PREFETCHING POLICIES FOR REMOTE WALKTHROUGHS

Christopher Zach, Konrad Karner

VRVis Research Center for Virtual Reality and Visualization
Inffeldgasse 16, 8010 Graz, Austria
{zach, karner}@vrvis.at

ABSTRACT

We present a 3D data streaming approach for remote walkthroughs, that integrates local optimization techniques for realtime rendering with prefetching techniques for remote scene graphs. Especially culling methods, that don't possess frame to frame coherence, can successfully be combined with remote scene databases, if the prefetching algorithm is adapted accordingly. We present a quantitative transmission policy, that takes the limited bandwidth of the network and the limited memory available at the client computer into account.

Keywords: Distributed/network graphics, 3D data streaming, remote walkthrough, realtime rendering

1 INTRODUCTION

The VRVis application research on virtual habitat aims on the automatic three dimensional, photorealistic modeling of urban, suburban and rural areas and on the organisation, simulation and visualisation of this data. The goal of one project at the VRVis center is the acquisition of high resolution data for the city of Graz. The facades of historical and otherwise prominent buildings will be modeled from terrestrial images and laser scanner data, and the less important buildings are represented by a block model extracted from aerial images. A user navigating through this virtual city should be able to view close-ups of detailed facades and to overview the whole city as well. An impression of the initial data set for the city of Graz is given in Figure 1.

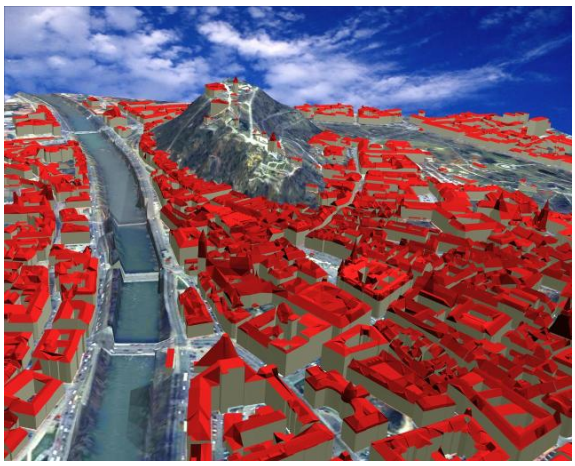

Figure 1: A view on the inner city of Graz

Since the data set used to visualize the City of Graz is far too large to fit into main memory, we decided to utilize an approach for remote walkthrough and flyover settings. Within our visualization system loading, indexing and server side caching of data is handled by a relational DBMS, that resides on a server machine. This server can operate several clients using network connections with different bandwidths.

On the client side visualizing complex urban data sets with current graphics hardware still needs a lot of rendering optimizations. If realtime rendering is targeted, the visual quality of the rendered urban scene is mostly limited by two factors: By the speed of the server in conjunction with the network bandwidth, and by the rendering capability of the client itself.

In order to maximize the visual quality within a realtime rendering setting we have to take these factors into account to guide the communication between the client and the server. Further we need to combine various rendering techniques such as level of detail management, impostor generation and rendering, visibility culling, efficient texture handling and others with techniques to retrieve relevant data from a huge data

set residing on a slow secondary storage.

We utilize an approach based on the estimation and optimization of certain quantities. This is in contrast to other approaches, that emphasize qualitative heuristics often based on a predefined sematics of the objects within the scene (e.g. area of interest [Hesin98], [Schma97]).

This work is a first step towards a visualization framework that fits our requirements.

2 BACKGROUND

2.1 Frame rate control

Even with current 3D graphics hardware rendering large data sets still needs acceleration techniques to achieve sufficient high frame rates. Acceleration schemes essentially choose the objects from the complete data set, that should be drawn in the next frame, and the appropriate rendering method, dependend on the current viewing parameters. In [Funkh93] a system that guarantees minimal frame rates is discussed.

Some of these techniques have an impact on the visual quality of the rendered scenes, others are lossless in this respect. More important we distinguish between acceleration methods, that comprise spatial coherence, if the viewing parameters change slightly, and methods that produce very different results for nearby viewing parameters.

A very basic rendering optimization is *view frustum culling*. Only objects, that are at least partially within the current viewing frustum, need to be rendered. This technique directly depends on the viewing parameters.

