Available online at http://www.journalcra.com

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

International Journal of Current Research
Vol.2, Issue, 1, pp. 078-085, January, 2011

ISSN: 0975-833X

REVIEW ARTICLE

ANALYSIS OF COLLISION DETECTION ALGORITHMS IN NON IMMERSIVE VIRTUAL WORLD

Merriliance, K[1]* and Mohamed Sathik, M[2]

[1]MCA Department, Sarah Tucker College, India.
[2.]Computer Science Department, Sadakkathullah Appa College, India.

ARTICLE INFO

Article History:
Received 11th September, 2010
Received in revised form
15th November, 2010
Accepted 25th December, 2010
Published online 6th January, 2011

Key words:

Virtual Environments,
Non Immersive VE,
Collision detection,
Bounding volume,
BV tree.

ABSTRACT

With the rapid development of virtual processing and internet technologies a great number of objects are being moving in every movement. Therefore it is necessary to develop the list of the closest features between all pairs of objects must be checked for collision. Collision checking must be performed on all object pairs because it is not known which objects will be collided in non immersive virtual environment. The set of object pairs that have to be checked for collisions is constructed based on the static model of the virtual objects that "co-exist" with one another. This paper presents an investigation of various collision detection algorithms and has been carried out the discussion of comparing these techniques. A user study of these was performed which revealed their characteristics and deficiencies of objects in the non immersive virtual world, and led to the development of a new class of techniques. These analytical studies provide distinct advantages in terms of ease of use and efficiency because they consider the tasks of collision detection in various objects in the VR world.

INTRODUCTION

Applications of virtual environments are becoming both more diverse and more complex. This complexity is not only evident in the number of polygons being rendered in real time, the resolution of texture maps, or the number of users immersed in the same virtual world, but also in the interaction between the user and the environment. Users need to navigate freely through a three-dimensional space, manipulate virtual objects with six degrees of freedom, or control attributes of a simulation, among many other things. Knowledge about the visual attention of users is a highly attractive benefit for information interfaces. In this paper we deal with collision detection techniques enhanced with the moving objects. Real-time collision detection of polygonal objects undergoing rigid motion is of critical importance in many interactive virtual environments. To achieve some degree of physicality and interaction in the VE, it is necessary to implement collision detection with respect to the walls during a walk-through of the

***Corresponding author:** *mmdsadiq@gmail.com*

scene, i.e., the user must not be permitted to pass through solid objects. In particular, simulation algorithms, utilized in virtual reality systems to enhance object behavior and properties, often need to perform several collision queries per frame. It is a fundamental problem of dynamic simulation of rigid bodies and simulation of natural interaction with objects (Cecilia Sik Lanyi *et al.,* 2006). We spepecially study their properties i.e. advantages and disadvantages which occur in the presence of virtual objects. With respect to the features they show, we adapt them to achieve the improvement of their properties. Finally, the evaluation of the new technique with a user study shows that it compares very favorably to conventional techniques. One of the main goals of using a VR system for assembly simulation is the potentially high degree of "reality" which can be experienced when immersed in a VE. Simulators might want to scale or shift a part while the system checks all relevant safety distances. In order to check serviceability of a part, the VR system has to track the work space necessary for the disassembly path and for the tools, and it has to report both intentional and "forbidden" collisions (Sebastian Knodel, 2008). In this paper, we analysis a simple but efficient view of collision detection algorithm that can be used in non immersive virtual world. The Virtual Environment can contain all kinds of quite physical objects, which are tangible, but whose appearance is constantly being altered as the setting changes. Things which are inside the Virtual Environment are known as objects.

