Expanding Spheres: A Collision Detection Algorithm for Interest Management in Networked Games

Graham Morgan, Kier Storey, Fengyun Lu

School of Computing Science
Newcastle University, Newcastle upon Tyne, NE1 7RU, UK
Telephone: + 44 191 222 7983, Fax: + 44 191 222 8232
{Graham.Morgan, Kier.Storey, Fengyun.Lu}@newcastle.ac.uk

Abstract. We present a collision detection algorithm (Expanding Spheres) for interest management in networked games. The aim of all interest management schemes is to identify when objects that inhabit a virtual world should be interacting and to enable such interaction via message passing while preventing objects that should not be interacting from exchanging messages. Preventing unnecessary message exchange provides a more scalable solution for networked games. A collision detection algorithm is required by interest management schemes as object interaction is commonly determined by object location in the virtual world: the closer objects are to each other the more likely they are to interact. The collision detection algorithm presented in this paper is designed specifically for interest management schemes and produces accurate results when determining object interactions. We present performance figures that indicate that our collision detection algorithm is scalable.

1. Introduction

Interest management has been used to satisfy the scalability requirements of computer supported collaborative work (CSCW) [5] and military simulations [7]. These environments provide similar functionality to networked games by presenting geographically distributed users with access to a shared virtual world. Therefore, we assume that such techniques may be employed by networked games to promote scalability by lowering the volume of message passing required to ensure players receive a mutually consistent view of the gaming environment. Ideally, interest management limits interactions between objects that inhabit the virtual world by only allowing objects to communicate their actions to other objects that fall within their influence. This is achieved via the spatial division of the virtual world with message dissemination among objects restricted by the boundaries associated with spatial division. Objects exchanging the same set of messages are associated to a group. To achieve message dissemination in this manner a multicast service is commonly used to manage group membership (objects leaving/joining groups) and the sending of messages to group members. As the membership of such groups is achieved via consideration of

the geographic location of objects in the virtual world, a collision detection algorithm is required to aid in identifying an object's appropriate group membership.

Collision detection has been well studied in the literature and a number of algorithms have been proposed that perform better than $O(n^2)$ (e.g., [1] [2] [3] [4] [8]). Such algorithms commonly reduce the number of comparisons between objects via an ability to disregard many pairwise comparisons due to the large distances that may exist between objects. A number of these algorithms have been used to aid in determining appropriate object groups for interest management implementations [5] [6].

Collision detection algorithms are primarily designed to determine exact collisions between solid objects and are used to promote realism in virtual environments by making objects appear solid (e.g., objects should not pass through each other). However, in an interest management scheme it is common for the influence of an object to extend over the same virtual space that is occupied by other objects. Ideally, a collision detection algorithm suitable for interest management is required to identify the set of objects that may influence each other. This set provides the appropriate groupings for a multicast service. We believe that the development of a collision detection algorithm that performs better than $O(n^2)$ is required that is specifically designed for the purposes of interest management (i.e., identification of areas of the virtual world where area of influence of one or more objects coexist). Such an algorithm will provide the appropriate groupings of objects as expected by a multicast service and be able to more appropriately reflect the interest requirements of objects than existing collision detection algorithms.

The rest of the paper is organized as follows. In section 2 we describe our algorithm. In section 3 we discuss the performance of our algorithm and in section 4 we present our conclusions and future work.

2. Algorithm

We assume each object present in the virtual world to be a sphere. We consider these spheres to be auras associated with virtual world objects. An aura identifies an area of a virtual world over which an object may exert influence [5] [6]. All objects in the virtual world are members of the set *VR*. A collision between two objects, say O_a and O_b, is said to have occurred if O_a and O_b intersect (i.e., there exists an area of the virtual world that lies within the spheres of O_a and O_b). A *collision relation* (*CR*) identifies a set of objects that share, in part or fully, an area of the virtual world. Therefore, an object that belongs to a collision relation, say CR_i, collides with every other object that belongs to CR_i. Collision relations provide the appropriate groups that dictate the regionalization of the virtual world and provide a multicast service with appropriate group memberships. A set *SU* contains objects that are to be considered for collision; hence *SU* contains a subset of objects in the virtual world such that $SU = \{O_1, O_2, O_3, \ldots, O_n\}$. The determination of this subset is arbitrary, but will usually be equivalent to the full membership of *VR* at the start of the process of determining collisions. When an object, say O_a, from *SU* is being considered for collision it is said to be the *object under consideration* (*OUC* – when an object, say O_a, becomes the object under consideration we write O_a^{OUC}). A set *SC* contains the collision rela-

tions between objects previously identified as the *OUC*. A collision relation is an element of set *SC*. For example, if $SC = \{\{O_a, O_b, O_c, O_d\}, \{O_a, O_e\}, \{O_f, O_g\}, \{O_h\}\}$ we may state that SC contains four collision relations that are made up from the objects O_a, O_b, O_c, O_d, O_e, O_f, O_g and O_h. A 2D graphical representation of the collisions between these objects is shown in figure 1 (the scheme is applicable in 3D virtual worlds but we limit our diagrams to 2D for clarity).