In densely occluded scenes like urban walkthrough scenarious *occlusion culling techniques* [Telle91], [Cohen00] can further reduce the amount of geometry to be rendered without loss in the visual quality. The space of view positions is partitioned into view cells and the set of visible objects from every view cell is computed. Therefore for a given view position the set of unoccluded objects can be determined and afterwards combined with view frustum culling to obtain the set of actually visible objects. Occlusion culling is usually not very effective in fly over settings.

Culling methods reduce the amount of data to be rendered, but this can still be too much to guarantee the frame rate requirements. The remaining acceleration techniques choose the rendering method for objects, that assures a suitable visual quality. Unlike culling methods these techniques may reduce the visual quality of the rendered image. These methods represent a trade-off between image quality and rendering speed.

Level of detail management [Clark76] and image based rendering [Macie95] replace the object's full geometry with a representation, that is faster to render. Since our data set has a huge spatially extension, we employ the LOD-R-tree concept [Kofle98] for a combined indexing and LOD management system.

2.2 Rendering over a Network

In our opinion, rendering over a network, where the complete data set for the scenery resides on a remote computer, is not substantially different from rendering data sets, that are too large to fit into main memory and have to be paged in from a secondary storage. The remote rendering setting allows the data set to be shared among the client in contrast to the local rendering setting with paging.

Thus we view the server in the remote rendering setting as some kind of secondary storage with a bandwidth that is order of magnitudes slower than local hard disks.

The rendering system is faced with 2 decision problems: Which data should be retrieved over the network and what should be rendered from the directly available data in the client cache for the next frame? Basically these two challenges can be treated independendly, such that the requests to the server for data retrieval are not directly related to the needs of real-time rendering.

Nevertheless rendering acceleration techniques should guide the communication with the server.

2.3 Streaming for Remote Walkthroughs

Our framework is strongly inspired by the work done by Teler and Lischinsky [Teler01]. They formulated the remote walkthrough setting as an optimization problem as follows: For a known path of the viewer, optimize the visual quality along this path. More formally: Let the walkthrough start at time $t = 0$ and end at time T. For every object i there exists a set of different representations r_{ij}. These can be various levels of detail, impostors etc. For a given viewing parameter $\vec{v}$, that consists essentially of position and viewing direction, we can estimate the influence on the visual quality if r_{ij} is rendered. This quantity is called the *benefit* $b_{ij}(\vec{v})$ of r_{ij}. As in [Funkh93] the benefit is a product of the following factors:

- The *accuracy* is a measure, how well the rendering of the representation is compared to the rendering of the full model.
- The *visibility* determines, how much of the object is seen. Visibility determination is based on view frustum culling and occlusion culling.
- Not every object in the scenery has the same *importance* to the viewer. Usually the viewer focuses objects in the center of the viewport, but importance calculation depends strongly on the field of application.
- Switching between different representations for the same object may cause noticeable artifacts and therefore the switching should be as smooth as possible.

These factors are estimated with empirically found formulas based on the relative position of the object to the viewer. Details are given in [Funkh93] and [Teler01].

Since the representations are available at the client side only after retrieving it from the server, the set $R_i(t)$ of representations for object i at time t is limited:

$$\sum_i \sum_{j \in R_i(t)} d_{ij} \leq t, \tag{1}$$

where d_{ij} is the time needed to retrieve r_{ij}. Further we define the maximum benefit $b_i^t(\vec{v})$ of the representations of object i available at the client side at time t as

$$b_i^t(\vec{v}) = \max_{j \in R_i(t)} b_{ij}(\vec{v})$$

Assuming that the path $\vec{v}_t$ is known in advance, the task is now to optimize the cumulative benefit along this path:

$$\text{Maximize } \sum_i \int_{t=0}^{T} b_i^t(\vec{v}_t)\, dt \text{ subject to Eq. 1}$$

