World-in-Miniature Interaction for Complex Virtual Environments

Ramón Trueba, MOVING Group, Universitat Politècnica de Catalunya, Spain

Carlos Andujar, MOVING Group, Universitat Politècnica de Catalunya, Spain

Ferran Argelaguet, MOVING Group, Universitat Politècnica de Catalunya, Spain

ABSTRACT

Object occlusion is a major handicap for efficient interaction with 3D virtual environments. The well-known World in Miniature (WIM) metaphor partially solves this problem by providing an additional dynamic viewpoint through a hand-held miniature copy of the scene. However, letting the miniature show a replica of the whole scene makes the WIM metaphor suitable for only relatively simple scenes due to occlusion and level of scale issues. In this paper, the authors propose several algorithms to extend the idea behind the WIM to arbitrarily complex scenes. The main idea is to automatically decompose indoor scenes into a collection of cells that define potential extents of the miniature replica. This cell decomposition works well for general indoor scenes and allows for simple and efficient algorithms for preserving the visibility of potential targets inside the cell. The authors also discuss how to support interaction at multiple levels of scale by allowing the user to select the WIM size according to the accuracy required for accomplishing the task.

Keywords: *3D Selection and Manipulation, 3D User Interfaces, Occlusion Management, Virtual Environments, World-In-Miniature*

INTRODUCTION

During the last few decades much research has been devoted to develop new strategies for facilitating user interaction with complex, densely-occluded virtual environments. Among the different approaches that have been proposed for measuring scene complexity, the most relevant one, from the point of view of 3D user interfaces, is *depth complexity*, which depends on the number of occluding objects. Occlusion is a big handicap for the efficient accomplishment of 3D interaction tasks, including selection and manipulation, as most interaction techniques for these tasks require the object, along with relevant context information, to be visible. Navigating to a location where potential targets are visible is a common solution to this problem, but requiring the user to navigate every time an occluded object must be selected hinders performance in manipulation-intensive applications.

In this paper we address the problem of facilitating the interaction with highly-occluded, indoor scenes, focusing on applications running on spatially-immersive displays such as CAVEs. This work is based on the World in Miniature

DOI: 10.4018/jcicg.2010070101

Figure 1. User interacting with the WIM

(WIM) (Stoakley, Conway, & Pausch, 1995), which is particularly appropriate for performing tasks requiring relevant context information to be visible. The WIM complements the first-person perspective offered by typical Virtual Reality applications with a second dynamic view of a miniature copy of the virtual world (Figure 1). This second exocentric view of the world helps users to understand the spatial relationships of the objects and themselves inside the virtual world. Objects in the miniature replica of the scene are referred to as *proxies*. Typically the miniature is hand-held, the non-dominant hand being used for rotating the WIM, thus establishing the frame of reference for further interaction tasks, and the dominant hand is used for selection, manipulation, and navigation tasks. Its hand-held feature also allows it to be quickly explored from different viewpoints without modifying the immersive point of view.

The WIM metaphor supports most 3D user interaction tasks, including selection, manipulation and navigation. Objects can be selected either by pointing directly at them or by pointing at their WIM proxy. For simple scenes, a rotation of the non-dominant hand is enough to view and pick objects that are occluded from the immersive viewpoint. Likewise, objects can be manipulated either at the one-to-one scale offered by the immersive viewpoint, or at the WIM scale. By representing the virtual camera with a 3D avatar, the WIM provides a convenient way to quickly change its location inside the virtual environment.

Unfortunately, the WIM metaphor has two important limitations regarding its scalability which have prevented its widespread use. The first one is concerned with the level-of-scale at which different interaction tasks have to be accomplished. Different interaction tasks require, broadly speaking, different levels of accuracy. Since the WIM covers the whole scene, it is appropriate only for rough, coarse-level interaction tasks. For example, a WIM showing a full house might be suitable for quickly moving the camera from a room to another, but it lacks accuracy for finer tasks, such as laying out furniture pieces. Most extensions to the WIM are concerned with this problem. A second major limitation of the WIM is occlusion management, i.e. how to keep relevant objects simultaneously visible while preserving important context information.

A key observation is that many user interaction tasks are *local*, i.e. they can be accomplished with a minimum amount of context information. For instance, adjusting the position of a piece of equipment in a room might be accomplished disregarding the contents of other rooms. This suggests that the part of the scene covered by the miniature replica should be adapted according to the user task. Therefore two key problems have to be addressed. On the one hand, decide which part of the virtual environment must be included in the replica. As stated above, using a replica of the whole environment is only feasible for simple scenes like a single room. Some authors have extended the WIM metaphor to cope with complex models (Chittaro, Gatla, & Venkataraman, 2005; LaViola et al., 2001; Wingrave, Haciahmetoglu, & Bowman, 2006). These approaches define a region of interest (ROI) and put in the miniature copy only those objects inside this ROI. The user is allowed to scale and move the ROI, either automatically or manually. Although these techniques allow the accomplishment of user tasks at different levels of scale, the solutions adopted are valid for a limited class of models, or assume the input scene already provides some information about its logical structure. On the other hand, the adoption of the WIM for complex scenes requires some strategy to handle 3D occlusion. Bounding geometry such as walls must be conveniently removed to make interior objects visible. Notice that backface culling techniques (Stoakley, Conway, & Pausch, 1995) are suitable only for simple models. General 3D models require more sophisticated techniques for handling 3D occlusion (Diepstraten, Weiskopf, & Ertl, 2003).

