VOL. 10, NO. 2, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

IMPLEMENTATION OF AXIS-ALIGNED BOUNDING BOX FOR OPENGL 3D VIRTUAL ENVIRONMENT

Hamzah Asyrani Bin Sulaiman[1], Mohd Azlishah Othman[1], Mohamad Zoinol Abidin Abd. Aziz[1] and Abdullah Bade[2]
[1]Universiti Teknikal Malaysia Melaka (UTeM), Durian Tunggal, Melaka, Malaysia
[2]Universiti Malaysia Sabah, Kota Kinabalu, Sabah, Malaysia
E-Mail: h.a.sulaiman@ieee.org

ABSTRACT

This paper describes a simple and straight forward implementation of Axis-Aligned Bounding-Box (AABB) for OpenGL 3-Dimensional (3D) virtual environment for games and simulation purpose. The implementation of AABB is conducted in OpenGL graphic library version 1.2 with C++ programming language by using Visual C++. The implementation could help young and begineer computer graphics student to master the implementation of basic Bounding-Volume (BV) for collision detection and other fields with related to the Axis-Aligned Bounding-Box (AABB).

Keywords: AABB, BV, virtual environment, OpenGL, 3D.

INTRODUCTION

Collision detection technique has been widely used in computer graphics and visualization areas especially in medical simulation (Kim, De *et al*. 2002; Gasson, Lapeer *et al*. 2004; Gan, Dai *et al*. 2008; Guiyun and Changzheng, 2010; Youngjun, Sang Ok *et al*. 2010; Vlasov, Friese *et al*. 2012; Lingtao, Tao *et al*. 2013), computer games (da S. Junior, Clua *et al*. 2010; Kaiqiang and Jiewu, 2010; Lu and Guofeng, 2010; Hanwen and Yi, 2011; Yanchun and Xingyi, 2011; Yue, Chang *et al*. 2011) and animation (Jing and Lixin, 1996; Ponamgi, Manocha *et al*. 1997; Dongliang and Yuen, 2000; Mezger, Kimmerle *et al*. 2002; Atencio, Esperanca *et al*. 2005; NengWen and Yunbo, 2008; Lu and Guofeng, 2010; Hanwen and Yi, 2011). The increasing demand for virtual environment application that previously helps researchers and programmers to understand better real world problems also helps computer games industries and animation to gain better profits for country economics.

Our research has proven to improve the game industries and simulation by having real-time simulation with outstanding capabilities of computer graphics area. Given that situation, collision detection research starts to develop various type of techniques that cater different needs of virtual environment applications. Medical simulation required a sophisticated accurate collision detection technique in order to have better accuracy in performing virtual surgeon by the doctors (Kim, De *et al*. 2002, Sulaiman, Bade *et al*. 2012).

Meanwhile for computer games development, a fast and approximately accurate collision detection is required in order to retain fast-pace realism situation when user or player play their games. For instance, distance computation could be enhanced (Kaiqiang and Jiewu 2010, Dyllong 2012, Jia, Chitta *et al*. 2012, Sulaiman, Othman *et al*. 2013) where the point of contact between at least two intersecting primitives can be calculated precisely (Chakraborty, Jufeng *et al*. 2008), and penetration depth for collided objects or primitives is improved (Jia, Chitta *et al*. 2012, Sulaiman, Bade *et al*. 2012, Othman, Noor *et al*. 2013). Those are some ways how they satisfy the virtual world by giving them real-world experiences in virtual environment world. Hence, this research intends to further improve current technique for narrow phase collision detection in order to maintain and sustain the need of real-world virtual environment behavior.

There were various types of technique proposed for each component of narrow phase collision detection such as distance computation, point of contact determination and penetration depth. In distance computation technique, according to (Jia, Chitta *et al*. 2012), I-Collide, Bullet, ODE, RAPIF, V-Clip, OPCODE, and PQP have been developed since the past decade to cater specific queries based on targeting applications. SOLID technique (Sulaiman, Bade *et al*. 2010) used Axis-Aligned Bounding-Box (AABB) Bounding Volume (BV) to perform collision checking; V-Collide (successor of I-Collide) from (Sulaiman, Bade *et al*. 2009) using hierarchical Oriented Bounding Box (OBB) and exact contact determination test between pair of triangles (nearly colliding triangles); and PQP library from (Suaib, Bade *et al*. 2009) took advantages of rectangular swept sphere (RSS) trees to perform distance computation.

