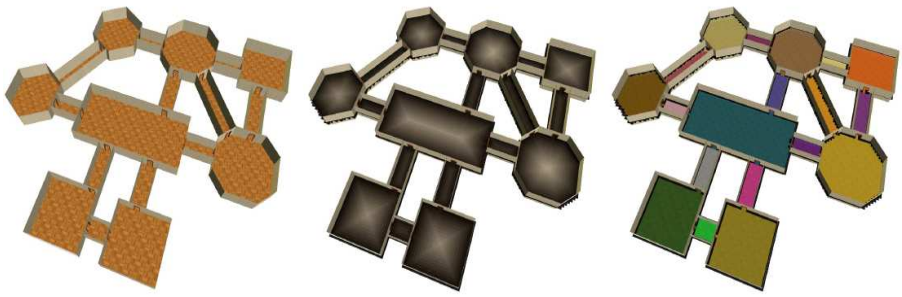


Fig. 2. Cell detection algorithm on a simple building

4.2 Region extraction

At this step, each cell is represented as a collection of voxels. We perform a dilation operation by enlarging this set of voxels to include adjacent non-empty voxels. This step is required to include the geometry enclosing the cell (e.g. room walls). Then we extract a simplified surface enclosing the cell using the surface extraction algorithm proposed in [1]. At the end of this step, the region associated with each cell is represented by a polygonal surface approximating it.

5. WIM Clipping

During runtime, the once the region to be included has been computed (the one containing the user location), we must render only the scene geometry inside this region. This can be seen as a particular case of CSG rendering, as we must render the intersection of the scene with the polygonal region. Of course, we can use a coarse-level, CPU-based culling to the region's bounding box to quickly discard geometry not to be included in the miniature. However, this coarse-level clipping must be combined with a fine-level clipping to the polygonal region. Fortunately, there are efficient algorithms for rendering CSG models using the GPU [8], whose implementation can be greatly simplified when the CSG tree consists of a single boolean operation.

The algorithm we propose is based on building a Layered Depth Image [15] of the region's boundary from the current WIM viewpoint. Layered Depth Images (LDIs) can be efficiently constructed using depth-peeling [7]. An OpenGL framebuffer object (FBO) can be used to render each layer directly into a depth texture. Since our WIM selection algorithm tends to produce regions with low depth complexity, these regions can be encoded with just a few LDI layers and one rendering pass for each layer (note that the LDI is build from a simplified polygonal description of the cell, which is geometrically much simpler than the part of the scene inside the cell).

Once the LDI has been computed, the coarsely-culled scene is rendered using a fragment shader that checks the fragment's depth against the LDI and discards outside fragments. Alternatively, OpenCSG [12], a freely available hardware-accelerated CSG library, can be used. When the cell is found to be convex or approximately convex, we adopt a faster solution. We compute the convex hull of the region and measure the approximation error using a vertex-to-plane metric. If the error is below a given threshold, we take the convex hull as a good approximation of the cell geometry. In this case, we simply use the convex hull planes as clipping planes. If the number of planes is above six (the OpenGL fixed-pipeline limit on additional clipping planes), then a straightforward fragment shader is used to perform the clipping.

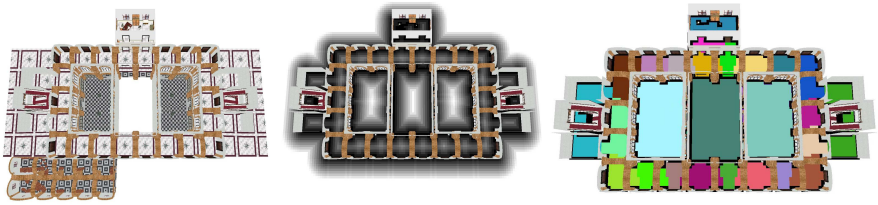


Fig. 3. Cell detection on the test building. Cut-away view of the original model (left), discrete distance field (middle) and cell decomposition (right).