In this paper we extend the WIM metaphor so that it can be used in arbitrarily-complex, densely-occluded scenes. Our approach, dubbed Dynamic Worlds in Miniature (DWIM) is based on a selection of the region to be included in the miniature copy through a subdivision of the scene into logical structures such as rooms, floors and buildings. These elements are computed automatically from the input scene during a preprocessing step. The rationale here is that matching the WIM content with logical entities of the environment will help the user to accomplish interaction tasks. A logical decomposition provides additional cues to better understand the spatial relationships among the parts of the scene, and facilitates bounding those regions which provide the context information required to accomplish a local task. Furthermore, this subdivision offers a more intuitive and clear view of the nearby environment, suitable for precise object selection and manipulation. Indeed, our subdivision greatly simplifies the generation of cut-away views of the miniature copy, so that enclosing geometry does not occlude interior objects (Figure 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.

PREVIOUS WORK

In this section we briefly review previous work related to the problem being addressed (enhancements to the WIM metaphor) and to our adopted solution (cell decomposition and occlusion management).

Worlds in Miniature

The WIM was proposed originally by Stoakley, Conway, and Pausch (1995) who discussed their application to selection, manipulation and traveling tasks. They found that users easily understood the mapping between virtual world objects and their proxy counterparts in the WIM. Unfortunately, using a replica of the whole environment in the miniature limits its application to simple models like a single room, due to the occlusion and level-of-scale

Figure 2. Examples of our improved WIM rendering. The miniature replica provides a cut-away view of a part of the model according to the user's position.

problems discussed in the preceding section. Several WIM extensions have been proposed to overcome these limitations. The STEP WIM proposed by La Viola et al. (2001) 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. The Scaled and Scrolling WIM (SSWIM) (Wingrave, Haciahmetoglu, & Bowman, 2006) supports interaction at multiple levels of scale through scaling and scrolling functions. SSWIM adds functionally and hence complexity because the user has to scale the model manually. The scrolling is automatic though when the user moves to a position outside of a dead zone. Chittaro, Gatla, and Venkataraman (2005) propose an extension to support user navigation through virtual buildings, allowing users to explore any floor of a building without having necessary to navigate. Unfortunately, it requires an explicit identification of all the polygons on the different floors, which can be a time-consuming task for 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

Portal culling techniques (Airey, 1990) accelerate the rendering of indoor models by dividing the scene into cells (that usually correspond to rooms and hallways) connected by portals which represent the openings (for example doors and windows) that connect the cells. The resulting structure is called a cell-and-portal graph (CPG) and encodes the visibility of the scene. Only a few papers refer to the automatic determination of CPGs. Furthermore, most of them work under important restrictions (Cohen-Or et al., 2003), such as axis-aligned walls (Teller & Séquin, 1991). Hong, Muraki and Kaufman (1997) 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 watershed transform of the scene (Haumont, Debeir, & Sillion, 2003; Andújar, Vázquez, & Fairén, 2004). Our approach for cell detection is based on the scene subdivision proposed in (Andújar, Vázquez & Fairén, 2004).

Note however that we do not need to recover the exact geometry of the portals, but only to approximate the geometry bounding each cell.

Occlusion Management

The presence of occluding surfaces and other distracting objects in complex 3D models is a big handicap to the efficient accomplishment of discovery, selection, manipulation and navigation tasks. Current 3D selection techniques, for example, require the target object to be visible. Occlusion management techniques are valuable tools for helping viewers understand the spatial relationships between the constituent parts of complex scenes. Elmqvist and Tsigas (2008) identify five main design patterns for occlusion management: Multiple Viewports (using multiple views of the scene), Virtual X-Ray (turning occluded objects visible), Tour Planners (camera animations reveal the otherwise occluded geometry), Interactive Exploders (adopted for the WIM in (Chittaro, Gatla, & Venkataraman, 2005) through a floor sliding mechanism) and Projection Distorter (using nonlinear projections to combine multiple views into a single view). Among these, Virtual X-Ray techniques are particularly relevant for WIM-based manipulation as they make the discovery task straightforward and facilitate access to the potential targets by selectively removing occluders. Occluders can be removed by turning them semi-transparent or invisible (Elmqvist & Assarsson, 2007). Transparency is particularly simple to implement as it eliminates the need for identifying distracting objects, but rendering semi-transparent objects increases the visual complexity of the scene and the cognitive load to the user. Occluder removal techniques can be view-independent (Coffin & Hollerer, 2006) or view-dependent (Chittaro, Gatla & Venkataraman, 2005; Burns & Finkekstein, 2008). Since the WIM replica is hand-held and can be rotated by the user interactively, we only discuss view-dependent techniques.