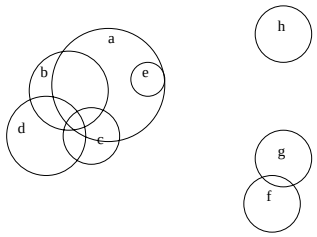

Fig. 1. Collision relations.

We now consider the identification of collision relations in more detail. We base our approach on determining if two spheres collide, say O_a and O_b, if the distance separating O_a and O_b is less than the sum of the radii of O_a and O_b. We assume an object's position vector and radius is known and may be described as O^{pos} and O^{rad} respectively. Additionally, our method also requires a *CR* to maintain position vector and radius information (CR^{pos}, CR^{rad}). CR^{pos} is taken from the first object identified in the *CR* and CR^{rad} is initially taken from the first object identified in the *CR* when the *CR* is created (i.e., *CR* only has one object within it). CR^{rad} is re-evaluated each time an object is added to a *CR*. Assume an object, say O_b, is to be added to a *CR*, say CR_i, the value of CR_i^{rad} is incremented using the following method:

1. Let *SD* be the separating distance between O_x^{pos} and CR_i^{pos}.
2. If $SD+O_x^{rad}$ is less than or equal to CR_i^{rad} then CR_i^{rad} remains unchanged.
3. If $SD+O_x^{rad}$ is greater than CR_i^{rad} then let CR_i^{rad} become $SD+O_x^{rad}$.

Extending CR^{rad} provides an opportunity to reduce the number of comparisons that need to be made in determining which objects intersect. We show how this can be possible by further considering the example shown in figure 2 when determining the collision relations of O_f (O_f is the *OUC*). Assume we have already deduced collision relations and calculated their associated CR^{rad} values for the first five objects considered in alphabetical order giving $CR_1 = \{O_a, O_b, O_c, O_d\}$ and $CR_2 = \{O_a, O_e\}$ (CR_2 is actually the true radius of O_a). It is clear that O_f does not intersect CR_1 and CR_2. Therefore O_f cannot intersect with any of the objects that lie within CR_1 and CR_2. By comparing O_f with CR_1 and CR_2 (two comparisons), we do not need to compare O_f with any of the objects within CR_1 and CR_2 (five comparisons). The pseudo code describing our algorithm is presented in figure 2.

Pseudo code main collision detection algorithm:

```
Algorithm CollisionDetection
Inputs  SU : Set of Objects;
Returns SC : Set of CRs // CR – Collision Relation;
Variables  Ox, Oi: Object; CL, CRy: CRs; newCRs: Set of CRs

Begin
 SC:= ∅;
 for each Ox ∈ SU do
  newCRs:= ∅;
  for each CRy ∈ SC do
   if (Ox collides with CRy) then
    CL:= ∅;
    /** Find colliding objects in CR **/
    for each Oi ∈ CRy do
     if (Ox collides with Oi) then CL:=CL ∪ {Oi} fi
    od
    /** Remove CRy if complete collision, add later **/
    if (card CL = card CRy) then SC:=SC\{CRy} fi
    if (CL ≠ ∅) then
     /** Turn CL into a CR by adding Ox **/
     CL:=CL ∪ {Ox};
     /** Add to set of new CRs for Ox **/
     /** Check for sub/super sets in newCRs **/
     newCRs:= addCR(CL, newCRs);
    fi
   fi
  od
  /** Add new CRs for Ox to SC **/
  if (newCRs = ∅) then /* Add singleton CR */
   SC:= SC ∪ {{Ox}};
  else
   SC:= SC ∪ newCRs;
  fi
 od
 return SC;
End
```

Fig. 2. Pseudo code describing the collision detection algorithm.

3. Performance Evaluation

Experiments were carried out to determine the performance of the expanding sphere algorithm. To enable comparative analysis of the performance figures a brute force collision detection algorithm with appropriate post-processing to determine collision relations was implemented. The number of comparisons is a count of the number of sphere-sphere (pairwise) intersection tests required to determine appropriate object group membership for all objects between two consecutive frames of animation. This measurement may not be influenced by implementation details (e.g., hardware configuration, memory management).

Object numbers were increased gradually from 1000 to 4000 with the number of comparisons recorded at each increment. The experiments were repeated with 3 different levels of aura coverage via the resizing of the virtual world. Each level identifies the percentage of the virtual world contained within auras given that no two auras

overlap (low density (5%), medium density (10%), and high density (20%)). Experiments were conducted on a Pentium III 700MHz PC with 512MB RAM running Red Hat Linux 7.2.