The solution of this problem is a sequence of retrievals. Since the representations are retrieved sequentially, the optimization problem can be reduced to a sequence of one-step decision problems: Let $t_0, \ldots, t_m$ be the decision times. At time t_k, maximize

$$\sum_i \int_{t=t_k}^{T} \max(b_i^t(\vec{v}_t) - b_i^{t_k}(\vec{v}_t), 0)\, dt, \tag{2}$$

again subject to Eq. 1. Eq. 2 is called the *added benefit integral*. Hence we will write $a \ominus b$ instead of $\max(a-b, 0)$ for convenience. Solving this problem the scheduler knows, which representation should be retrieved next. The greedy strategy for the scheduler is the selection of that r_{ij}, that maximizes the added benefit to cost ratio

$$\frac{1}{d_{ij}} \int_{t=t_k}^{T} b_{ij}(\vec{v}_t) \ominus b_i^{t_k}(\vec{v}_t)\, dt.$$

Since we don't know the true viewing parameters at time t, we utilize a prediction of the future path based on the motion of the user in the recent past. If we expect that the predicted path is valid for some time interval $[t_k, t_k + \delta]$, then the scheduler would choose that r_{ij}, that maximizes

$$\frac{1}{d_{ij}} \int_{t=t_k}^{t_k+\delta} b_{ij}(\vec{v}_t) \ominus b_i^{t_k}(\vec{v}_t)\, att(t - t_k)\, dt.$$

Here $att(\cdot)$ is some attenuation function, that possibly assigns less weight to the more distant future, thus expressing the lower confidence about temporally distant predictions. Since we deal with the uncertainty of the predicted paths in a different way as described in the next section, we assume, that $att \equiv 1$.

Within this framework it is implicitly assumed, that streaming over a network is the only limiting bottleneck in the visualization system. Neither the memory available to the client nor the rendering capabilities of the client influences the scheduling.

3 THE BENEFIT INTEGRAL REVISITED

A worst case scenario In this section we argue, that maximizing the added benefit integral in Eq. 2 is not sufficient for our urban visualization system. Imagine the following situation: The user is navigating on the ground and inspecting the facades in a close up view, thus most buildings are occluded. While navigating the user approaches the facade of a historical building and the predicted path suggests, that the texture of this particular facade should be refined. Since this texture is large, it needs some time to be transmitted over the network. In the mean time, the user possibly changes his intention and probably moves straight upwards above the roofs.

Now the scheduler is faced with two options:

1. Occluded objects are not retrieved during the navigation on the ground. This yields poor visual quality of the objects at some distance, if the new viewpoint is above the roofs, in case the network bandwidth is too small to retrieve the necessary objects fast enough.
2. Representations of occluded objects are retrieved during the pure walkthrough navigation. If the user continues his walkthrough, the visual quality of the facades is unnecessarily low.

The main point is, that the path prediction is not perfect and the user may change his path suddenly while a representation is retrieved. Since the path prediction becomes more uncertain in the more distant future, one may use some weighting in the benefit integral in Eq. 2. But such a weighting scheme doesn't help, since the only effect is, that long term planning is mostly inhibited and the scheduler is not guided to prefetch crucial data.

Augmenting the objective function We would like a scheduling policy with the following property: Maximize the visual quality along the predicted path while maintaining a sufficient quality, if the prediction fails. Thus we claim, that the *worst case benefit*

$$W^t(\varepsilon) := \min \left\{ \sum_i b_i^t(\vec{v}) : \vec{v} \in N(\vec{v}_t, \varepsilon) \right\} \quad (3)$$

is as close to some achievable benefit as possible. $N(\vec{v}_t, \varepsilon)$ is some neighbourhood of $\vec{v}_t$ discussed later. We will denote this constraint rather informally as

$$W^t(\varepsilon) \geq Q. \quad (4)$$

Eq. 2 together with Eq. 1 and this additional constraint for a suitable ε forms a new optimization problem. We can observe the following properties of this problem:

1. At startup time of the client ($t = 0$), no solution is feasible. Therefore an auxiliary problem has to be solved to satisfy the worst case benefit constraint Eq. 4.

2. Fulfilling the worst case benefit constraint the client would retain every representation it receives during the walkthrough. Thus the client may exhaust the memory resources and an additional constraint to limit the memory consumption by the client is required. Let m_{ij} denote the memory required for representation r_{ij} and M the total available memory at the client side, then we must postulate, that

$$\sum_i \sum_{j \in R_i(t)} m_{ij} \leq M. \quad (5)$$

   Note that this constraint is not a 'hard' one. It only expresses, that the size of the resident set of objects at the client side is limited. The scheduler is still free to discard already retrieved representations. The implication of this is considered in Section 4.