Early cut-away papers used depth buffer and image processing techniques to replace occluding surfaces by semi-transparent representations (Feiner & Seligmann, 1992). Diepstraten, Weiskopf, and Ertl (2003) propose a number of efficient algorithms for generating cut-away views, provided that the scene objects have been labeled as interior or exterior. Some approaches use CSG rendering techniques so as resulting holes are rendered as if they were actually cut into the obstructing geometry, making the interior of solid objects visible by the cutaway action (Coffin & Hollerer, 2006). Recent works on interactive cutaways (Burns & Finkelstein, 2008) introduce an image space algorithm based on jump flooding to generate a depth-image representation of the cutaway surface at interactive rates. Note that the above techniques require the user to manually define the shape of the cutout volume, or to explicitly identify the objects to be exposed. We will show that our cell subdivision of the scene greatly simplifies the task of removing occluding objects.

OUR APPROACH

Overview

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

(a) Delimitation: instead of putting the whole scene in the miniature, we compute the part of the scene to be included according to the user's location. The region of the space covered by the WIM is represented by a polyhedral cell.
(b) Clipping: Our clipping solution can be seen as a particular case of Constructive Solid Geometry (CSG) rendering, as we must render the intersection of the scene with a polyhedral approximation of the current cell.
(c) Revealing: we keep the walls of the cell from occluding interior objects without sacrificing the outer context entirely.

A key ingredient of our approach is the automatic decomposition of the scene into a collection of *cells*. Roughly speaking, a cell is a region with approximately-constant visibility,

with no significant occluding surfaces on its *interior* (i.e. all occluding surfaces must be on the *boundary* of the cell). Using cells as basic units for defining the extents of the miniature replica provides several advantages. First, cells often match with the intuitive notion of rooms of a model. Second, since no large occluding surfaces occur inside cells, occlusion management in a cell can be solved by selectively removing parts of the enclosing boundary, so that interior objects are visible.

Our WIM delimitation strategy has two main steps (described in detail in next subsections): *cell detection*, where the scene is discretized into a voxelization and then voxels are grouped into cells with approximately constant visibility, and *polyhedral approximation*, where a low-polygon approximation of the cell boundary is computed. Our subdivision scheme produces cells which roughly correspond to rooms and hallways.

The user will be provided with a miniature containing only the objects inside the current cell. Since regions are detected during preprocessing, we could clip the scene geometry to each cell and store the resulting geometry during preprocessing. 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. Our solution (described in next subsections) can be seen as a particular case of CSG rendering which can be implemented easily on the GPU (Kirsch & Döllner, 2005). Finally, we apply a view-dependent cut-away technique for removing the frontmost geometry of the cell so that internal objects are visible (revealing). An important benefit of using cells with approximately constant visibility is that they greatly simplify the occluder removal task, as removing the enclosing geometry will be enough to reveal interior objects.

Preprocessing

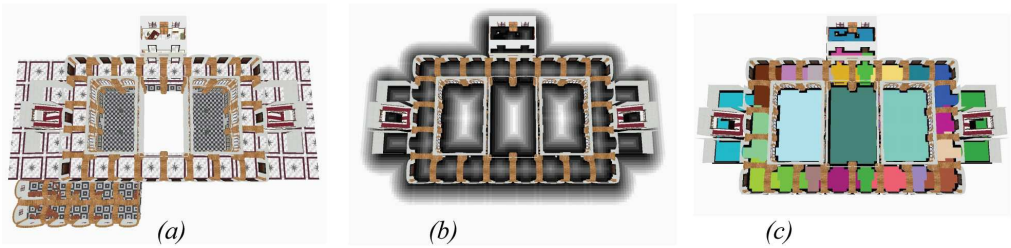
As a preprocess, we compute a cell decomposition of the scene and create a rough polygonal approximation for each cell. We first convert the input model into a voxelization where voxels intersected by the scene geometry are detected. This conversion can be performed in the GPU (Dong et al., 2004). Our current implementation is based on the voxelization algorithm by Eisemann and Décoret (2006). The algorithm builds the voxelization by simply rasterizing the geometry with a fragment shader that computes the depth of each fragment. Although the algorithm is not exact and misses some intersected voxels (about 0.2% in the test models) due to inherent features of graphics hardware rasterization, we have chosen it because it is extremely fast: voxelizations up to 2048^3 of a 2M faces model can be computed in less than one second. Missing voxels are unlikely to have a large impact on the distance field and cell detection steps, making more conservative algorithms for GPU voxelization not appropriate for this task.

We then compute a discrete distance field (Figure 3(b)). Distance fields have been used for generating cell-and-portal graph decompositions (Haumont, Debeir, & Sillion, 2003; Andújar, Vázquez, & Fairén, 2004) as the valleys of the distance field approximately correspond to the portals of the scene. We approximate the distance field using a 5x5x5 Chamfer distance matrix (Jones & Satherley, 2001) which requires only two traversals.

Voxels are grouped into cells through a region-growing process. We start from the voxel having the maximum distance value. During this process, all conquered voxels are assigned the same cell ID. The propagation of the cell ID continues until the whole cell is bounded by voxels having either negative distance (meaning already-visited voxels), zero distance (non-empty voxels) or positive distance greater than the voxels at the cell boundary. When the growth of one cell stops we simply choose a new unvisited maximum and repeat the previous steps until all voxels have been classified (see Figure 3(c)). Since geometric noise produced by furniture pieces severely distort the distance field, we filter out small cells (Figure 4) as proposed in (Andújar, Vázquez, & Fairén, 2004).

