Development of Real-Time Virtual Environment with Hierarchical Construction

Hamzah Asyrani Sulaiman
GRAVS Lab
Faculty of Electronic and Computer Engineering
Universiti Teknikal Malaysia Melaka
+6065552150
asyrani@utem.edu.my

Abdullah Bade
GRAVS Lab
School of Science and Technology
Universiti Malaysia Sabah
+6088320000
abb@ums.edu.my

Mohd Harun Abdullah
School of Science and Technology
Universiti Malaysia Sabah
+6088320000
harun@ums.edu.my

ABSTRACT
The development of real-time virtual environment is always a fundamental task for research to come out with a good testing procedure. Regardless any software application that has been used to develop the virtual environment, maintaining real-time aspect such as physic simulation, fluid simulation, collision detection, and others is definitely important. Numerous attempts has been introduced in order to develop nearly perfect virtual environment but at the end the solution only cater for some specific settings that must be implemented before we properly visualize the virtual environment. In this paper, we consider few elements that can be used to visualize their virtual environment and perhaps becoming a common visualization procedure to differentiate and compare with others.

Categories and Subject Descriptors
I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation;

General Terms
Algorithms, Design, Experimentation, Theory.

Keywords
Virtual environment, Simulation, Visualization, Bounding-volume Hierarchies

1. INTRODUCTION
In three-dimensional (3D) world, the number of objects used to represent the real world is varies according to their complexity of the application [1-6]. Some researchers might to develop for example an urban simulation that consists of few buildings with low polygon counts for each object while some wants to properly visualize the urban simulation with high detailed objects that the number of polygon for each object is high. The application itself determine whether the researcher or the designer needs to use high definition models or not in their design.

ICUIMC'12, February 20–22, 2012, Kuala Lumpur, Malaysia.

For each 3D object, the number of polygon, which is their primitives or triangles, is the key point to define whether the object is complex or not. By taken an example of 3D Stanford Bunny model, it has variety of polygon counts starting from the hundreds of polygons to the tens of thousands of polygons. The complexity of the 3D object is determined by their polygon count. When the same 3D object but with the different complexity is to be compared to each other, the highest polygon counts is the one that looks nearly fine with the real world object. However, in term of the resources used to visualize the same object with the different complexity, the low polygon counts is definitely the one that use low resources compared to the high polygon counts. Thus, the researcher and the designer must be able to comprehend with this matter in order to use virtual environment for their testing procedure depending on what type of application that they want to use. At one hand, the researcher might just want to test the fundamental testing of collision detection between rigid bodies model. Thus, different types of object complexity are used in order to analyze the speed and the accuracy of the detection. On the other hand, the designer might want to develop a virtual tourism application where user can navigate and walkthrough the scene. Hence, high detailed objects need to be used along with a culling system to cull an object that located far from the current user position. This is to minimize the amount of the resources used to visualize high detailed virtual environment. Los Angeles city by UCLA [7], Virtual Dublin [4] in 2003 to interactively visualize the urban area of Dublin, Ireland, and Virtual Nicosia developed by Dikaiakou et al. [8] in 2003 are examples of 3D virtual environments that have been setup for commercialization and research purpose. In this case however, we cannot properly measure the complexity of the whole virtual environment as each object in the virtual environment might vary according to the complexity of the application itself. Some object might have high polygon counts for example human, animals, trees while building might just have only low polygon count as they just use a texture to cover the 3D object.

In this paper, we discussed few elements that need to be considered to visualize real-time virtual environment with a normal frame rates per second (FPS), which is 30 FPS to 60 FPS. Section II described the elements in virtual environment, and section III discussed the development and step by step for good virtual environment. Section IV shows our proposed testing procedure for any 3D virtual environment and section V concluded our discussion.

2. VIRTUAL ENVIRONMENTS

2.1 Object Placement and Coordinate System

In virtual environment world, the coordinates system might defer than the real world. The virtual environment itself is just a very huge box containing many types of objects with different complexity and behavior. Usually, the researcher or the designer needs to propose specific size of terrain before the object needs to be placed. The terrain itself must be unique in order to realize it in virtual environment application. Some might develop it by using a program that could randomly generate the virtual environment terrain.

