Dynamic Primitive Caching for Haptic Rendering of Large-Scale Models

Mashhuda Glencross   Roger Hubbold   Ben Lyons

Advanced Interfaces Group
Department of Computer Science
The University of Manchester
UK
E-mail: mashhuda@manchester.ac.uk

Abstract

In this paper we present a software approach to managing complexity for haptic rendering of large-scale geometric models, consisting of tens to hundreds of thousands of distinct geometric primitives. A secondary client-side caching mechanism, exploiting partitioning, is used to dynamically update geometry within the locality of a user. Results show that the caching mechanism performs well, and that graphical rendering performance becomes an issue before the caching mechanism fails. The performance penalty of the caching technique was found to be dependent on the type of partitioning method employed.

1 Introduction

Haptic rendering of massive models poses a major challenge: how to achieve a rendering speed of 1KHz [2], or better. Even with today's fast processing speeds, the number of geometric primitives that can be rendered must be reduced to a small subset of the complete model. It is attractive to be able to haptically render large-scale virtual environments for computer aided design, path planning, evacuation planning and similar applications.

Many common APIs, such as GHOST [4] use fast bounding box tests to reduce the haptic rendering cost. For haptic display of large-scale models, researchers Raymaekers and Coninx [3] and Acosta and Tempkin [1] each discuss the limitations of this approach. In preliminary trials we found that stable haptic rendering of the GHOST scene graph is not possible beyond 500 distinct primitives. This is consistent with Acosta et al.'s [1] findings, meaning that haptic rendering of large-scale models through GHOST alone is not easily viable. The use of bounding boxes and hierarchical processing alone is insufficient for haptic rendering of large-scale models. Our approach decouples the choice of which primitives to consider from the haptic scene graph dynamically re-creating the haptic model.

In subsequent sections we describe the *haptic cache*, and results from a number of large-scale virtual environments containing between 2,000 and 120,000 distinct geometric primitives. We conclude with observations and performance metrics from this study.

2 The haptic cache

The haptic cache transparently provides a secondary client-side buffer, within which geometric primitives in the locality of the haptic device's end effector are automatically maintained. The size of the cache and the number of primitives that can be supplied to the haptic server are configurable, as different haptic APIs provide very different limitations. The state of the haptic server-side model is dynamically maintained on-the-fly by the cache which enables/disables or creates/destroys geometry, thereby minimising the number of primitives that need to be haptically rendered.

Large-scale models are partitioned into one of two possible forms, a coarse uniform voxel grid cell, or hierarchies of axis aligned bounding (HBB) volumes. These are used to dynamically determine a possible set of geometric primitives in the user's *locality* and populate the cache. If required, the number of primitives is further reduced using proximity queries.

3 Case Studies

In this section we present a case study application capable of haptically rendering large-scale virtual environments using the cache. We profile the application by comparing frame rates against the size of model, with and without haptic feedback. We also compare the performance penalty associated with using voxel grids and HBBs.

Screen-shots from two models of process plants show primitives being activated and de-activated as the user navigates through the environment. The model shown in

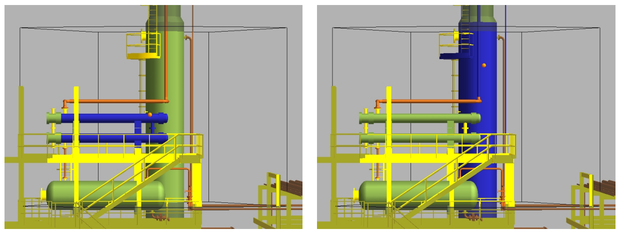

Figure 1: Interaction sequence with a small model

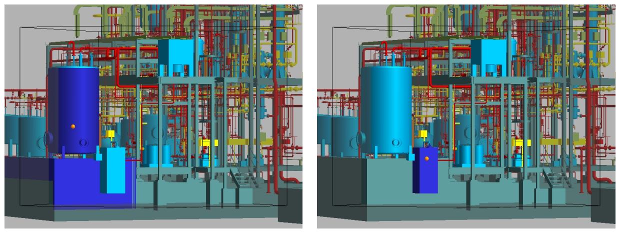

Figure 2: Interaction sequence with a large model

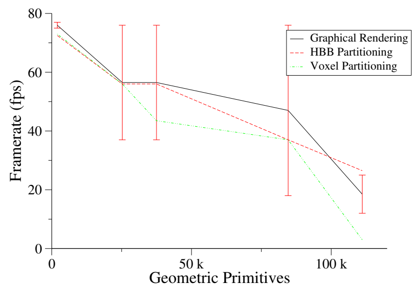

Figure 3: Performance graphs

Figure 1 contains just 1,974 distinct geometric primitives while the second in Figure 2 contains 25,230. The user may haptically interact with any of the primitives in the locality. Performance metrics are presented for a range of models containing up-to 111,110 distinct geometric primitives (approximately six million triangles).

Both sequences of images show the user moving the end-effector (rendered as a small sphere) through the environment. As geometric primitives are enabled they are highlighted, reverting to their original colour when disabled in the haptic server.

3.1 Performance metrics

The graph shown in Figure 3 plots performance for purely graphical rendering and graphical/haptic rendering using HBBs, and voxel grids to populate the cache. Error bars indicate the range of frame-rates achieved for a particular number of geometric primitives. Since the renderer makes use of view culling techniques to minimise the rendering performed for each frame, this range is broad.

Performance is shown to slightly degrade through the use of the cache in conjunction with HBBs. Initially, the voxel-grid approach seems to outperform the HBB approach but for larger models the computation overhead of the proximity queries has a notable negative impact on frame-rate. The number of queries needing to be performed depends on the granularity of the partitioning scheme and this situation can be improved by increasing the number of subdivisions. However, this comes at a significant additional cost to the memory footprint. For large-scale model rendering, this additional memory overhead is critical and cannot be afforded. Therefore, the primitive based partitioning scheme using HBBs is preferable.

4 Conclusions

We have shown that a secondary haptic cache can be used to manage the complexity of large-scale haptically enabled virtual environments which are too complex to be handled in their entirety by haptic APIs. We use two partitioning schemes one being spatial and the other being primitive based. HBB partitioning scheme is shown to be more suited to large-scale model rendering compared to the voxel grid approach due to the lower dependency on granularity of the partitions and associated memory footprint. We have shown this result quantitatively with models containing under 120,000 distinct geometric primitives.

References

[1] Eric Acosta and Bharti Temkin. Scene complexity: A measure for real-time stable haptic applications. In *Sixth PHANToM Users Group Workshop*, Aspen, CO, USA, 2001.

[2] Grigore C. Burdea. *Force and Touch Feedback for Virtual Reality*. John Wiley and Sons, Inc., 1996.

[3] Chris Raymaekers and Karin Coninx. Improving haptic rendering of complex scenes using spatial partitioning. In *Eurohaptics*, pages 193–205, Dublin, July 2003.

[4] SensAble Technologies. GHOST software development kit. Online document – active on October 2004. ⟨`http://www.sensable.com/products/phantom_ghost/ghost.asp`⟩.