We approximate each cell (consisting of a collection of voxels) with a polyhedron using the surface extraction algorithm proposed by

Figure 3. Cell detection example:(a) input model; (b) distance field; (c) detected cells



Andújar, Ayala, and Brunet (2002). This step is made for the solely purpose of accelerating the GPU-assisted clipping of the geometry outside the cell during runtime. Before extracting the surface, a dilation operation is performed to each cell to include adjacent non-empty voxels. After dilation, cells are no longer disjoint. For example, a wall separating two cells will belong to both cells. The dilation step is required for the cell to include the geometry enclosing it, such as walls (see Figures 6(c) and 6(f)). This surrounding geometry will be selectively drawn or removed depending on the viewpoint, so as to keep as much context information as possible, as discussed in next sections.

Delimitation and Clipping

At runtime, we identify the cell containing the user's viewpoint, using random-access to the voxelization described in the preceding section. We use the associated polyhedral representation of the cell to define the extents of the WIM: the user will be presented with a miniature replica comprising only the objects inside the current cell.

Since regions are detected during preprocessing, the clipping can be performed either during preprocessing or at runtime. Our current implementation adopts a runtime approach for clipping the scene geometry to the polyhedral approximation of the current cell. The main

Figure 4. A partial view of a building and its color-coded cell decomposition, prior to the dilation operation

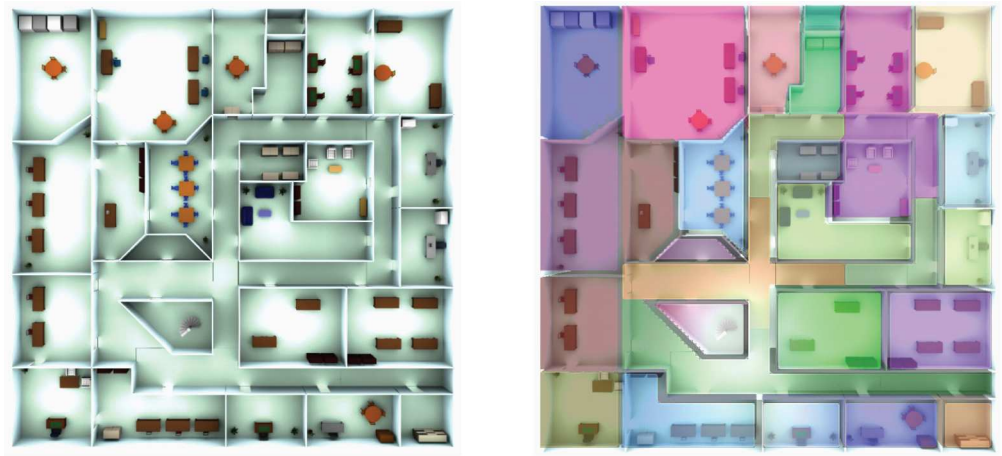
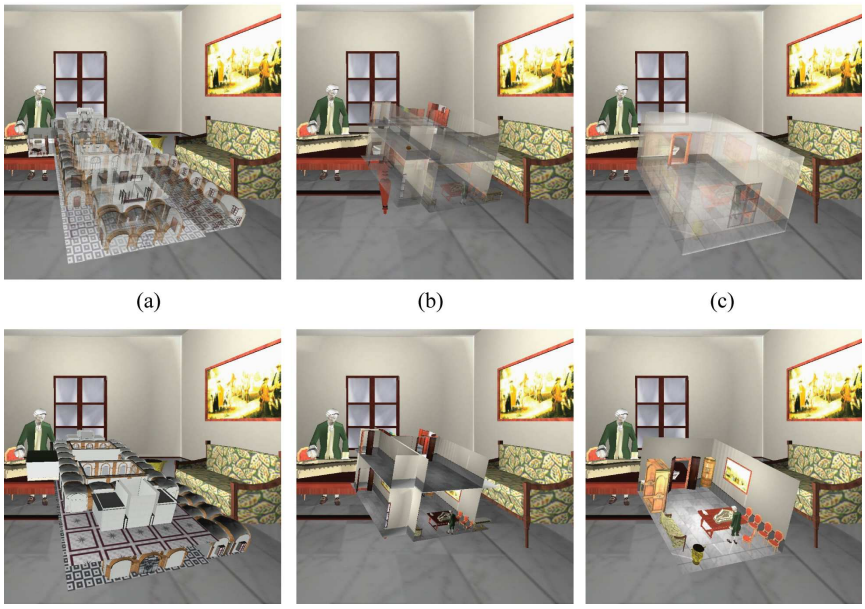