Meanwhile for point of contact and penetration depth solution, (Sulaiman, Bade *et al*. 2009, Meiping, Liyun *et al*. 2010, Gong, An *et al*. 2011) used the foundation of Lin-Canny technique (Sulaiman, Bade *et al*. 2008) where the solutions provided fast response on narrow phase collision detection technique and still applicable until today's research (Heng, KunChao *et al*. 2012, Xinyu and Kim 2012, Ka-Wai, Kuen Hung *et al*. 2013, Landry, Henrion *et al*. 2013, Yu and Yamane 2013). (Sulaiman, Bade *et al*. 2010) proposed a proximity queries based on popular technique for narrow phase collision detection which are Gilbert-Johnson-Keerthi (GJK) technique for computing distance, point of contact and penetration depth between pair of convex objects or primitives. It used simplex-based formulation and construction style in order to find the corresponding nearly

VOL. 10, NO. 2, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

intersecting objects or primitives in unique ways and low computational cost.

PROBLEM BACKGROUND

In virtual world, two common objects properties exists which are solid object and deformable object. If the object is defined as solid object, it is set as nothing happens to the shape of the object even though some force might be applied to it (James 1988, Redon, Kheddar *et al*. 2002, Rahul and Adam 2006, Sulaiman, Bade *et al*. 2010, Liu and Kim 2013, Zhang, Kim *et al*. 2014). It is also called as rigid bodies where the object can only move, rotate and translate. Meanwhile, deformable models can be considered as any object that has change in shape after some forces have been applied. Clothes simulation, medical simulation, and some fracture simulation required the object to be designed as deformable models. By considering the rigid bodies' simulation, deformable models simulation is much slower and required higher computation cost compared to rigid bodies simulation(Gilbert, Johnson *et al*. 1988, Redon, Kim *et al*. 2004, Mendoza and O'Sullivan 2006, Chen, Ye *et al*. 2012, Wei and Jing 2012, Zheng and James 2012). Figure-1 shows the differences between rigid bodies and deformable models.

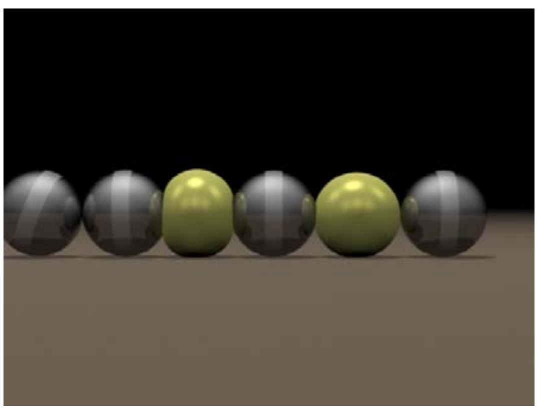

Figure-1.

Object intersection between rigid bodies has become one of the most important areas in order to bring realism to simulation or animation (Baraff 1989, Redon, Kheddar *et al*. 2002, Rahul and Adam 2006, Chang, Wang *et al*. 2009, Kwatra, Wojtan *et al*. 2010, Sulaiman, Bade *et al*. 2010, Sulaiman, Othman *et al*. 2013). Rigid bodies stand for geometric models that are fixed and assumed would remain static if and only there is some force being applied. In collision detection case, when two geometric models have collided, the system would notice that both objects cannot change its dimensions and sizes (Ningjun, Kai *et al*. 2011, Arcila, Dinas *et al*. 2012, Wei and Jing 2012, Taib, Othman *et al.* 2013, Ng, Ong *et al*. 2015). Any deformation to rigid bodies is ignored because of this behaviour and the collisions only affect its location or the movement of both objects (James 1988, Redon, Kheddar *et al*. 2002, Rahul and Adam 2006, Rocha and Maria Andre'ia Formico 2008, Zhang, Kim *et al*. 2014). Since the earlier era of 3D simulation and animation, problems prevailed in detecting object interference parts where numerous attempts by researchers have been made to find the solution of the collision detection between rigid bodies. Baraff has made one of the earliest moved concerning detecting object interference between rigid bodies (Baraff 1989).