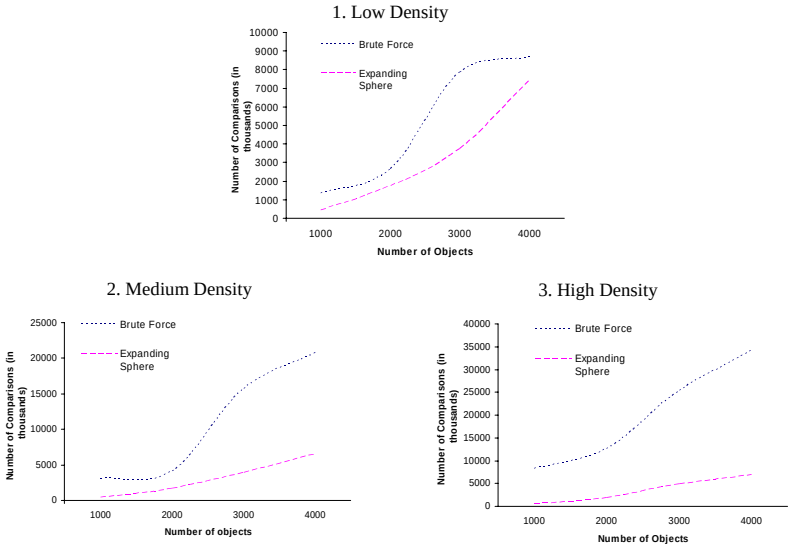

Fig. 3. Performance results.

An attempt is made to provide realistic movement of objects within the virtual world. A number of targets are positioned within the virtual world that objects travel towards. Each target has the ability to relocate during the execution of an experiment and objects may change their targets during the execution of an experiment Given that the number of targets is less than the number of objects and relocation and change direction events are timed appropriately, objects will cluster and disperse throughout the experiment. The auras of objects are uniform in size and their size does not change throughout the experiments.

Performance results are presented in figure 3. The first observation to be made is that expanding sphere outperforms brute force in all density types. Graphs 1, 2 and 3 show that expanding sphere carried out significantly less comparisons than brute force. This is particularly evident in medium and high densities. Furthermore, when object numbers increase to 4000 the number of comparisons performed by expanding sphere is approximately a sixth of the number of comparisons performed by brute force. This indicates that expanding sphere is scalable.

The performance increase provided by expanding sphere over brute force is not as noticeable in low density worlds as it is in medium and high density worlds. This can be explained by the fact that a low density world would have fewer collision relations with lower memberships, an environment that requires expanding sphere to carry out more comparisons (if there exists no aura overlap then expanding sphere will perform the same as brute force).

4. Conclusions and Future Work

We have presented a collision detection algorithm tailored to the needs of interest management schemes that rely on the aura based approach for determining object interaction. As we use aura overlap for determining spatial subdivision, our algorithm more accurately reflects the groupings of objects that may be interacting than existing collision detection algorithms. This has the added advantage that the number of groups that a multicast service is required to support when providing message dissemination services for interest management schemes is kept to a minimum. This is in contrast to existing interest management schemes that do not tackle this scalability problem. We have provided performance figures that show our algorithm to perform better than $O(n^2)$ and maintains such performance when large numbers of objects are present, indicating that our algorithm is scalable.

Future work will concentrate on further development of our algorithm. In particular, we want to apply our algorithm to our own interest management scheme [9].

Acknowledgements

This work is funded by the UK EPSRC under grant GR/S04529/01: “Middleware Services for Scalable Networked Virtual Environments”.

References

1. M. H. Overmars, “Point Location in Fat Subdivisions”, Inform. Proc. Lett., 44:261-265, 1992
2. P. M. Hubbard, “Collision Detection for Interactive Graphics Applications”, IEEE Transactions on Visualization and Computer Graphics, 1(3) 218-230. 1995
3. S. Gottschalk, M, C. Lin, D. Monocha, “OBB-Tree: A Hierarchical Structure for Rapid Interference Detection”, SIGGRAPH 93, p247-254, USA, 1993
4. M. S. Paterson, F. F. Yao, “Efficient Binary Space Partitions for Hidden-Surface Removal and Solid Modeling”, Disc. Comput. Geom., 5, p 485-503, 1990.
5. C. Greenhalgh, S. Benford, “MASSIVE: a distributed virtual reality system incorporating spatial trading”, Proceedings IEEE 15th International Conference on distributed computing systems (DCS 95), Vancouver, Canader, June 1995
6. C. Greenhalgh, S. Benford, “A Multicast Network Architecture for Large Scale Collaborative Virtual Environments”, Proceedings of European Conference on Multimedia Applications, Services and Techniques 1997 (ECMAST 97), Milan, Italy, May 1997, p 113 – 128
7. D. Miller, J. A. Thorpe. “SIMNET: The advent of simulator networking”, In Proceedings of the IEEE 83(8), p 1114-1123, August 1995
8. J. D. Cohen, M. C. Lin, D. Manocha, M. K. Ponamgi, “I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments”, In Proceedings of the 1995 symposium on Interactive 3D graphics, pages 189–196, 218. ACM, Press, 1995
9. G. Morgan, F. Lu, “Predictive Interest Management: An Approach to Managing Message Dissemination for Distributed Virtual Environments”, Richmedia2003, Switzerland, 2003.