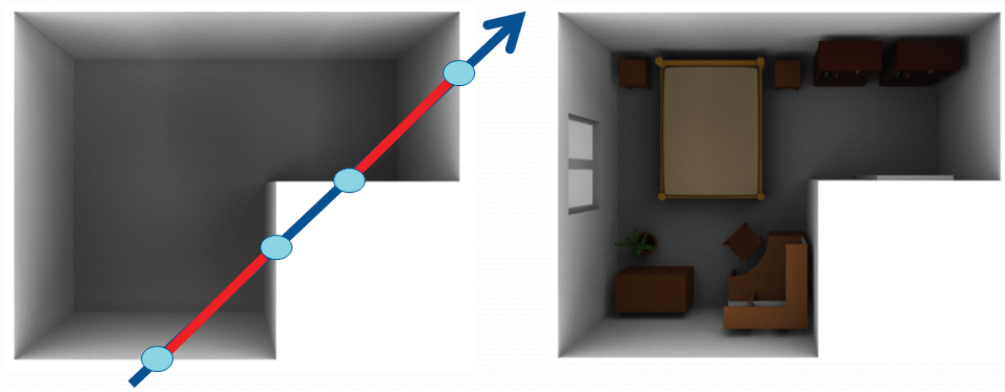
Figure 6. Different strategies for WIM rendering



advantage of this approach is that it supports recomputing the cell decomposition whenever some large occluding surface such as a wall is modified as a result of a user manipulation task.

Our clipping solution can be seen as a particular case of CSG rendering, as we must render the intersection of the scene with a polyhedral approximation of the current cell. We 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 polyhedral cell. Fortunately, there are efficient algorithms for rendering CSG models using the GPU (Hable, Rossignac, & Blister, 2005; Kirsch & Döllner, 2005), whose implementation can be greatly simplified when the CSG tree consists of a single boolean operation, as in our case. The algorithm we propose is based on building a Layered Depth Image (Shade et al., 1998) of the polyhedral approximation from the current WIM viewpoint. Layered Depth Images (LDIs) can be efficiently constructed using depth-peeling (Everitt, 2001). We use OpenGL framebuffer objects (FBOs) to render each layer directly into a depth texture. Since the cell detection algorithm tends to produce regions with very 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 low-polygon description of the cell's boundary, 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 fragments outside the LDI (and hence outside the cell). Our current implementation uses the OpenGL's texture array extension to encode the LDI, so that the fragment program is allowed to access an arbitrary number of LDI layers, although the LDI representation of a typical cell does no require more than 4-6 layers (Figure 5).

Figure 5. The LDI is built from a polyhedral approximation of the current cell (left). For each viewing ray, the LDI defines alternating intervals of outside (blue) and inside (red) geometry, which allows clipping the scene to the cell's boundary (right)



Revealing

Note that cells include the bounding walls of the rooms they represent. Fortunately, our clipping strategy can be trivially extended to keep frontmost enclosing geometry from occluding interior objects. This can be accomplished by adding an offset to the frontmost faces of the polyhedral region in the opposite direction of their normals. Indeed, we apply the offset as an epsilon used during the depth comparison with the first LDI layer. The resulting effect is that the frontmost geometry is discarded in the fragment shader, leaving interior objects visible (Figure 6(f)). Furthermore, this strategy can be easily extended when adding support to multiple levels of scale, as discussed below.

RESULTS AND DISCUSSION

We conducted a user-study to evaluate potential advantages of our *WIM delimitation* and *WIM revealing* strategies in comparison with competing approaches. We focused on selection and manipulation tasks performed in spatially-immersive displays such as CAVEs. The test model was a three-story building with about 60 rooms and 150k polygons. Figure 3(c) shows the results of the cell decomposition step on the test model, using a 128^3 voxelization. The running time for the cell decomposition was 3.8 seconds, including the voxelization (80 ms), distance transform (120 ms) and polyhedral approximation (3.6 s) steps, measured on a 2.66GHz QuadCore PC.

Concerning WIM delimitation, we compared our approach (based on automatic cell detection) with a *user-adjustable cube* defining the part of the scene to be included in the replica. Our adjustable-cube implementation is inspired in SSWIM (Wingrave, Haciahmetoglu, & Bowman, 2006) with some modifications to match our Virtual Reality system. At the beginning of the task, the cube was centered at the user's position and covered a fraction of the model (in the experiments the cube covered about a 10% of the model). Users were able to manually scale up and down the cube at constant speed (2.5 m/s for the experiments), using two Wanda buttons. Similar to (Wingrave, Haciahmetoglu, & Bowman, 2006), the cube center was updated when the user navigated to a location farther away from the initial position (we set the distance threshold to a 60% of the cube size). The adjustment of the cube size and position only affected the WIM coverage; the apparent size of the WIM remained constant during the experiment (about $30cm^3$). Regarding WIM revealing, we considered two options for

distractor removal: turning them invisible (using our *z-offset* approach) or semi-transparent (using unsorted alpha blending with depth buffer write operations disabled).

Given that our WIM delimitation strategy automatically matches the WIM extents to the current cell, in the best case scenario one could expect DWIM delimitation to have a positive impact on selection performance. In practice, however, this may not be the case. On the one hand, DWIM's lack of manual adjustment can keep the user from finding a suitable level of scale for performing the spatial task. On the other hand, DWIM's cells include enclosing geometry which might distract the user unless efficiently removed. We also expected the WIM revealing strategy to have a varying impact depending on the WIM delimitation strategy and the amount of intervening distractors. In any case, rendering semi-transparent objects increases the visual complexity so we expected the transparency option to have a negative impact on selection performance.