3. Q depends on the bandwidth of the network and the memory available to the client. If the bandwidth or the local memory are enlarged, the image quality even in the worst case should increase. Q depends on the current viewing position, too: In less complex regions it is simpler to approach the best achievable benefit than in regions with high complexity. Therefore Eq. 4 shouldn't be read literally.

Instead of optimizing the added benefit integral with three constraints, we lift the worst case benefit constraint into the objective function. The new objective function is now

$$\frac{\alpha}{\delta} \sum_i \int_{t=t_k}^{t_k+\delta} b_i^t(\vec{v}_t)\, dt + (1-\alpha) W^t(\varepsilon) \quad (6)$$

α is essentially a Lagrangian multiplier and needs to be determined by the appropriate min-max problem. Usually α will be assigned empirically, depending on the expected accuracy of the path prediction method. The added benefit integral is normalized (divided by δ) to obtain the expected gain along the predicted path.

The Neighbourhood $N(\vec{v}_t, \varepsilon)$ Since the next decision by the scheduler is made at time $t_k + d_{ij}$, the user has time $\varepsilon = d_{ij}$ to move completely in a different way than predicted or to look in a different direction. The viewing position and direction that can be reached in this time depends on the maximal translational velocity ϑ and the maximal angular velocity ω. All possible viewing parameters at time $t_k + d_{ij}$ are gathered in the set $N(\vec{v}_t, d_{ij})$.

The shape of $N(\vec{v}_t, d_{ij})$ depends on the interaction facilities of the viewing interface. E.g. if the user interface allows either to rotate the viewing direction or to move forward along a straight line, then the set of accessible viewing positions within period ε is part of a spiral shape.

Instead of using the true neighbourhood we enlarge this set, such that it consists of all viewing parameters, where the view point has distance less than $\vartheta\varepsilon$ to $\vec{v}_t$ and the viewing direction change is $\omega\varepsilon$ at most.

The Impact of the Augmented Objective Function Integrating the worst case benefit into the objective function has the benefit, that rendering optimizations can guide the scheduling, that don't have a frame to frame coherence. Consider view frustum culling: Modifiying the viewing direction changes the set of visible objects within few frames. Thus every remote rendering system requires some prefetching of objects, that are within an enlarged view frustum.

Essentially the same is true for occlusion culling. Yet occluded objects may become visible after a slight

change of the viewing position. To compensate for errors of the path prediction some kind of prefetching of potentially visible objects is needed again. Our objective function takes care of these strategies implicitly.

Of course, distance based LODs and impostered in the far field are much less sensitive to a failure of the path prediction as long as the maximum velocity of the viewer is rather limited.

4 THE GREEDY SCHEDULING POLICY

We ignore the problem of heap fragmentation and assume that the available memory M can be used thoroughly without any unused memory holes due to fragmentation.

Benefit Gains and Losses The increase of the benefit of object i after retrieving the representation r_{ij} is the benefit gain:

$$\Delta b_i^t = \alpha \Delta b_{ij}^t + (1-\alpha)\Delta \tilde{b}_{ij}^t, \tag{7}$$

where

$$\Delta b_{ij}^t = \frac{1}{\delta}\int_t^{t+\delta} b_{ij}(\vec{v}_t) \ominus b_i^t(\vec{v}_t)\, d\tau$$

and

$$\Delta \tilde{b}_{ij}^t = \min\{b_{ij}(\vec{v}) \ominus b_i^t(\vec{v}) : \vec{v} \in N(\vec{v}_t, d_{ij})\}.$$

Here Δb_{ij}^t estimates the expected added benefit along the predicted path and $\Delta \tilde{b}_{ij}^t$ measures the added benefit in the case of a completely wrong prediction. Discarding the representation r_{ij} by the client may result in a loss of $-\Delta b_{ij}^t$ in the estimated visual quality.