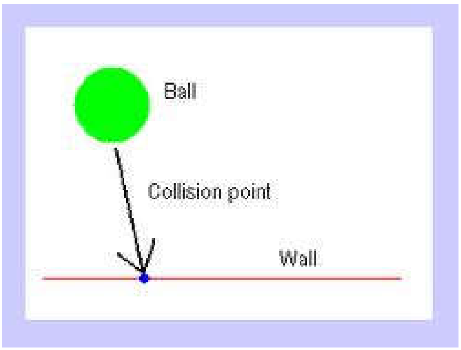

Fig.1. Collision occurs in virtual objects

Interaction means objects in the scene can be manipulated. In the real world, all objects are attached or connected to other objects. Objects may be moving. So the list of the closest features between all pairs of polygonal objects must be checked. Collision detection in virtual Environment involves algorithms to check for collision among objects. Simulating what happens once a collision is detected is sometimes referred to as "collision response" (Wu *et al.,* 2002). A feature corresponds to either a face or to one of its edges or vertices. Each object has its own elasticity coefficient. So that we have to find the elastic coefficients for all pairs of objects and compute from one frame to the next frame in the virtual Environment. The following observations are based on collision detection techniques. (i) In the non immersive virtual world, all objects are attached or connected to other objects. (ii) Objects do not interpenetrate each other. Several of these systems use collision detection to prevent interpenetration of objects. (iii)
Users seem to consider the entire area of visual overlap of the (moving) foreground object with the static Background scene when deciding where the object is.

A.Why collision occurs?

In a non Immersive Virtual Environment an object as a collection of planar patches, typically triangles, which together form a object. Figure 2 presents an outline of the entire collision system.

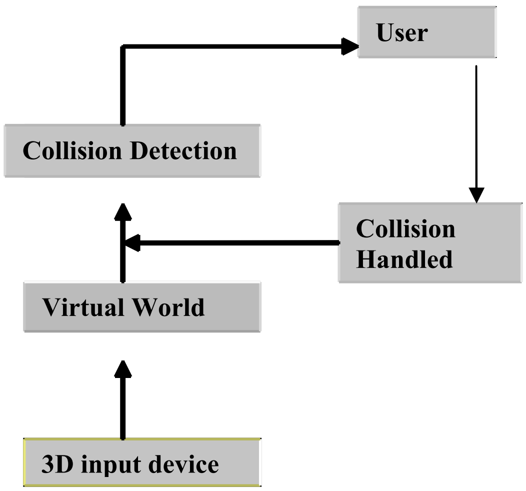

Fig. 2. Collision concept through 3D input device and presented to the user.

Collision detection is found out always in the following way:

```
loop {
Move objects by simulation
        Check collisions
        If collision occurs
        Take necessary action to avoid collision
//collision handled
 (Ex: highlight objects, do back-tracking)
    }
```

The collision detection method will determine which objects are intersecting and how they are intersecting. This information is then passed on to the collision Handled method which alters the objects. Once two objects have collided, a response needs to be computed. The second alternative again identifies the first surface behind the moving object, but ignores any surface closer to the viewer than the moving object. In this method, the moving object does not immediately pop out to the surface in front of the user, unless the moving object becomes the one closest to the viewer. When a small object moves forward, it may penetrate another object in front. When the distance between the features is less than a minimum tolerance value, the objects are considered to have collided. Once a collision is found, the object jumps also in front of the colliding object, as with the first alternative depending on the relative rotational speed of the object pair. Most of researches on collision detection have been concentrated on intersection check between the geometric objects in dynamic virtual world

II COLLISION DETECTION USING MINIMAL DISTANCE

The method is to track the minimal distance between two objects. That distance is realized by two features, where a feature denotes a vertex, an edge, or a polygon. Closest features mean the minimal distance between two objects. If a pair of features has been closest features during the last frame, then it is likely that the same pair is also the pair of closest features in the current frame (. If it is not, then the new pair of closest features is “nearby”, depending on how much the two objects rotated relative to each other. Given a pair of features, a new pair of closest features can be found efficiently. (see Fig. 3)

i.e $dist(P, Q) := \min\{|p-q| : p \in P, q \in Q \} > 0$ ----------------(1)
where P and Q are two objects.