Apparatus

All the experiments were conducted on a four-sided CAVE with active-stereo projectors at 1280x1280 resolution. The input device was a 6-DOF Ascension Wanda and a tracking system with 2 receivers providing 60 updates/s with 4 ms latency. The experiment was driven by a cluster of 2.66GHz QuadCore PCs with Nvidia Quadro 5500 cards.

Participants

Eleven volunteers (1 female, 10 male), aged from 24 to 35, participated in the experiment. Some participants (4) had no experience with VE applications; 5 had some experience and 2 were experienced users.

Procedure

The task was to select some objects and move them to a target destination represented by a sphere. Target destinations were chosen to be more than 4 meters away from the initial position so that the task could be accomplished more efficiently using the WIM proxies rather than the actual objects. Selection and manipulation in the WIM was accomplished using the virtual hand metaphor. Users were requested to complete the task as quickly as possible. Users were allowed to navigate at 2m/s constant speed using the wanda's joystick.

Design

A repeated-measures within-subjects design was used. The independent variables were WIM delimitation (adjustable cube, current cell) and WIM revealing (transparency, z-offset), resulting in four combinations. Each participant performed the experiment in one session lasting approximately 15 min. The session was divided into four blocks, one for each technique.

Before each block users were provided with a short training session which required them to complete practice trials. We measured the time to complete the task, distinguishing between selection, manipulation and navigation times (some users decided to navigate to find a better view to accomplish the task).

Various options for WIM delimitation and rendering are shown in Figure 6. Figure 6(d) shows the replica covering the whole model. Note that this WIM is not suitable for object manipulation due to occlusion and level-of-scale problems. Rendering the WIM with alpha blending (Figure 6(a)) does not provide a sufficiently clear view of the model and still provides a scale unsuitable for object manipulation. Figures 6(b) and 6(e) show sample renderings with the adjustable cube. The result of our clipping algorithm (with transparency instead of z-offset revealing) is shown in Figure 6(c). Now the miniature includes only the geometry inside the automatically-detected cell, providing a better scale for accurate selection and manipulation

Results

Results of the experiment are shown in Figure 7. The two-way ANOVA showed a significant effect ($p < 0.01$) for the WIM delimitation factor, users performing much better (30% less time) with the WIM delimited automatically to

the current cell rather than manually with the cube. This result was expected as our approach eliminates the need to manually adapt the WIM extent to the task. We also found a significant effect ($p = 0.04$) for the WIM revealing factor, and a clear interaction effect ($p < 0.01$) between delimitation and revealing. When the WIM was delimited automatically to the current cell, the choice of the revealing strategy had a significant effect, users performing much better when removing occluding walls with the z-offset approach. This seems to confirm our hypothesis on the limitations of the transparency option for performing spatial tasks (see Figures 6(c) and 6(f)). However, when delimiting the WIM with the adjustable cube, the revealing strategy had little impact on overall user performance. Our explanation for this is that rendering the WIM with transparency resulted in higher visual complexity of the resulting images, but this was compensated by the fact that users were not forced to perform a fine adjustment of the cube to avoid occluding walls (see Figures 6(b) and 6(e)).

We conducted a second experiment with identical setup except for the object to be manipulated, which was moved to an adjacent room. This change implied that, when using our approach, users were required to navigate to the target room, whereas for the adjustable cube users were able to choose whether to navigate or to scale-up the cube. We also chose a target room with less occluding objects to favor transparency against our z-offset approach. The results of the second experiment are shown in Figure 7 (right). Despite being a worst-case scenario for our approach, we still found a significant effect for WIM delimitation ($p = 0.01$), users performing better with DWIM. Our best explanation is that, in densely-occluded environments, scaling-up the cube rapidly increases the number of distractors, adding occluders from neighboring cells and keeping it from providing a clear view for carrying out spatial tasks. Concerning WIM revealing, we did not found a significant effect on completion times ($p=0.1$). This seems to confirm our hypothesis that the alpha blending option is only usable with a low number of occluders.