BOUNDING-VOLUME

Performing collision detection between rigid bodies using primitive-primitive intersection checking required expensive computation cost. Some buildings might have thousands of polygons that need to be checked for collision when some other objects undergoing intersection test with the buildings. By using brute force approach, each primitive will be tested for collision until the program found the correct intersection point. So, one of the way to handle this is to use a Bounding-Volume (BV).

The purpose of using BV is to reduce the computational cost to detect object interference. If the object performs primitive-primitive testing without applying BV, it consumes longer time as it needs to check each triangle with other object triangle set (Klosowski, Held *et al*. 1998, Gottschalk 2000). However, time to check for each collision can be reduced through enveloping highly complex object with BV where the number of testing is hugely reduced. Instead of using single BV for one particular object in order to perform collision detection, the introduction of hierarchical representation called Bounding-Volume Hierarchies (BVH) could help performing collision detection better than a single BV. BVH provides a hierarchical representation that could split the single BV into certain level before performing primitive-primitive testing for accurate collision detection.

At the present time, there are several famous BVs such as spheres (Liu, Wang *et al*. 2007), Axis Aligned Bounding Box (AABB) (Weller, Klein *et al*. 2006, Zhang and Kim 2007, Tu and Yu 2009), Oriented Bounding Box (OBB) (Gottschalk, Lin *et al*. 1996, Chang, Wang *et al*. 2009, Tu and Yu 2009), Discrete Oriented Polytope (k-DOP) (Klosowski, Held *et al*. 1998), Oriented Convex Polyhedra (Bade, Suaib *et al*. 2006), and hybrid combination BV (Kockara 2007). Most of large scale 3D simulations used bounding box because of the simplicity, require less storage, fast response of collision, and easy to implement (Lin and Manocha 2004). Figure-2 illustrates the most commonly used bounding volume.

ARPN Journal of Engineering and Applied Sciences

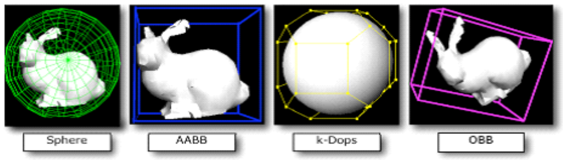

Figure-2.

FUNDAMENTAL CONSTRUCTION OF AABB

Axis-aligned bounding box (AABB) is the most common BV used for collision detection. It is a rectangular box with six surfaces which normally parallel with the standard axes. Also it can be represented as two points' minimum and maximum (min-max) in world coordinate space. There are three common representations for AABB (refer to Figure-3). The first representation starts by defining the min-max coordinate values along the standard axes. The second representation is using minimum corner point and the width or extended diameters are d_x, d_y, and d_z, from this corner. The last representation can be specified as a centre point *C* and half width extents r_x, r_y, and r_z along the standard axes. As stated by (Bergen 2004, Ericson 2004) that the centre-radius representation is the most stable representation and require less storage (memory) as compared to two other representations. To reduce storage requirements, AABB must utilize integers rather than floats and if the object only moves by translation, updating the last two representations is cheaper than min-max representation as it updates only three of six parameters (Ericson 2004).

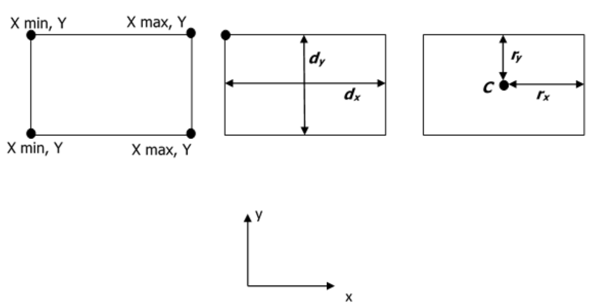

Figure-3. AABB representation: (a) Min-max; (b) Min-widths; (c) Centre-radius. X and Y coordinates system is shown in order to visualize the corresponding AABB into Cartesian coordinate in this example (source (Ericson 2004)).