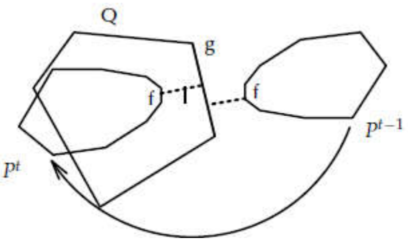

Fig. 3. Closest faces must be checked for being on the inside of the other object.

The idea is to replace the “closest-feature” criterion and to consider only faces. The distance dist (f, g) between two faces f and g is realized by two points which can either lie on their interior, or on the interior of an edge, or on a vertex. Select a pair of faces check the distance of all pairs of adjacent faces. If any of them has little distance than the current pair, on that area may have a chance to occur collision.

B.Collision handling method

This can help to increase the speed when each object has its own elasticity coefficient. Collision checking must be performed on all object pairs because it is not known which objects will be collided in virtual environment. This gives a maximum of n^2 collision pairs, where n is the number of objects.(see Fig. 4). Recall that the collision detection scheme requires a check to be performed between all possible object pairs. These checks are performed in parallel.

The set of object pairs that have to be checked for collisions is constructed based on the static model of the virtual objects that "co-exist" with one another. The number of object pairs is typically significantly less than n^2, where n is the number of objects in the merged environment [4]. If an object pair is determined to be in collision, the associated processor will compute the collision response. In addition for creating a geometrical model of the virtual objects that interact with the other objects must be created with equally realistic colors and textures. Proper calibration of the static virtual objects and their computer generated counterparts depends not only on a properly measured model. A typical head-mounted display, which a field-of-view of 60 degrees, will display the image shifted to one side by one sixth the display resolution!

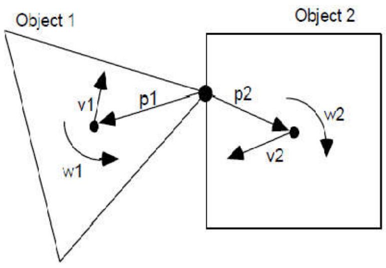

Fig. 4. Two objects colliding.

Another application is chaining several actions in a way which does not depend on time. Arrows show linear velocity v, angular velocity w and vector p from collision point to each object's center of mass [11]. A collision input is the "output" of the module for exact collision detection of objects can be changed by actions for loading, saving, deleting, copying, creating, and attaching objects or sub trees. Several actions can change object attributes like visibility, wire frame, and transformations. Others change material attributes, such as color, transparency, or texture. Transformation actions can be used to position an object at a certain place or to move objects incrementally[4]. Objects can be scaled interactively with the "stretch" action. When it is invoked, handles will be displayed at the corners and faces of the bounding box of the object. These can then be grabbed and will scale the object as they are moved. This action is quite useful to select a certain volume of the world or to create place holders from generic objects like spheres, cylinders, etc.

III Collision Detection using Bounding box

Real-time collision detection of polygonal objects undergoing rigid motion is of critical importance in many interactive virtual environments. In particular, simulation algorithms, utilized in virtual reality systems to enhance object behavior and properties, often need to perform several collision queries per frame. The internal representation of graphical objects has great impact on the choice of algorithms, not only for collision detection, but also for rendering, modeling, and many other parts of an interactive graphical system[10]. Several different approaches to object representation have been developed. The basic operation with almost all collision detection algorithms the basic operation is polygon/polygon intersection, or edge/polygon intersection. Basically, all collision detection algorithms are concerned with trying to avoid as much polygon/polygon tests as possible. Some of the polygon/polygon intersection algorithms gain their efficiency by reduction of dimensionality [5].