6. WIM Revealing

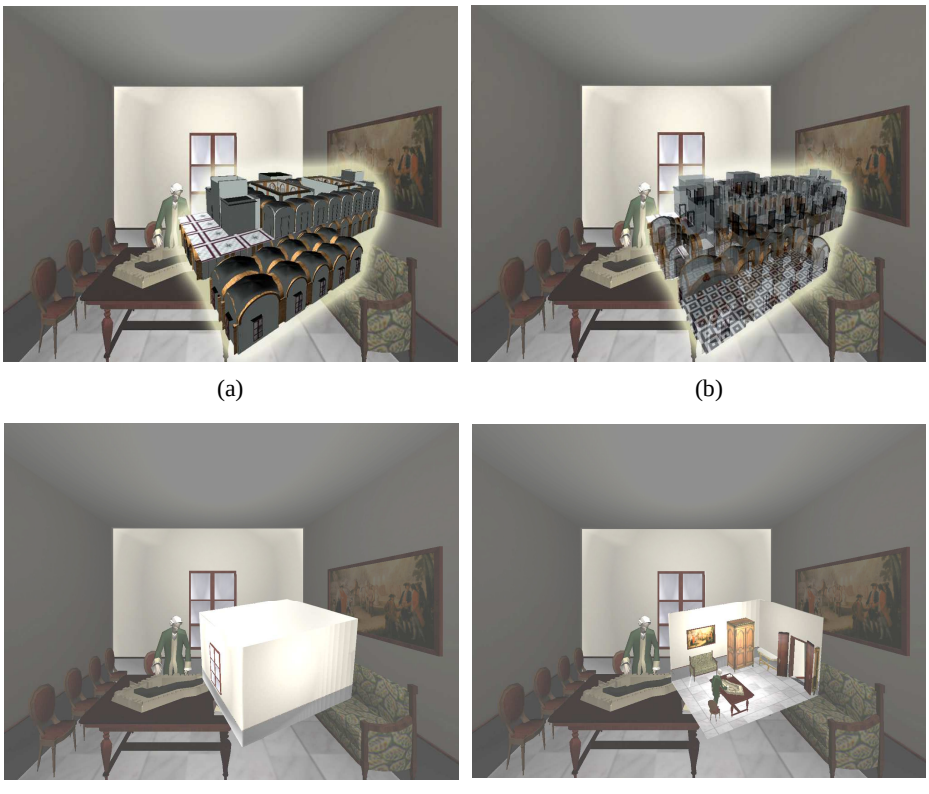
Note that the WIM clipping strategy discussed above does not solve the occlusion problem due to enclosing geometry such as walls. Fortunately, the algorithms for WIM clipping described above can be trivially extended to keep enclosing geometry from occluding interior objects. The only modification required is to offset the front-face planes/faces of the selected region, in the opposite direction of their normals (i.e. increasing the *d* coefficient of the implicit plane equation). The resulting effect is that the frontmost geometry will be culled away by the fragment shader. This idea applies both to convex cells (clipped through clipping planes) and concave cells (clipped through the LDI). In the later case, the LDI is build from biased frontfaces. Due to the simple nature of our cells, this simple approach yields good results (see Figures 1 and 4-d). Note that this approach provides a cut-away view of the miniature copy at no extra cost.

7. Results And Discussion

We have implemented a prototype version of the algorithms described in this paper. Figure 3 shows the results of the cell decomposition step on a three-story building with 150k triangles. Only one of the floors is shown for clarity. Note that resulting cells are quite simple and many of them are approximately convex. A 128x128x128 voxelization was used. The running time of the cell decomposition was 8 seconds on a PentiumIV at 2.0GHz, including the voxelization, distance field computation and cell construction.

Various WIM approximations can be seen in Figure 4. Figure 4-a shows the replica obtained using the original WIM paper (i.e. including the whole scene in the miniature). Note that this WIM is not suitable for object selection and manipulation due to occlusion and scale problems. Rendering the WIM with alpha blending (Figure 4-b) does not provide a sufficiently clear view of the model and still provides a scale unsuitable for object manipulation. The result of our clipping algorithm (excluding the revealing step of our algorithm) is shown in Figure 4-c. Now the miniature includes only the geometry inside the automatically-detected cell, providing a scale suitable for most selection and manipulation tasks. Finally, Figure 4-d shows the final output of our algorithm, including now the revealing step. The quality of the results can also be observed in the accompanying videos [19].

Note that we could replace our LDI-based revealing method by correct (sorted) transparency through depth-peeling, but this time the depth complexity (i.e. the number of layers and hence the number of rendering passes) would depend on the complexity of the objects inside the room, not on the complexity of the geometry enclosing the room, as in our case. Furthermore, note that the cut-away view provides a much more clear representation compared with the transparency-based option. In our current prototype implementation, the WIM is rendered by doing a coarse-level clipping using the replica's bounding volume and a fine-level clipping using the LDI. Although precomputing the scene geometry inside each cell during preprocess might seem a better option, it can make more difficult the integration of our approach with existing applications. Furthermore, most 3D rendering engines provide view frustum culling functionalities which can be easily adapted to perform a conservative culling to the cell's convex hull. Unlike other WIM extensions using continuous scrolling, in our approach the extents of the miniature copy is preserved while the user remains in the same room. The room center and not the current viewpoint position is used as pivot point for hand-held manipulation and visualization of the miniature. We think this behaviour is less distracting to the user.