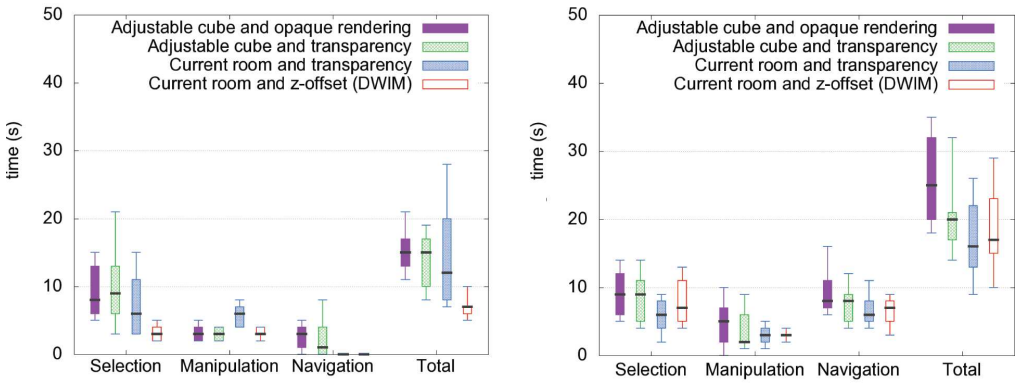
Runtime Overhead

Our technique introduces some performance overheads with respect to competing WIM implementations. Our rendering algorithm requires building the LDI of a low-polygon approximation of the cell and drawing the objects with a simple fragment shader. Notice that the LDI is built using a FBO with the size of the WIM viewport, which is much smaller than the immersive viewport. The overhead was found to be less than one millisecond, which had no noticeable impact on the application frame rate. When compared with WIM implementations putting a replica of the whole scene, our approach clearly wins as only a fraction of the geometry is rendered every frame.

Supporting Several Levels of Scale

Our technique can be extended to provide a miniature replica at multiple levels of scale, allowing the user to select the extents of the WIM according to the accuracy of the intended interaction task. The WIM delimitation strategy can be modified so as to allow the user to interactively expand the extent of the miniature copy by adding cells adjacent to the current cell. This can be easily accomplished as cell adjacency information can be computed as a subproduct of the cell detection algorithm (Andújar, Vázquez & Fairén, 2004). The main advantage with respect to continuously scaling a cube is that the WIM is expanded in a discrete manner, showing always a complete subset of cells, i.e. cells are not splited during zooming operations. Of course, putting more than one cell on the replica implies modifying the WIM revealing strategy, so as to keep potential targets visible in the presence of large occluding surfaces *inside* the WIM. A simple solution consists in extending the WIM revealing algorithm as follows. Suppose the WIM contains a set of neighboring cells $\{C_1, \ldots, C_n\}$. Let C_k be the cell containing the user's hand position. We build the LDI by rendering cells in $\{C_1, \ldots, C_n\}$, but giving visibility priority to C_k, so that the frontmost geometry of C_k belongs to the first LDI layer. Using such LDI would provide a clear view of

Figure 7. Box plots for task 1 (left) and task 2 (right)



the objects inside C_k. Assuming that potential targets are limited to the cell containing the user's hand, this strategy would provide a clear view of potential targets while preserving most context information (see Figure 8).

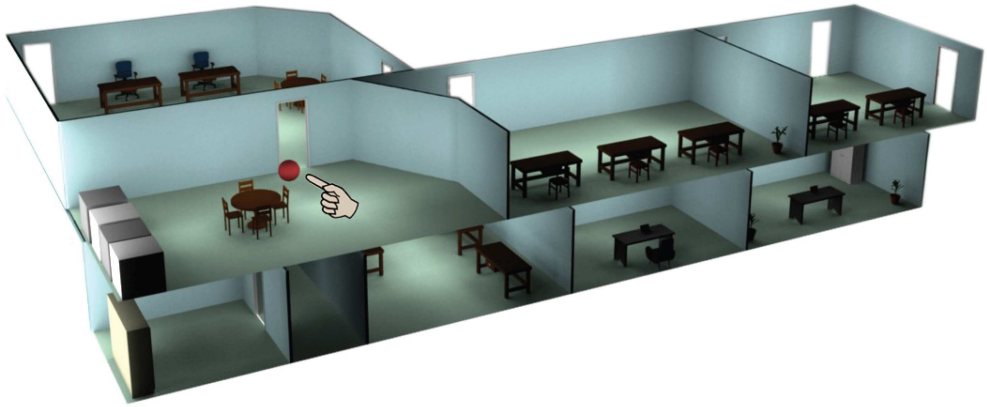
Limitations

Our approach is oriented towards indoor scenes with well-defined room structures (e.g. office buildings). It is obviously not appropriate for outside environments, but occlusion management on outdoor scenes is not a critical problem. Our WIM revealing strategy is slightly sensitive to the offset used for comparisons against the LDI representation of the current cell. As a result, an object being manipulated might partially disappear when getting close to one of the frontmost walls of the cell. However, this can be avoided by simply rotating the hand-held WIM so that the wall becomes backmost geometry. The cell detection method is also limited by the size of the voxel grids that can be realistically managed. A solution for large-scale models is to split the model into smaller parts and build separate voxelizations for each part.

CONCLUSION AND FUTURE WORK

In this paper an enhanced version of the WIM metaphor 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 help the user to accomplish spatial tasks. We have shown empirically that our approach facilitates accurate selection and manipulation of 3D objects through their WIM proxies, providing a clear, low-complexity view automatically adapted to the user's location. Our current system allows users to switch automatic WIM delimitation on and off; when turned off, the WIM shows a miniature replica of the whole scene, which can be more suitable for way-finding tasks at large scale levels. Unlike other WIM extensions using continuous scrolling, our approach preserves the extents of the miniature copy while the user remains in the same room. The room center instead of the current viewpoint is used as pivot point for hand-held manipulation and visualization of the miniature. We have found this behavior to be less distracting to the users.

Figure 8. Rendering a WIM with several neighboring cells facilitates manipulation tasks among cells. The WIM revealing strategy has been modified to keep visible the objects inside the cell containing the user's hand (drawn as a red ball).