Polygon/polygon intersection tests can be reduced trivially to edge/polygon intersection tests by testing each edge of one of the polygons against the other polygon, and vice versa. Other efficient polygon/polygon intersection tests have been presented by. The idea of the former is to compute the intersection of the supporting planes, which is a line. Then, each polygon is intersected with that line, which yields two intervals. The polygons can intersect only, if the intervals on the line intersect. The idea of the latter is to compute the intersection of one of the polygons with the supporting plane of the other one, which yields a line segment. Then, the line segment is tested for intersection with that polygon in 2D. The bounding volume must be much simpler an object than the object it bounds. Cylinders and prisms seem to be simple. However, they are seemed to be too useful, even with static scenes, probably because their geometry is already too complicated. For curved surfaces, such as splines, curved bounding volume such as spherical shells seems to be a good choice[9]. Three-dimensional bounding volume is a useful tool of accelerating various geometric calculations including collision detection. In checking whether two objects intersect or not, bounding volumes make this collision test more efficient, especially when the objects do not intersect most of the time [5]. A bounding volume approximates an object by another simpler object that contains the original. Because bounding volumes are chosen to have much simpler topology and geometry than the original objects, checking the intersection between bounding volumes can be performed with a lower computational cost. A Bounding Box for an object is just a rectangular box in three dimensional spaces, with sides parallel to the coordinate planes, that contains the object. The choice is highly dependent of the shape of the objects to be bounded. (see Figure:5)For elongated objects, possible solutions include bounding ellipsoids and cylinder. Check every edge of polyhedron P if it intersects any of the polygons of polyhedron Q, and vice versa. Alternatively, one can check polygons against polygons [7]. From this very simple algorithm, it is obvious that edge/polygon or polygon/polygon intersection tests are the basic operation of all collision detection algorithms.

The algorithm consists of two modules:

1. The first one searches for a point in the intersection of the two objects
2. Compute the actual intersection by taking the union of these duals, and finally computing the dual of the result again, which yields the collision.

¥ e€ E : e ∩ P ≠ ø -------------------------------(2)

Which represents

$$\text{bbox}(P) \cap \text{bbox}(Q) \neq \emptyset$$

where p and q are virtual objects. Doing bounding box checks in the appropriate coordinate system is an application of the principle of problem transformation. Edge/poygon intersection tests, then it might seem inefficient to do polygon/polygon tests instead of edge/polygon tests,

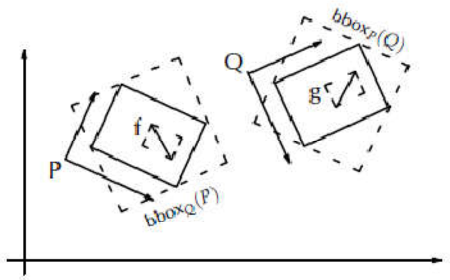

Fig.5. bounding box checks in the appropriate coordinate system

The pre-checks explained so far are bounding volume checks. Since such pre checks must be very efficient in order to gain anything. It is important that the bounding box checks are done in the most efficient coordinate system, i.e., by choosing the coordinate system carefully; we can save a lot of transformations. So, the collect-phase for Edges belongs to P and Q coordinate frame. That way, no vertex transformations are needed to compute bounding boxes of polygons. Convex polyhedra allow one to view the problem in many more different ways than arbitrary objects could offer: one can consider a convex polyhedron not only as a collection of polygons, edges, and vertices with certain properties, but also as the intersection of half spaces, or as the convex hull of its vertices. These are not different internal representations of convex objects, but merely different ways to look at the data. However, convex hulls can be utilized earlier in the collision detection pipeline for filtering out unnecessary exact collision checks and they might be useful for special applications. The efficient construction of convex hulls has been a long-time area of research in the field of computational geometry.

III COLLISION DETECTION USING BV TREE

Before a pair of objects is tested for intersection, a bounding box test is usually done. This is a conservative test of “non-intersection” on both sets of polygons as a whole. By taking this idea further, applying it recursively to polygon subsets of both objects, and using various types of bounding volumes, we arrive at the notion of bounding volume hierarchies and hierarchical collision detection[8]. The idea is to construct at initialization time a BV tree for each object. During run-time, when a collision query is to be processed for a given pair of objects, the transformations of the objects are applied to their BV trees By traversing the BV trees, regions of interest can be found more quickly; these regions are those parts of the objects which are “close” to one another.