(a) (b)

(c) (d)

Fig. 4. Different strategies for WIM rendering.

8. Conclusions

In this paper an enhanced version of the WIM has been presented. Our approach supports arbitrarily-complex, densely-occluded scenes by selecting the region to be included in the miniature copy using a semantic subdivision of the scene into logical structures such as rooms. The rationale of our approach is that matching the miniature copies with logical entities of the environment would give the user additional cues to better understand the spatial relationships between the constituent parts that make up these environments. This would also facilitate accurate selection and manipulation of nearby objects through their WIM proxies. Our approach can be combined with classic WIM implementation when the miniature covers the whole scene, for way-finding tasks at large scale levels.

Acknowledgements
This work has been funded by MCYT Ministry under grant TIN2007-67982-C02.

References

1. Andújar, C., Ayala, D., and Brunet, P. *Topology Simplification through Discrete Models.* ACM Transactions on Graphics, 20 (6), pp. 88-105, 2002
2. Andújar, C., Vázquez, P., and Fairén, M. *Way-Finder: Guided Tours Through Complex Walkthrough Models.* Computer Graphics Forum, 2004, 23 (3), 499-508
3. Chittaro, L. and Venkataraman, S. *The Interactive 3D BreakAway Map: A navigation and examination aid for multi-floor 3D worlds*. Proceedings of the 2005 International Conference on Cyberworlds (Washington, DC, USA) 59 -66
4. Cohen-Or, D., Chrysanthou, YL., Silva, CT., and Durand, F. *A survey of visibility for walkthrough applications.* IEEE TVCG, 2003, 9(3), 412-431
5. Diepstraten, J., Weiskopf, D. and Ertl, T. *Interactive Cutaway Illustrations*. Proceedings of EUROGRAPHICS 2003, 523—532
6. Elmqvist, N., and Tsigas, P. *A Taxonomy of 3D Occlusion Management Techniques*. Proc. IEEE VR'07.
7. Everitt, C. *Interactive Order-Independent Transparency*. White Paper, NVIDIA Corporation, May 2001. http://developer.nvidia.com/object/Interactive_Order_Transparency.html
8. Hable, J. and Rossignac, J. 2005. *Blister: GPU-based rendering of Boolean combinations of free-form triangulated shapes*. SIGGRAPH'05, 1024-1031
9. Haumont, D., Debeir, O., and Sillion, F. *Volumetric cell-and-portal generation.* Computer Graphics Forum, 22(3): 303–312, September 2003.
10. Hong, L., Muraki, S., Kaufman, A., Bartz, D., and He, T. *Virtual voyage: interactive navigation in the human colon*. In Proc. Computer Graphics and Interactive Techniques International. 2007, 27-34.
11. Jones, M., Satherley, R. *Using distance fields for object representation and rendering*. In Proc. 19th annual Conference of Eurographics (UK chapter), 37–44, 2001.
12. Kirsch, F., and Döllner, J. OpenCSG: *A library for image-based CSG rendering*. Proc. USENIX 05.
13. La Viola, Jr., J., Feliz, D., Keefe, D., and Zeleznik, R. *Hands-free multi-scale navigation in virtual environment*. Proc. of the Symposium on Interactive 3D Graphics'01, 9-15
14. Pausch, R., Burnette, T., Brockway, D., Weiblen, M.E. *Navigation and Locomotion in virtual worlds via flight into hand held miniatures.* SIGGRAPH'95, 1995, 399-400
15. Shade, J., Gortler, S., He, L., and Szeliski, R. 1998. *Layered depth images*. SIGGRAPH '98. 231-242
16. Stoakley, R., Conway, M., and Pausch, R. *Virtual reality on a WIM: interactive worlds in miniature*. SIGCHI '95: SIG on Human factors in computing systems, 1995, 265-272
17. Teller, S. J., and Séquin, C. H., *Visibility preprocessing for interactive walkthroughs.* In Proc. SIGGRAPH '91. 61-70.
18. Wingrave, C., Haciahmetoglu, Y., Bowman, D. "*Overcoming World in Miniature Limitations by a Scaled and Scrolling WIM*," 3D User Interfaces (3DUI'06) 2006, 11-16.
19. http://www.lsi.upc.edu/~virtual/WIM