Selecting the Next Object to Retrieve Due to the memory restriction the retrieval of a new representation may require some representations to be discarded on the client side. If r_{ij} is a candidate for retrieval, the set D_{ij}^t of discarded objects must satisfy

$$M - \sum_{r_{i'j'} \notin D_{ij}^t} m_{i'j'} \geq m_{ij}. \tag{8}$$

The gain of receiving r_{ij} must be compared with the loss of discarding D_{ij}^t. Determining the optimal set D_{ij}^t yields to a nested optimization problem, therefore the following greedy algorithm is applied: Sort the representations resident at the client according to the estimated loss to memory size ratio $-\Delta b_{i'j'}^t / m_{i'j'}$ and insert them in ascending order into D_{ij}^t until Eq. 8 is satisfied. Often it may happen, that many representations resident at the client have a loss of zero. In this case an LRU (least recently used) scheme can be applied to discard objects, that are probably not useful in the near future.

To choose the representation to retrieve next, compute the net return

$$\Delta b_{ij}^t - \sum_{r_{i'j'} \in D_{ij}^t} b_{i'j'}^t. \tag{9}$$

If the net return is negative for all candidates, a download will result in a decrease of the objective function. Therefore the scheduler postpones this decision for a predetermined duration and no communication over the network takes place. Otherwise the scheduler initiates the retrieval of the representation with the best net return to transmission time ratio.

5 IMPLEMENTATION ISSUES

So far we didn't mention, wether the scheduler resides at the client or at the server. Since the server is responsible for multiple clients, we assume, that every client has its own scheduler and the transmissions are initiated by the client. This approach fits well with the use of a database management system on the server. Thus, we suppose that any data required by the scheduler (like the transmission times d_{ij} for every object) is available at the client side instantly. In real settings, this data will be sent over the network as well, but we ignore the impact of these transmission on the actual scheduling.

As already mentioned in 2.2 we propose an independent scheduler to achieve constant frame rates. Nevertheless the scheduler responsible for the network communication should somehow prefer representations that are faster to render. Imagine a distant landscape, that is modeled as a regular mesh and assume, that the scheduler may retrieve a corresponding imposter or the mesh itself, both with approximately the same transmission time and the same benefit. The impostor is usually preferable to speed up rendering of the frames. One heuristical solution is to augment the benefit b_{ij} with an additional factor, that expresses the geometric complexity to image quality ratio. An exact solution is not feasible, because it requires solving a nested optimization problem.

The evaluation of the added benefit integral is discussed in [Teler01]. It remains to describe the estimation of the worst case benefit. In Section 3 we already discussed the set $N(\vec{v}_t, \varepsilon)$ of viewing parameters, that can be accessed during navigation in the worst case. At first, estimation of the worst case benefit involves the computation of visibility. Since our

occlusion culling system is based on a precomputed PVS (potentially visible set from a view cell) algorithm, the set of objects visible from $N(\vec{v}_t, \varepsilon)$ can be estimated as the union the PVS of every view cell, that is within distance $\vartheta\varepsilon$ from $\vec{v}_t$. If $\omega\varepsilon < \pi$, view frustum culling with an enlarged frustum can be applied additionally.

Estimating the worst case accuracy of a representation within $N(\vec{v}_t, \varepsilon)$ depends on the type of formula for accuracy calculation. Usually the minimal benefit is realized on the boundary of $N(\vec{v}_t, \varepsilon)$ and independent of the viewing direction. Thus, we are faced with a minimization problem on a circle around the current viewing position $\vec{v}_t$, which often can be solved easily. E.g. the accuracy estimation for LOD representations in [Teler01] is based on the distance of the object to the viewing position. Therefore the worst accurary for LOD objects is attained at the closest position to the considered object within $N(\vec{v}_t, \varepsilon)$. The situation is quite similar for impostors, since their accuracy depends on the relative position of the viewer to the position the impostor was generated for.

So far we considered only representations for single objects. Especially far field impostors are not generated for individual objects but for several spatially related objects at once. Therefore impostors usually affect the benefit values of several objects and thus the framework has to be slightly generalized to take this into account. The same is true for our LOD-R-tree concept, since the transmitted LOD geometry may represent several buildings.

Finally we have to estimate the transmission durations d_{ij}. We assume, that d_{ij} correlates strongly with the size of the transmitted representation and we add a fixed duration for communication overhead. If M_{ij} bytes are transmitted for r_{ij}, then $d_{ij} = c\,M_{ij} + \tau$. M_{ij} must not be equal to m_{ij}, because transmitted data may be compressed or otherwise different from the memory representation. The two parameters c and τ can be estimated from few test transmissions done in the initialization phase of the client.