C.Outline of the algorithm

The goal of any collision detection scheme is to discard quickly many pairs of polygons which cannot intersect. Assume boxes A and B overlap; if box A1 does not overlap with box B1, then we do not need to check any polygon enclosed completely by A1 against any polygon enclosed completely by B1.

A key functionality that a walk-through system must possess is view collision detection - detecting collisions between the user's viewpoint and the virtual objects to prevent the user from penetrating through the object. The sub-space decomposition algorithms such as these algorithms have been widely used for the view collision detection.(see Figure:6) These methods partition the entire bounding box into several subspaces (Ex a1,a2)so that at any given view position only a small fraction of sub-spaces is subject to be checked. We can use the box-box tests to prevent checking unnecessary polygon pairs[12]. Of course, this is done recursively, which yields a simultaneous traversal of two bounding volume hierarchies.

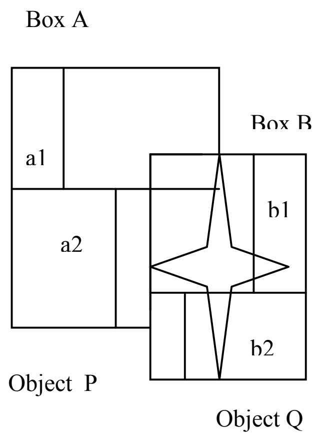

Fig. 6. Only faces and edges of overlapping boxes have to be checked for intersection.

The following steps outline the basic scheme of collision detection algorithms based on a bounding volume hierarchy:

```
1. Collision detection( ) a and b are the bounding volume of A's tree and B's tree a[i] and b[i] are the children of a and b, respectively.
2. If a or b is empty then return
3. Else if a and b have leaves process the next level polygons enclosed by a and b return
4. Else if a has not leaf and b has leaf then recursively checks all pairs of their children with b for collision.
5. Else if a has leaf and b has not leaf recursively checks all pairs of their children with a for collision.
6. Else a not leaf & b has not leaf recursively checks all pairs of their children with a and b for collision.
7. Return FALSE    // No collision occurs
```

Fig. 7. Pseudo code for the BV algorithm

With some BV schemes, it can be more efficient sometimes, if one of the children of a BV are empty. This can help to approximate the object better. The test between an empty and a non-empty BV is trivial. Usually, leaf BVs will contain exactly one polygon[6]. However, this is not inherent to the idea of hierarchical BV trees or simultaneous traversal. So, the data structure associated with hierarchical collision detection is bounding volume trees. Each node in such a tree is associated with a subset of the primitives of the object, together with a bounding volume that encloses this subset. Given two objects and the roots of their associated BV trees, a simultaneous traversal of the two trees recursively checks all pairs of their children BVs for overlap. If such a pair does not overlap, then the polygons enclosed by them cannot intersect.

D. Advantages of using BV Tree Algorithm

- BV tree techniques exploit the rasterization of objects for collision and self-collision detection.
- Higher performance.
- Easy to detect collisions.
- No pre-processing required.
- Suitable for rigid and deformable objects.
- To prevent checking unnecessary polygon pairs.
- To prevent the user from penetrating through the object.
- Provide simultaneous traversal of two bounding volume hierarchies.
- Discard quickly many pairs of polygons which cannot intersect.

- Regions of interest can be found more quickly because these regions are those parts of the objects which are “close” to one another.
- Checking the collision between bounding volumes can be performed with a lower computational cost.
- it is much easier to implement numerically.
- Very efficient due to its simplicity.
- This algorithm is very fast, because it nvolves only bounding box comparisons.
- The method is very simple and reveals a good real-time performance.

When we analysis these algorithms in the view of performing the collision test of average speedup on N (see Figure:7) BV tree algorithm is about 30 times faster than the other algorithms and Other object types yielded similar results with slightly different.