In order to construct an efficient and fast AABB BV, several common rules must be followed or applied. Among these rules are (Ericson, 2004):-

- AABB must be realigned each time the object undergoes transformation update or rotation update.
- AABB must be tightly bounded to the object without any loose part.
- AABB creation must be automatic without user intervention each time new object is created by simulation. For example, each building in urban simulation must be automatically bounded with AABB without user need to manually enclose each building with their AABB.
- AABB creation must be fast and efficient in order to maintain the speed of simulation thus memory handling for storing AABB data structures must be handled correctly.

AABB CONSTRUCTION IN OPENGL ENVIRONMENT

In this stage, we have implemented standard Axis-Aligned Bounding-Box for our implementation. The construction of AABB starts by first reading the vertices from source object. From the total vertices read by our program, the maximum and minimum points that stored in vector points consists of x, y, and z values can be found simply by checking the most longest distance points among them. The first parameter to construct AABB in this implementation is to find the minimum and maximum points for each x, y, and z values of the corresponding object.

Creating AABB starts by declaring a class function for storing a minimum and maximum vertex point. Then, apply a function to enable the program to automatically iterate all the vertices point exists in the corresponding triangle or object in order to find the minimum and maximum point. It reads all the vertices for each triangle available inside the object or based on the triangle itself (for single triangle - 2D object). Figure-4 shows a pseudo code iterated the process of finding minimum and maximum vertex point.

1. For each triangle of the object
    a. Check a minimum point for x,y,and z axis
    b. Set a new minimum point for x,y,and z
    c. Check a maximum point for x,y, and z axis
    d. Set a new maximum point for x,y, and z
2. Repeat for all triangles

Figure-4. Continuous loops on finding the best maximum and minimum points for all axes in order to construct AABB based on the maximum and minimum points.

The pseudo code in Figure-4 works by looping through all vertices for each object. For example, if the object has a total of 1000 vertices, then the targeted AABB will compare the current minimum and maximum points with the current vertex that were read by the program. Hence, in order to read complex object, more times were needed to load the corresponding object into simulation. For the implementation wise, the minimum and maximum point's calculation will be used to construct AABB. Figure-5 shows how the minimum and maximum point for AABB construction was used.

VOL. 10, NO. 2, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

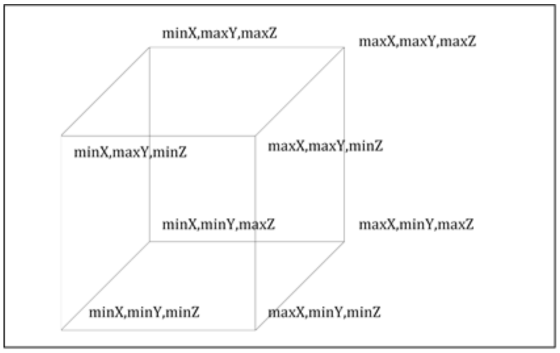

Figure-5. Coordination system in OpenGL environment for AABB setting.

Once the AABB minimum and maximum values are computed, the AABB bounding-volume can be drawn into corresponding object using GL_QUADS function in OpenGL. But first, the AABB must has its own shape that later on will be assigned with minimum and maximum value. AABB normally has eight vertices and six surfaces to draw (refer to Figure-6). Each vertex stored information of minimum and maximum value. Therefore, once vertices have been assigned with minimum and maximum points, the program will draw the AABB according to the minimum and maximum value taken from the enclosed object. Two multidimensional arrays are declared to store the variable from minimum and maximum points in order to draw the AABB. Figures 7 and 8 describe how the number representation is used to construct AABB with their pseudo code. There are six surfaces to be read from their number representation ranging from 0 until 7. The number is to represent the minimum and maximum for each axis of x, y, and z that will be assigned to draw AABB.