There are several directions that can be pursued to extend the current work. It may be interesting to explore alternative visibility-aware decompositions of the scene suitable for outdoor environments, enabling fast identification and removal of occluding geometry, as well as cell decompositions requiring no preprocessing at all. Although we have focused on its implementation for the WIM, it may be interesting to explore the application of our revealing strategy to manipulate directly the objects shown in the immersive view.

ACKNOWLEDGMENT

This work has been partially funded by the Spanish Ministry of Science and Technology under grant TIN2007-67982-C02.

REFERENCES

Airey, J. (1990). *Increasing update rates in the building walkthrough system with automatic model-space subdivision and potentially visible set calculations.* Unpublished PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill.

Andújar, C., Ayala, D., & Brunet, P. (2002). Topology simplification through discrete models. *ACM Transactions on Graphics*, *20*(6), 88–105.

Andújar, C., Vázquez, P., & Fairén, M. (2004). Wayfinder: Guided tours through complex walkthrough models. *Computer Graphics Forum*, *23*(3), 499–508. doi:10.1111/j.1467-8659.2004.00781.x

Burns, M., & Finkelstein, A. (2008). Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Transactions on Graphics*, *27*(5), 1–7. doi:10.1145/1409060.1409107

Chittaro, L., Gatla, V. K., & Venkataraman, S. (2005). The interactive 3d breakaway map: A navigation and examination aid for mult-ifloor 3d worlds. In *Proceedings of CW '05: 2005 International Conference on Cyberworlds* (pp. 59-66). Washington, DC: IEEE Computer Society.

Coffin, C., & Hollerer, T. (2006). *Interactive perspective cut-away views for general 3d scenes*. In *Proceedings of3DUI '06: IEEE Symposium on 3D User Interfaces* (pp. 25-28).

Cohen-Or, D., Chrysanthou, Y., Silva, C., & Durand, F. (2003). A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, *9*(3), 412–431. doi:10.1109/TVCG.2003.1207447

Diepstraten, J., Weiskopf, D., & Ertl, T. (2003). Interactive cutaway illustrations. In . *Proceedings of Eurographics*, *2003*, 523–532.

Dong, Z., Chen, W., Bao, H., Zhang, H., & Peng, Q. (2004). Real-time voxelization for complex polygonal models. In *Proceedings of PG '04: 12th Pacific Conference on Computer Graphics and Applications* (pp. 43-50).

Eisemann, E., & Décoret, X. (2006). Fast Scene Voxelization and Applications. In *Proceedings of the 2006 ACM Symposium on Interactive 3D Graphics and Games* (pp. 70-78).

Elmqvist, N., Assarsson, U., & Tsigas, P. (2007). Employing dynamic transparency for 3d occlusion management: Design issues and evaluation. In *Proceedings of INTERACT*, *2007*, 532–545.

Elmqvist, N., & Tsigas, P. (2008). A taxonomy of 3d occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, *14*(5), 1095–1109. doi:10.1109/TVCG.2008.59

Everitt, C. (2001). *Introduction to interactive order-independent transparency*. NVIDIA Corporation.

Feiner, S., & Seligmann, D. (1992). Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations. *The Visual Computer*, *8*(5), 292–302. doi:10.1007/BF01897116

Hable, J., & Rossignac, J. (2005). Blister: Gpu-based rendering of boolean combinations of free-form triangulated shapes. *ACM Transactions on Graphics*, *24*(3), 1024–1031. doi:10.1145/1073204.1073306

Haumont, D., Debeir, O., & Sillion, F. (2003). Volumetric cell-and-portal generation. In *Proceedings of the Computer Graphics Forum* (pp. 3-22)

Hong, L., Muraki, S., Kaufman, A., Bartz, D., & He, T. (1997). Virtual voyage: interactive navigation in the human colon. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (pp. 27-34).

Jones, M., & Satherley, R. (2001). Using distance fields for object representation and rendering. In . *Proceedings of Eurographics*, *2001*, 37–44.

Kirsch, F., & Döllner, J. (2005). Opencsg: a library for image-based csg rendering. In *Proceedings USENIX 05* (p. 49).

LaViola, J. J., Jr., Feliz, D. A., Keefe, D. F., & Zeleznik, R. C. (2001). Hands-free multi-scale navigation in virtual environments. In *Proceedings of the Symposium on Interactive 3D Graphics'01* (pp. 9-15).

Shade, J., Gortler, S., He, L. W., & Szeliski, R. (1998). Layered depth images. In . *Proceedings of SIGGRAPH*, *98*, 231–242.

Stoakley, R., Conway, M. J., & Pausch, Y. (1995). Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of SIGCHI'95: SIG on Human factors in computing systems* (pp. 265-272).

Teller, S. J., & Séquin, C. H. (1991). Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Computer Graphics*, *25*(4), 61–70. doi:10.1145/127719.122725

Wingrave, C. A., Haciahmetoglu, Y., & Bowman, D. A. (2006). *Overcoming world in miniature limitations by a scaled and scrolling wim. In 3D* (pp. 11–16). User Interfaces.