Spatial Indexing Benefit computation involves the determination, wether an object overlaps with an enlarged viewing frustum. Doing this with a linear search is not feasible for a large data set, thus an acceleration scheme for geometric queries must be employed. Usually some hierarchical method is utilized. We currently use an R-tree [Guttm84] for spatially organizing the scene. Due to the large size of the spatial index the full R-tree is located at a server, that is not necessarily identical with the scene database server. The client requests and caches parts of this tree, as it is required during traversal of the scene.

In the future this R-tree will be replaced with a read-only hierarchical index, that additionally stores LOD representations of its subtrees in its inner nodes. This way simplified geometry for objects can be retrieved during the traversal of the spatial index without the need to visit unnecessary leaf nodes.

This shortened traversal must be consistent with the scheduling policy. If the tree traversal stops too early an unsuitable coarse LOD representation might be returned. On the other hand, unnecessary visits of nodes should be minimized. The solution is quite simple: We assume, that the LOD representations stored at the inner nodes are visually perfect if viewed at least from a certain distance. This distance is known for every node in advance. The representation stored in an inner node is good enough, if the viewing position has at least this distance to the corresponding objects until the next decision is made, regardless of the movement of the viewer.

The same considerations must be taken into account for other spatially organized objects like view dependend ground textures and terrain meshes.

6 CONCLUSION AND FUTURE WORK

We have presented a framework for streaming scenes in a remote fly over setting. Currently this framework still is of theoretic nature and experiments to investigate the actual properties of the scheduler need to be done. Integrating this framework into our software is ongoing work.

Nevertheless this framework can serve as a starting point for prefetching algorithms part of remote rendering systems, since it is not specific to any field of application and contains general applicable quantities.

Further work is needed to tackle the visual artifacts, that appear, whenever visible objects change their representation. Currently we are uncertain if hysteresis reduction can be done solely within our framework or if it is beyond the scope and should be handled separately. Using a hysteresis factor in the benefit value the scheduler will prefer less drastic changes in representations, but ignores the fact, that representations should be allowed to change significantly for currently invisible objects.

The integration of partial traversal of spatial indices with benefit estimation is one step to reduce the time required by the scheduler itself. If the user navigates permanently, our initial implementation of the scheduler uses a lot of time to evaluate the benefit function. Whenever the user rests the system reuses al-

ready computed benefit values to speed up the decision process. If we need to reuse the benefit values in case of permanent movement, we can substitute ε (see section 3) with $\max(d_{ij}, D)$, where D denotes some duration. Thus, the benefits need to be evaluated every D time units at most. Perhaps additional acceleration schemes based on frame to frame coherence can be exploited to speed up scheduling.

This work has been done in the VRVis research center, Graz and Vienna/Austria (http://www.vrvis.at), which is partly funded by the Austrian government research program Kplus.

REFERENCES

[Clark76] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, 1976.

[Cohen00] D. Cohen-Or, Y. Chrysanthou, and C. Silva. A survey of visibility for walkthrough applications. In *Proc. of EUROGRAPHICS'00, course notes*, 2000.

[Funkh93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics*, 27(Annual Conference Series):247–254, 1993.

[Guttm84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Conference*, pages 47–57, 1984.

[Hesin98] G. Hesina and D. Schmalstieg. A network architecture for remote rendering. In A. Boukerche and P. Reynolds, editors, *Proceeding of Second International Workshop on Distributed Interactive Simulation and Real-Time Applications*, pages 88–91, July 1998.

[Kofle98] Michael Kofler. *R-trees for Visualizing and Organizing Large 3D GIS Databases*. PhD thesis, Technische Universität Graz, 1998.

[Macie95] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. In *Symposium on Interactive 3D Graphics*, pages 95–102, 211, 1995.

[Schma97] D. Schmalstieg. *The Remote Rendering Pipeline*. PhD thesis, Technical University of Vienna, 1997.

[Teler01] Eyal Teler and Dani Lischinski. Streaming of complex 3d scenes for remote walkthroughs. In *EUROGRAPHICS'2001 Annual Conference*, 2001.

[Telle91] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics*, 25(4):61–68, 1991.