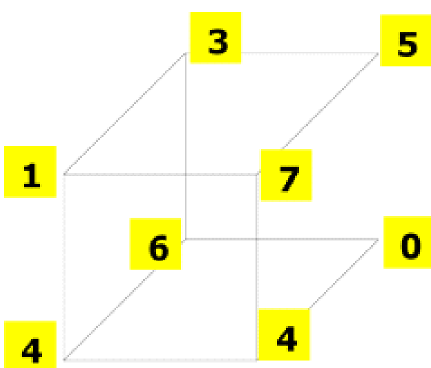

Figure-6. Number assignment for each of point in order to draw AABB in OpenGL environment.

```
GL QUADS Function
   For every i until i = 6, i++
       Draw AABB min and max point according to number system
       Complete draw the rectangle using glVertex3f command
   End looping For
```

Figure-7. Pseudo code to draw AABB BV.

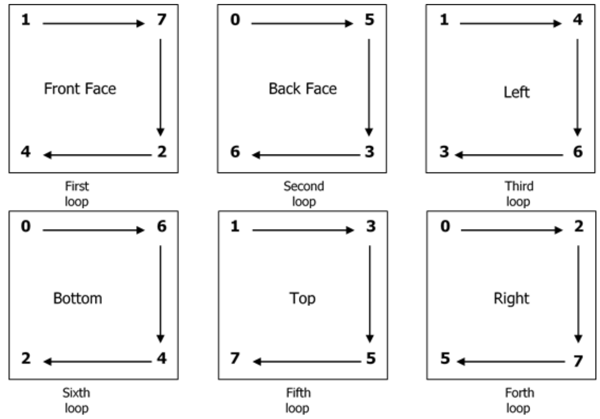

Figure-8. GL_QUADS function to draw six faces from AABB vertices list where each of corresponding arrow shown in the fiture represented the sequences of reading from OpenGL command glVertex3f.

By using GL_QUADS function, the AABB will be drawn using glVertex3f command in OpenGL using pseudo code in Figure-7. The glVertex3f read the targeted AABB vertices and connects each points becoming a shape of box that fitted the object based on minimum and maximum points. Since the AABB is a box that required six surfaces thus by using for looping system to repeat six times numeration, six surfaces will be drawn and combined it using GL_QUADS. For example, at the first loop, the GL_QUADS will draw right face using number assignment of each vertex. Figure-8 shows how these six points is connected using glVertex3f command.

In order to maintain the alignment of AABB bounding-volume each time object having transformation update such as rotation, AABB needs to be recomputed with its object orientations. Figure-9 shows a pseudo code with mathematical formula to calculate the orientation of transformation update.

```
Declare Variable tmpv
      vectorf tmpv;
Assign TempVector value according to all axes
      tmpv.x = v->x;
      tmpv.y = v->y;
      tmpv.z = v->z;
For every axis
      Set vector v with respective matrix m;
```

Figure-9. Transformation update for AABB.

By using an example of **P** which is an AABB and rotation matrix, **m**, the transformation updates work as follows:

- P affected by a rotation matrix m, resulting an orientation of previous P into P`.
- The first three column (m [0] to m [10]) of rotation matrix m representing the current local coordinates frame for P`.
- The vector v stores the P` new coordinates by multiply the corresponding rotation matrix m with current AABB P and thus giving AABB a new updated value

of minimum and maximum points. Figure-10 shows the realignment.

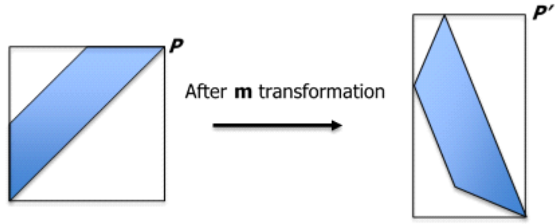

Figure-10. Realignment of AABB based on transformation matrix for coordinate P into P'.

CONCLUSIONS

This paper presented a brief detail on how the implementation of Axis-Aligned Bounding-Box (AABB) in 3D Virtual Environment using OpenGL graphics library. The implementation was successfully tested on Stanford 3D models.

ACKNOWLEDGEMENTS

The authors would like to thank Universiti Teknikal Malaysia Melaka and Ministry of Higher Education Malaysia for the financial support given through the research grant FRGS(RA CE)/2013/FKEKK/4F00197 (A Novel Interference Detection Algorithm Using Planar Subdivision of Axis-Aligned Bounding-Box for Virtual Reality Application).

