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

Hamzah Asyrani Bin Sulaiman¹, Mohd Azlishah Othman³, Mohamad Zoinol Abidin Abd. Aziz¹ and Abdullah Bade²
¹Universiti Teknikal Malaysia Melaka (UTeM), Durian Tunggal, Melaka, Malaysia
²Universiti 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 beginner 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. 2002; Atencio, Esperanca et al. 2004; Gan, Dai et al. 2005; 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
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.

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.
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).

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.

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.

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.
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.

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.

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

1. P affected by a rotation matrix m, resulting an orientation of previous P into P'.
2. The first three column (m [0] to m [10]) of rotation matrix m representing the current local coordinates frame for P'.
3. 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](image)

**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**


Information Engineering (ISIE), 2011 International Conference on.