Object placement is done when the entire object has been properly designed using 3D editor such as 3D Studio Max, Autodesk Maya, or Google SketchUp software. Then, all the objects are copied into certain folder in order to let the program recognize the folder that it needs to read. The problem might occur where object placement required the user to be able to create a program that match the coordinates system that has been created by the 3D tools. Thus, it is best to confirm that our program has the same coordinate system with the 3D tools or otherwise we need to program it using our own coordinates system. It totally depends on how we are going to place every object in the virtual environment by integrating the object one by one or just load all the objects as a single entity. The programmer can do this process manually although it might time consuming for each object to be properly loaded into environment. It is acceptable practice to include all the static objects into one object (save the object as one entity using 3D editor) to save time.

The coordinates system varies for each type of virtual environment but it still use x, y and z coordinate system. Although the 3D editor define the distance between point A to point B with different distance when the object is loaded into virtual environment, we still can try to resemble the coordinate just like the one that we are going to use. For example, when we developed a building that 10 meters high, our 3D program that use to visualize the virtual environment cannot 100% visualize the same size even though we have properly set the parameters. The solution is to create a grid system in our "box" world which are 100 units high, 100 unit widths, and 100 units long and make it every unit as 1 meter x 1 meter. Then we can successfully integrate our building into the environment and resize it according to our specification. The same principle applies for each object that we have developed.

2.2 Texture Mapping

Texture mapping is one of important elements for researcher or designer to develop a well-designed 3D virtual environment. Instead of using library based coloring system provided for example, the OpenGL, the researcher or the designer has used textures in order to give few objects in the virtual environment feels just like the real world. Large amounts of researches have been conducted in just texture mapping area [9-12].

Given an example of a simple rectangle box, the process of giving the surface of each rectangle shape (six of them) with a few colors or texture is called texture mapping. By applying texture to the surface, we will create an astonishing object that looks like the real world entity. Each vertex of the box surface is assigned with a texture coordinate system either by manually added by the assigner or automatically define by the texture function in the 3D loading program.

One of the techniques for applying texture mapping is multitexturing. Multitexturing can be used to put more than one texture map into the object. Another technique is called bump mapping, which allows the texture to follow the 3D program lighting systems. It is useful for good appearance such as tree bark or rough concrete where it takes the lighting into details. It is popular technique used in computer and console game.

2.3 Lighting and Shadowing

Another important element of creating virtual environment is to have good lighting and shadowing system for the application. Although it depends on the quality of the texture that has been mapped into the surface of each type of object in virtual environment, a mechanism to handle the lighting system in order to build the aesthetic quality of the graphics themselves is useful.
In 3D virtual environment, lighting can be categorized into few types. The first one is the ambient light, it is a light that comes from all over the place and this is not happening in the real world. Some researchers might just want to add simple ambient lighting into their system by assuming the object get the light source in all direction.

Second type is a diffuse lighting system. It is a normal lighting system where we have only one source (for example: the Sun) and thus the object might have some shadow at the back of the opposite surface that facing the sun. The third type of the lighting is specular lighting system. It is more realistic where we have a source of lighting, and then there is some kind of mirror at the surface of the object that bounce off to some particular direction. It looks like polished stuff such as polished metals or glasses. The last one is the emission type lighting, where the object itself emitted the lighting and it is equally in all directions. All this type of lighting is refer from the OpenGL basic lighting system.

2.4 Level of Detail and Culling System

In virtual environment world that has high detailed objects and massive scale simulation, the designer or the researcher used a technique so called Level of Detail (LOD) to decrease the complexity of the far away object from the viewer current position [13-16]. Given an example of large-scale virtual environment of a big city, the position of the viewer might start at some point and at the same time, the viewer cannot see the far away object with the current position. By decreasing the object complexity using vertex transformations, we can limit the object polygon counts for any object that has different distance from the viewer. It is likely the viewer will not notice the different of the object when moving fast or in the far away position.

By working together with the culling system, we can cull away the object that is currently located at the back of the viewer screen eye or any object that located at the back of other objects. This will burst the speed of the simulation and improved the FPS rate

2.5 Collision Detection

Given the properties of two or more objects in virtual environment, in is vital to detect the potential of contacts point when collision occurs. The mechanism in handling the collision detection must be carefully designed to make sure the interactiveness of the simulation. Collision detection mechanism is essential for almost every simulation running in virtual environment [17-20]. For example, in medical simulation, the precise contact between the object and the human body must be

simulated correctly as we do not want to harm the human body. In virtual surgeon for example, the colliding area between the scalpel and the human bodies needs to use additional computational resources. This is because the simulation needs to recalculate the bounding area used to detect collision and create a new one. In computer games, the speed of intersection plays important roles in creating fast response collision detection during gameplay. Most of computer games just use simple bounding box to detect collision as it performs faster and does not require re-calculation of bounding box. