GPU Implementation of 3D Object Selection by Conic Volume Techniques in Virtual Environments

Tobias Rick* Anette von Kapri† Torsten Kuhlen‡

Virtual Reality Group
JARA, RWTH-Aachen University

ABSTRACT

In this paper we present a GPU implementation to accurately select 3D objects based on their silhouettes by a pointing device with six degrees of freedom (6DOF) in a virtual environment (VE). We adapt a 2D picking metaphor to 3D selection in VE's by changing the projection and view matrices according to the position and orientation of a 6DOF pointing device and rendering a conic selection volume to an off-screen pixel buffer. This method works for triangulated as well as volume rendered objects, no explicit geometric representation is required.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

1 INTRODUCTION

Object selection is among the basic building blocks of every interactive virtual reality system. It can be described as the process of identifying an object that is subject to further actions. Selection by ray-casting is done by pointing at objects with a virtual ray. The closest object is selected. In VEs, the virtual ray is usually attached directly to a 6DOF sensor controlled by the user and subject to hand jitter. The selection by volume approach is used in situations where precise pointing is not required or not possible. Such techniques extend ray selection by projecting a volume, for instance a cone into the world. Objects that lie inside the volume are candidates for selection.

Common implementations of object selection use bounding boxes or spheres to represent objects. The nearest bounding volume which is touched by the selection ray or volume is considered for selection. Bounding volumes may be good in certain applications with widely spread objects but for non-convex objects (cf. Figure 1) that lie densely next to each other, a silhouette based selection may be more appropriate. However, efficient silhouette exact intersection methods, most likely hierarchical, usually require a huge amount of code for the seemingly rather simple task of object selection. Furthermore, in most cases selection ambiguity is introduced by occlusion and cluttering of 3D objects. The limitation to bounding volumes will increase this ambiguity even more. Therefore, we believe that it is important to provide an efficient and above all simple way to perform object selection based on exact object silhouettes.

We propose an implementation based on a selection technique that is well known on 2D desktop environments: select the object that corresponds to the pixel at the mouse cursor position. This technique can be implemented as a back buffer selection method. In a second rendering pass unique object identifiers are rendered to an off-screen buffer instead of colors and the pixel at the mouse position is read back from the graphics card. We adapt this method to 3D selection in VEs by changing the projection and view matrices of the off-screen buffer according to the position and orientation of a 6DOF pointing device. Hence, after the second render pass the off-screen buffer allows the computation of pixel based object statistics. State-of-the-art scoring schemes can then be applied to choose the active object.

Our approach is independent of any explicit geometric representation, meaning that it will work on triangular geometry as well as volume rendered objects. In direct volume rendering (DVR) the proxy geometry of the volumetric data is typically a very poor approximation to its visual representation, especially when large regions of the volume are transparent. Therefore, it is probably more intuitive to select only non-transparent voxels (silhouette) rather than the bounding box. We successfully applied our implementation to the selection of brain areas which are only poorly represented by bounding volumes. Our technique is appropriate here since these brain areas lie close to each other and tend to have a non-convex shape, see Figure 4.

The main benefits of the proposed method are: (1) It is easy to implement. Our approach uses the rendering function (which is already available) for visual representation *and* for selection computation. Additional implementation code is kept to a minimum. (2) It reduces selection ambiguity. Selection based on object silhouettes rather than on bounding volumes can significantly reduce occlusion problems, especially for objects of non-convex shape.

2 RELATED WORK

Although this work is not about inventing new selection metaphors but rather about the implementation we will discuss selection techniques in more detail since there is not much published on their implementation. However, from a rendering point of view our approach is quite similar to the process of shadow volumes [1] and other image-based multi-pass techniques.

An overview of selection techniques by pointing is described in [5]. The interacting by pointing metaphor can be divided into ray-casting [4] and selection by volume (e.g. flashlight) [8]. With ray-casting the user points at objects with a virtual ray that defines the direction of pointing and chooses the closest object on the ray. Selection by volume techniques choose candidates for selection among objects that lie within a volume and are the preferred choice in VEs due to their robustness to hand jitter. Recent work on how to assist the user in selection of objects can be found in but is not limited to [7, 6] or [9]. SenseShapes [7] are volumetric regions of interest that provide statistical information about the user's interaction with objects in the environment. Similar to our approach they use an off-screen buffer for selection computations. Haan et al. present the IntenSelect [6] which is a selection metaphor which concentrates on giving the user an appropriate feedback to their selection action. Their method uses bounding spheres to calculate conic volume intersections. Steed [9] develops a more general model for scoring schemes that define how the active object is chosen from a list of candidates.

*e-mail: rick@vr.rwth-aachen.de
†e-mail: kapri@vr.rwth-aachen.de
‡e-mail: kuhlen@vr.rwth-aachen.de

IEEE Virtual Reality 2010
20 - 24 March, Waltham, Massachusetts, USA
978-1-4244-6238-4/10/$26.00 ©2010 IEEE

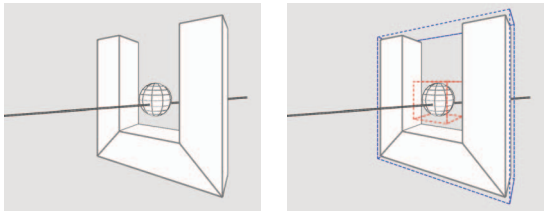


Figure 1: The bounding box of the sphere is contained within the bounding box of the non-convex U-shaped geometry. Selection based on bounding volumes would create selection ambiguity where a silhouette based approach would not.

Ulinski et al. [10] uses a two handed selection technique for the selection of volumetric data. With the two hands the selection volume (a six sided box) is manipulated and each voxel within this volume will be selected. In [3] Bornik et al. present a hybrid user interface to manipulate volumetric data. A special input device combines 2D interaction with a stylus tip and optical tracking to perform manipulation in 3D.

3 METHOD

After giving a rough overview of our proposed implementation technique we will describe the algorithm in more detail.

3.1 Overview

The basic idea behind our algorithm is to set up the camera settings such that the scene is seen through the eye of the pointing device. This is similar to the well known 2D desktop picking metaphor: select the object that corresponds to the pixel at the mouse cursor position. Our implementation uses a so-called back buffer selection method. All selectable objects are rendered twice. The second render pass assigns unique object identifiers instead of colors. To adapt this method to 3D selection we change the projection and view matrices of the second render pass to correspond to the position and orientation of a 6DOF pointing device. The concept is illustrated in Figure 2. Hence, the perspective view frustum of the pointing device represents a conic selection volume. The opening angle of the selection volume can be adjusted by changing the field of view of the perspective projection. Visible pixels of the objects within the selection volume will be recorded by their identifiers in the off-screen buffer. State-of-the-art scoring schemes according to pixel based object statistics can then be applied to choose the active object.

3.2 Setup of the Conic Selection Volume

In order to draw a clear picture of the selection volume setup we introduce the relevant coordinate frames and transformation matrices (according to the OpenGL standard): Objects are first transformed from their local reference frame (object space) into a global reference frame (world space) by the transformation matrix M_{Model}. The world space is then transformed by M_{View} into eye space where the camera is located at the origin and viewing into the negative z-axis. A projection matrix transforms eye space into screen space. Let M_{Proj} be a standard perspective projection matrix and M_{VCP} the projection matrix of a viewer centered projection that accounts for motion parallax in a head-tracked VE. Furthermore, let $M_{Tracking}$ be the matrix that maps the origin and orientation of the tracked pointing device from eye space into world space.

We have two camera setups for rendering objects, one for visual representation and the other for rendering selection IDs. Let the point p be the coordinate of an object defined in local object space. Usually the transformation from object space into the screen space of the VE is achieved by

$$p' = M_{VCP} \cdot M_{View} \cdot M_{Model} \cdot p.$$

In selection mode the projection of p into the screen space of the pointing device is

$$p_s = M_{Proj} \cdot M_{Tracking}^{-1} \cdot M_{Model} \cdot p.$$

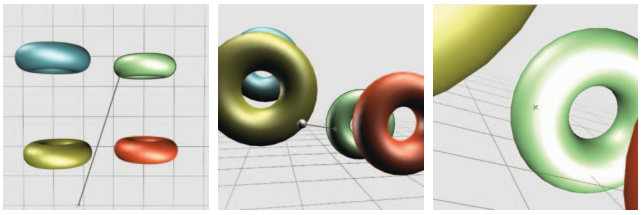


Figure 2: A pointing device selects the green torus. The scene as seen from above (left) and from the user's perspective (middle). The scene as seen from the pointing device (right).

3.3 Rendering Selection ID

Lets assume that the first render pass for visual representation looks like the following piece of pseudo code:

```
1. use_shader(fancy_lighting)
2. for_each(Object obj)
3.   obj.render()
```

A fancy lighting model is used (line 1) to render all objects for a nice visual impression (line 2 and 3).

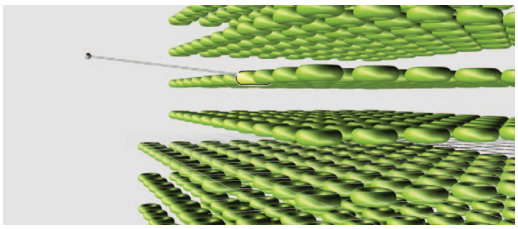
The second render pass for selection looks similar. We replace the lighting model by a shader program that writes selection IDs (line 4) and loads each object ID onto the GPU (line 6) prior to rendering:

```
4. use_shader(render_id)
5. for_each(Object obj)
6.   load_selection_id(obj.ID)
7.   obj.render()
```

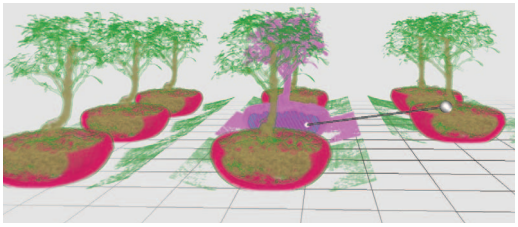
In an OpenGL implementation shader variables are commonly represented as so-called uniform variables which are usually expected to be constant over one rendering pass. Therefore, due to performance issues we will not use them. A better way to provide the shader with object specific information is to exploit unused rendering settings. We choose to use texture coordinates of the last texture unit (e.g. `GL_TEXTURE7`) that is used seldom and thereby will not interfere with the actual rendering process. When using this approach the fragment shader program for selection consists of a single line only: `gl_FragColor = gl_TexCoord[7].`

3.4 Create Pixel Based Object Statistics

The result of the selection rendering process is a two-dimensional pixel array P. Each pixel position contains a value that is either zero (no object) or an integer number (ID) that corresponds to an object. Since we set up the projection and transformations according to Section 3.2, the center pixel position c corresponds to the center of the selection cone. A popular scoring metric for instance used by [6] is the projected distance d between the cone center and an object at pixel position p which is $d = ||c - p||_2$. In [7] different rankings for object selection candidates have been introduced. Object statistics are created based on the time and stability of presence within the selection volume, distance to the volume's origin or center proximity. The pixel array P allows the computation of the above mentioned rankings, the choice of metric of course depends on user preference and/or application.



(a)



(b)

Figure 3: Setup of the torus (a) and bonsai scenario (b) as used for the performance benchmarks.

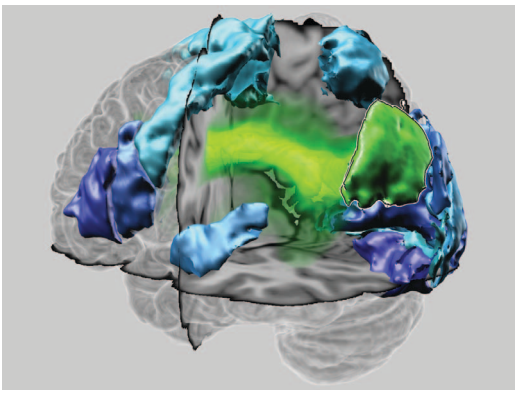


Figure 4: Visualization of the human brain (standard reference brain of the Montreal Neurological Institute) with selectable brain areas (triangular geometry) and a probabilistic fiber pathway (volumetric).

4 Extensions

We extend the basic algorithm by (1) describing how to get the actual contact point on the selected object and (2) how to select regions of volumetric objects that have been rendered with a volume rendering technique.

4.1 Contact Point on Selected Object

We have described how to find objects that are inside the selection volume. However, we do not know the exact contact point (CP) on the selected object which is often required in order to perform further actions or to provide visual feedback. We can exploit the structure of our selection implementation to avoid the computation of actual intersections. In fact, it is sufficient to save the three-dimensional vertex positions of the objects to the off-screen buffer additionally to the object ID. Hence, each pixel not only points to an object via an identifier but also to the 3D location that was mapped to the corresponding pixel position. The implementation on the GPU is straightforward, a vertex program which provides the subsequent fragment program (c.f. Section 3.3) with the 3D vertex location would contain the additional three lines of code (OpenGL Shading Language):

```
1.  float ID = gl_MultiTexCoord7.x;
2.  vec4 CP = gl_ModelViewMatrix*gl_Vertex;
3.  gl_TexCoord[7] = vec4(CP.xyz,ID);
```

4.2 Selection with Direct Volume Rendering

One advantage of using a pixel based selection implementation is that everything that can be rendered can be selected with little additional effort. Explicit geometric representations are not required. Therefore, our proposed method can also be used to select non-transparent or segmented parts of volume rendered objects. The transfer function (TF) must just to map density values to object IDs instead of colors. The volume renderer uses this alternative TF in the selection render pass. To ensure the uniqueness of object IDs, blending must either be disabled or the TF must provide alpha values with full opacity. Furthermore, the texture lookup to the TF must be done using nearest neighbor interpolation. Thus, the selection is restricted to parts that are closest to the selection cone. The rendering of the volume in the selection pass can be done at a much coarser resolution than the visual representation. This is achieved by simply increasing the sampling distance between texture slices. In practice one-fifths of the original number of slices turned out to be sufficient.

5 Results

For an evaluation of the usability of cone selection and scoring schemes we refer to [6] since we will analyze the performance of our implementation in terms of frame rates and computation time. We tested our method with both synthetic and real-world scenarios.

5.1 Benchmark Scenarios

In order to test the selection of triangular geometry we have created a synthetic scenario that consists of tori that are arranged in a regular grid, see Figure 3(a). The rendering command of the torus is available in the OpenGL library as `glutSolidTorus`. The geometric resolution of each torus has been set to 16 subdivisions for each radial section and for each number of radial divisions. Selection on volumetric datasets has been performed on the Bonsai data set [2]. The data set has a resolution of 256^3 at 8 bit precision. Similar to the tori scenario, the bonsai scenario consists of a regular grid of bonsai data sets as illustrated in Figure 3(b). Each bonsai has been rendered with a 3D texture based approach with $256 * \sqrt{2} = 362$ slices. Furthermore, we have tested our selection method in the real-world application of brain visualization (see Figure 4). The brain scenario consists of three major components, namely a reference brain that serves as context information (volumetric), brain areas (triangular geometry) and probabilistic fiber pathways (volumetric). The fiber pathways provide a probability distribution how likely specific brain areas are connected to each other. The data used for this visualization were obtained in the Institute of Neuroscience and Medicine of the Research Centre Jülich and the C. and O. Vogt Institute for Brain Research of the Heinrich-Heine-University Düsseldorf. The brain areas shown here were depicted from the Jülich-Düsseldorf cytoarchitectonic atlas[11].

5.2 Performance Analysis

We have analyzed the performance of our implementation based on the number of selectable objects and the corresponding frames per second. We have performed one benchmark without selection, thereby measuring pure rendering performance and one benchmark with selection at a fixed device position and orientation. Hence, the difference in frame rates hints to selection overhead and performance cost. Our benchmarks have been performed on an Intel Core 2 Quad CPU equipped with an NVIDIA GeForce GTX 285 graphics card. For rendering we have used a screen resolution of 1400×1050 pixels and for selection an off-screen buffer resolution of 16×16 pixels.

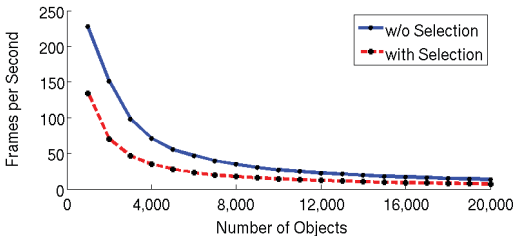


Figure 5: Performance benchmark with and without selection of triangular geometry.

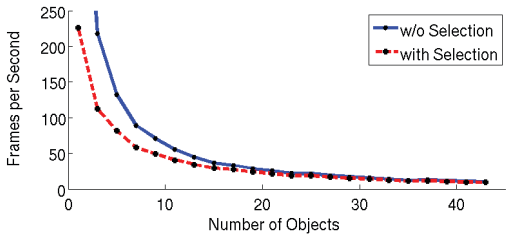


Figure 6: Performance benchmark with and without selection of volumetric data.

Torus Scenario (Geometric, Synthetic) The torus scenario consists only of triangular geometry, therefore the rendering performance is only limited by vertex processing. We have measured the frame rates that are obtained when rendering and selecting 1,000 to 20,000 objects as depicted in Figure 5. As expected, the frame rate has been cut in half when turning selection mode on since all objects have to be rendered twice. The low resolution of the off-screen buffer has no real impact because this scenario is geometry bound, the large amount of triangles is the major bottleneck. We guess that the culling of objects that do not lie inside the view and/or selection frustum would increase the overall performance considerately.

Bonsai Scenario (Volumetric, Synthetic) In contrast to the torus scenario, the bonsai scenario consists only of volumetric data. Volume rendering performance is mostly limited by fragment processing. Therefore, we have measured frame rates that are obtained when rendering 1 to 45 distinct bonsai data sets, see Figure 6. The performance overhead for the selection of a single data set is rather high. The switching between rendering to the screen and to the off-screen buffer has reduced the frame rates from 600 frames per second down to 200. However, the selection rendering of the volumetric objects itself can be performed at a much coarser resolution than the corresponding visual representation. Therefore, with an increasing number of volumetric objects, the selection overhead becomes almost neglectable. We conclude that if the rendering performance is limited by fragment processing (as in DVR) then the render pass for selection significantly benefits from the low resolution of the off-screen buffer.

Brain Scenario (Mixed, Real-World) The specific situation from the real-world brain application as seen in Figure 4 has eleven geometric brain areas corresponding to a total of 234,444 triangles and two volumetric data sets namely the context brain and one fiber pathway. Each volume has a resolution of $151 \times 188 \times 154$ data values at 16 bit precision. Rendering and selection can be done interactively. This specific configuration ran at 60 frames per second. This application benefits from the silhouette-based selection due to the fact that objects are mainly of non-convex shape and densely distributed.

6 Conclusion

We have presented a GPU implementation to accurately select 3D objects based on their silhouettes by a pointing device with 6DOF in VEs. The benefits of our approach are as follows: (1) It is easy to implement. Our selection approach uses the object's rendering function which is already available for visual representation so that additional implementation code is kept to a minimum. By exploiting the rendering performance of the GPU, the closest object on the ray is simply drawn to the screen, no intersections are actually computed. Thereby, we avoid the complex implementation of space partitioning hierarchies for efficient intersections of selection volume and objects. (2) It reduces selection ambiguity. Selection based on object silhouettes rather than on bounding volumes can significantly reduce occlusion problems, especially for densely distributed or non-convex objects that are poorly approximated by their bounding volumes. (3) It allows the selection of volume rendered objects. Our approach allows the easy integration of selection of non-transparent or segmented parts of volume rendered objects in the same way as triangulated geometry by using a different transfer function in the selection render pass.

The major disadvantage of our proposed implementation is the computational complexity which increases linearly with the number of selectable objects. In the worst case scenario this cuts the effective frame rate in half which has been the case for the torus scenario. The complexity could of course be reduced by culling selectable objects against the view frustum of the pointing device, but this would in turn increase the implementation complexity. However, for a reasonable amount of objects our approach worked fast enough in practice as has been demonstrated in the brain scenario.

References

[1] U. Assarsson and T. Akenine-Möller. A geometry-based soft shadow volume algorithm using graphics hardware. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 511–520, New York, NY, USA, 2003. ACM.

[2] Bonsai dataset courtesy of S. Roettger, VIS, University of Stuttgart. http://www.volvis.org/.

[3] A. Bornik, R. Beichel, E. Kruijff, B. Reitinger, and D. Schmalstieg. A hybrid user interface for manipulation of volumetric medical data. In *3DUI '06: Proceedings of the 3D User Interfaces*, pages 29–36, 2006.

[4] D. A. Bowman and L. F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *I3D'97: Proceedings of the ACM Symposium on Interactive 3D Graphics*, pages 35–38, 1997.

[5] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[6] G. de Haan, M. Koutek, and F. H. Post. IntenSelect: Using dynamic object rating for assisting 3D object selection. In *IPT / EGVE 2005*, pages 201–209, Oct. 2005.

[7] E. Kaiser, A. Olwal, D. McGee, H. Benko, A. Corradini, X. Li, P. Cohen, and S. Feiner. Mutual disambiguation of 3D multimodal interaction in augmented and virtual reality. In *ICMI '03: Proceedings of the 5th International Conference on Multimodal Interfaces*, pages 12–19, 2003.

[8] J. Liang and M. Green. JDCAD: A highly interactive 3D modeling system. In *Computers and Graphics*, volume 18, pages 499–506, 1994.

[9] A. Steed. Towards a general model for selection in virtual environments. In *3DUI '06: Proceedings of the 3D User Interfaces*, pages 103–110, 2006.

[10] A. Ulinski, C. Zanbaka, Z. Wartell, P. Goolkasian, and L. Hodges. Two handed selection techniques for volumetric data. In *3DUI '07: Proceedings of the 3D User Interfaces*, pages 107–114, March 2007.

[11] K. Zilles and K. Amunts. Receptor mapping: architecture of the human cerebral cortex. *Curr Opin Neurol*, 22(4):331–339, 2009.