REFERENCES

Arcila, O., *et al.* 2012. Collision detection model based on Bounding and containing Boxes. Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En.

Atencio, Y. P., *et al.* 2005. A Collision Detection and Response Scheme for Simplified Physically Based Animation. Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on.

Bade, A., *et al.* 2006. Oriented convex polyhedra for collision detection in 3D computer animation. Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia. Kuala Lumpur, Malaysia, ACM.

Baraff, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. Proceedings of the 16th annual conference on Computer graphics and interactive techniques, ACM.

Bergen, G. V. D. 2004. Collision Detection in Interactive 3D Environments. United States of America, Elsevier, Inc.

Chakraborty, N., *et al.* 2008. "Proximity Queries Between Convex Objects: An Interior Point Approach for Implicit Surfaces." Robotics, IEEE Transactions on 24(1): 211-220.

Chang, J.-W., *et al.* 2009. Efficient collision detection using a dual OBB-sphere bounding volume hierarchy." Computer-Aided Design In Press, Corrected Proof.

Chen, B., *et al.* 2012. Detection of Collision and Self-Collision Using QPSO for Deformable Models. Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on.

da S. Junior, J. R., *et al.* 2010. Fluid Simulation with Two-Way Interaction Rigid Body Using a Heterogeneous GPU and CPU Environment. Games and Digital Entertainment (SBGAMES), 2010 Brazilian Symposium on.

Dongliang, Z. and M. M. F. Yuen. 2000. Collision detection for clothed human animation. Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on.

Dyllong, E. 2012. "A Comparison of verified distance computation between implicit objects using different arithmetics for range enclosure." Computing 94(2): 281.

Ericson, C. 2004. Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology) (The Morgan Kaufmann Series in Interactive 3D Technology), Morgan Kaufmann Publishers Inc.

Gan, J.-H., *et al.* 2008. The Simulation of Benign Tumor Growth in Blood Vessels Circumstance. Computer Science and Software Engineering, 2008 International Conference on.

Gasson, P., *et al.* 2004. Modelling techniques for enhanced realism in an open surgery simulation. Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on.

Gilbert, E. G., *et al.* 1988. "A fast procedure for computing the distance between complex objects in three-dimensional space." Robotics and Automation, IEEE Journal of 4(2): 193-203.

Gong, J., *et al.* 2011. Research and Application for Collision Detection Algorithm in Virtools. Business Computing and Global Informatization (BCGIN), 2011 International Conference on.

Gottschalk, S., *et al.* 1996. OBBTree: a hierarchical structure for rapid interference detection. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM.

Gottschalk, S. A. 2000. Collision queries using oriented bounding boxes, The University of North Carolina at Chapel Hill: 174.

Guiyun, Y. and L. Changzheng. 2010. Application of VR in appendectomy surgery system. Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on.

Hanwen, L. and W. Yi. 2011. Coherent hierarchical collision detection for clothing animation. Haptic Audio Visual Environments and Games (HAVE), 2011 IEEE International Workshop on.

Heng, D., *et al.* 2012. Research and Implementation of Penetration Resolving for Multi-layered Virtual Garment Dressing. Digital Home (ICDH), 2012 Fourth International Conference on.

James, K. H. 1988. Realistic animation of rigid bodies. Proceedings of the 15th annual conference on Computer graphics and interactive techniques, ACM.

Jia, P., *et al.* 2012. FCL: A general purpose library for collision and proximity queries. Robotics and Automation (ICRA), 2012 IEEE International Conference on.

Jing, X. and Z. Lixin. 1996. Computing rotation distance between contacting polyhedra. Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on.

Ka-Wai, K., *et al.* 2013. "Dimensionality Reduction in Controlling Articulated Snake Robot for Endoscopy Under Dynamic Active Constraints." Robotics, IEEE Transactions on 29(1): 15-31.

Kaiqiang, G. and X. Jiewu. 2010. An Improved Algorithm of Collision Detection in 2D Grapple Games. Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on.