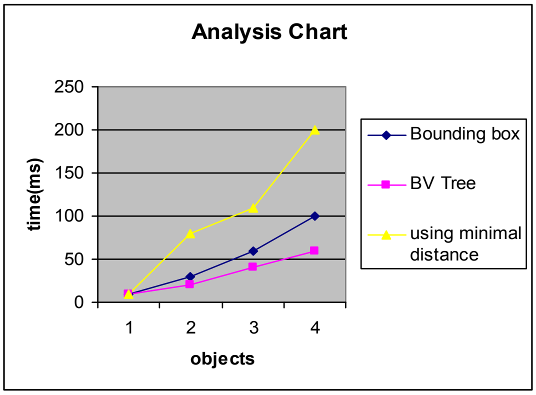

Fig. 7. Comparison of the bv tree algorithm with other algorithms of average speedup on N.

Conclusión

Hopefully the observations and lessons learned will benefit further research of such systems. Although this has not been an exhaustive depiction of all possible collision techniques, it provides a good exposure to many of the more common techniques currently in use. It is important to examine the collision techniques that are available and determine those that are best suited for the tasks that need to be accomplished. We have analysis these algorithms for collision tests between multiple objects in a non immersive virtual environment. The investigation resulted in new design guidelines that will allow for more usable design of non-immersive desktop, photo-realistic virtual environments. Furthermore, the study provides some new areas for future developments of usability evaluation methods. We discussed an implementation of the proposed algorithms, based on these guidelines, we present new forms of the collision detection techniques and collision handling method, which are augmented with positioning, selection and average cost of performing collision test feedback, to support within dense and occluded non immersive virtual environments. Our analysis indicated that our introduced visual feedback played the most critical role in aiding the selection task. In this paper, we perform the exact collision detection test in geometry polygons by taking advantage of its geometric processing capability. The overall approach makes no assumptions about the object's motion and can be directly applied to all geometrical models.

REFERENCES

Animation",Third Eurographics Workshop, Cambridge, England.

Cecilia Sik Lanyi, Zoltan Geiszt, Peter Karolyi, Adam Tilingerand Viktor Magyar, Virtual Reality in Special Needs Early Education, The International Journal of Virtual Reality, 2006.

Daniel G. Aliaga “Virtual Objects in the Real World”

Gabriel Zachmann “Virtual Reality in Assembly Simulation Collision Detection, Simulation Algorithms and Interaction Techniques”-

Gino Johannes Apolonia van den Bergen. Collision Detection in Interactive 3D Computer Animation. PhD dissertation, Eindhoven University of Technology.

Herman: “Virtual reality a new technology: A new tool for personal selection”. Journal of neurosurgery, 2002.

Lin M., Canny J., "Efficient Collision Detection for

Möller, T. and B. Trumbore, “Fast, minimum storage Ray triangle intersection,” in Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05), Los Angeles, Calif, USA, July-August 2005.

Moore M., Wilhelms J., "Collision Detection and Response for Computer Animation", Computer Graphics (Proc. SIGGRAPH), vol. 22, no. 4, pp. 289- 297, August 1988.

Regan, M. and R. Pose, “Priority rendering with virtual reality address recalculation pipeline,” Computer Graphics(Proc. of SIGGRAPH’94), pp. 155-162, 1999.

Sebastian Knodel.” Navidget for Virtual Environments” Proceedings of the 2008 ACM symposium on Virtual reality software and technology.

SonOu Lee, JunHyeok Heo and KwangYun Wohn” An Efficient Collision Detection Algorithm using Range Data for Walk-through Systems” Jean Griffin” Objects-First, Algorithms-Early with Bot World”.

Williams, G., McDowell, I. E. and M. T. Bolas. Human scale interaction for virtual model displays: A clear case for real tools. In Proc. of The Engineering Reality of Virtual Reality.

Wu, 2002. shin - ting, marcel abrantes, daniel tost, and harlen costa batagelo ”picking for 3d objects”.

Wu, X. 1992. A linear time simple bounding volume algorithm. In Graphics Gems III, David Kirk, Ed., chapter VI, pages 301–306. Academic Press, San Diego, CA.

****