Kim, J., *et al.* 2002. Computationally efficient techniques for real time surgical simulation with force feedback. Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on.

Klosowski, J. T., *et al.* 1998. "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs." IEEE Transactions on Visualization and Computer Graphics 4(1): 21-36.

Kockara, S. H., T.; Iqbal, K.; Bayrak, C.; Rowe, Richard. 2007. Collision Detection - A Survey. IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC.: 4046 - 4051.

Kwatra, N., *et al.* 2010. "Fluid Simulation with Articulated Bodies." Visualization and Computer Graphics, IEEE Transactions on 16(1): 70-80.

Landry, C., *et al.* 2013. Task assignment, sequencing and path-planning in robotic welding cells. Methods and Models in Automation and Robotics (MMAR), 2013 18th International Conference on.

Lin, M. C. and D. Manocha. 2004. Collision and Proximity Queries. In Handbook of Discrete and Computational Geometry, 2nd Ed. Boca Raton, FL, CRC Press LLC. 35: 787-807.

Lingtao, Y., *et al*. 2013. Geometric modeling and collision detection based on hybrid bounding box in virtual gallbladder surgery. Complex Medical Engineering (CME), 2013 ICME International Conference on.

Liu, F. and Y. Kim. 2013. "Exact and Adaptive Signed Distance Fields Computation for Rigid and Deformable Models on GPUs." Visualization and Computer Graphics, IEEE Transactions on PP(99): 1-1.

Liu, L., *et al.* 2007. A Volumetric Bounding Volume Hierarchy for Collision Detection. 10th IEEE International Conference on Computer-Aided Design and Computer Graphics, 2007

Lu, C. and Q. Guofeng. 2010. Optimization of the collision detection technology in 3D skeleton animation. Computer Application and System Modeling (ICCASM), 2010 International Conference on.

Meiping, W., *et al.* 2010. A Hierarchical Collision Detection Algorithm for VA. Control and Decision Conference (CCDC), 2010 Chinese.

Mendoza, C. and C. O'Sullivan. 2006. Interruptible collision detection for deformable objects." Computers and Graphics 30(3): 432-438.

Mezger, J., *et al.* 2002. Progress in collision detection and response techniques for cloth animation. Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on.

NengWen, Z. and R. Yunbo. 2008. Real Time Dense Smoke Simulation Based Particle System. Intelligent Information Technology Application Workshops, 2008. IITAW '08. International Symposium on.

Ng, L. T., *et al.* 2015. Real-time monitoring and assessment of groundwater responses due to dewatering of an abandoned 7m deep excavation pit in Kuching City. Computer Methods and Recent Advances in Geomechanics - Proceedings of the 14th Int. Conference of International Association for Computer Methods and Recent Advances in Geomechanics, IACMAG 2014.

Ningjun, R., *et al.* 2011. Real-Time Collision Detection Based on Surgery Simulation. Intelligence Science and

Information Engineering (ISIE), 2011 International Conference on.

Othman, M. A., *et al.* 2013. An analysis of dielectric constant of pharmaceutical medicines using microwave radiation exposure. Proc. of 2013 3rd Int. Conf. on Instrumentation, Communications, Information Technol., and Biomedical Engineering: Science and Technol. for Improvement of Health, Safety, and Environ., ICICI-BME 2013.

Ponamgi, M. K., *et al.* 1997. "Incremental algorithms for collision detection between polygonal models." Visualization and Computer Graphics, IEEE Transactions on 3(1): 51-64.

Rahul, S. and L. Adam. 2006. Rigid body collision detection on the GPU. ACM SIGGRAPH 2006 Research posters. Boston, Massachusetts, ACM.

Redon, S., *et al*. 2002. "Fast Continuous Collision Detection between Rigid Bodies." Computer Graphics Forum 21(3): 279-287.

Redon, S., *et al.* 2004. Fast continuous collision detection for articulated models. Proceedings of the ninth ACM symposium on Solid modeling and applications. Genoa, Italy, Eurographics Association.

Rocha, R. d. S. and R. Maria Andre'ia Formico. 2008. An evaluation of a collision handling system using sphere-trees for plausible rigid body animation. Proceedings of the 2008 ACM symposium on Applied computing. Fortaleza, Ceara, Brazil, ACM: 1241-1245.

Shinar, T., *et al.* 2008. Two-way coupling of rigid and deformable bodies. Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Dublin, Ireland, Eurographics Association: 95-103.

Suaib, N. M., *et al*. 2009. On faster bounding volume hierarchy construction for avatar collision detection. ICCTD 2009 - 2009 International Conference on Computer Technology and Development.

Sulaiman, H. A., *et al.* 2012. Development of real-time virtual environment with hierarchical construction. Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, ICUIMC'12.

Sulaiman, H. A., *et al.* 2008. Bounding volume hierarchies for detecting collision in urban simulation. Computer Games and Allied Technology 08, CGAT 08 - Animation, Multimedia, IPTV and Edutainment, Proceedings.

Sulaiman, H. A., et al. 2009. Bounding-volume hierarchies technique for detecting object interference in urban environment simulation. 2nd International Conference on Environmental and Computer Science, ICECS 2009.

Sulaiman, H. A., *et al.* 2010. Balanced hierarchical construction in collision detection for rigid bodies. Science and Social Research (CSSR), 2010 International Conference on.

Sulaiman, H. A., *et al.* 2010. Fast traversal algorithm for detecting object interference using hierarchical representation between rigid bodies. 2nd International Conference on Computer Research and Development, ICCRD 2010.

Sulaiman, H. A., *et al.* 2013. Distance approximation using pivot point in narrow phase collision detection. Proc. of 2013 3rd Int. Conf. on Instrumentation, Communications, Information Technol., and Biomedical Engineering: Science and Technol. for Improvement of Health, Safety, and Environ., ICICI-BME 2013.

Taib, S. N., *et al.* 2013. An analysis of low pass filter using bowtie Defected Ground Structure (DGS) at 10 GHz for radar application. Proc. of 2013 3rd Int. Conf. on Instrumentation, Communications, Information Technol., and Biomedical Engineering: Science and Technol. for Improvement of Health, Safety, and Environ., ICICI-BME 2013.

Tu, C. and L. Yu. 2009. Research on Collision Detection Algorithm Based on AABB-OBB Bounding Volume. First International Workshop on Education Technology and Computer Science, 2009. ETCS '09. .

Vlasov, R., *et al.* 2012. Haptic Rendering of Volume Data with Collision Determination Guarantee Using Ray Casting and Implicit Surface Representation. Cyberworlds (CW), 2012 International Conference on.

Wei, Z. and S. Jing. 2012. Collision Detection Research for Deformable Objects. Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on.

Weller, R. e., *et al.* 2006. A Model for the Expected Running Time of Collision Detection using AABB Trees. Eurographics Symposium on Virtual Environments (EGVE), Lisbon, Portugal.

Xinyu, Z. and Y. J. Kim. 2012. k-IOS: Intersection of spheres for efficient proximity query. Robotics and Automation (ICRA), 2012 IEEE International Conference on.

Yanchun, S. and S. Xingyi. 2011. Research and improvement of collision detection based on oriented bounding box in physics engine. Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on.

Youngjun, K., *et al.* 2010. Mesh-to-Mesh Collision Detection by Ray Tracing for Medical Simulation with Deformable Bodies. Cyberworlds (CW), 2010 International Conference on.

Yu, Z. and K. Yamane. 2013. Ray-Shooting Algorithms for Robotics. Automation Science and Engineering, IEEE Transactions on 10(4): 862-874.

Yue, C., *et al.* 2011. Phusis studio: A real-time physics engine for solid and fluid simulation. Computational Problem-Solving (ICCP), 2011 International Conference on.

Zhang, X. and Y. J. Kim. 2007. Interactive Collision Detection for Deformable Models Using Streaming AABBs. IEEE Transactions on Visualization and Computer Graphics 13(2): 318-329.

Zhang, X., *et al.* 2014. Continuous penetration depth." Computer-Aided Design 46(0): 3-13.

Zheng, C. and D. L. James. 2012. Energy-based self-collision culling for arbitrary mesh deformations. ACM Trans. Graph. 31(4): 1-12.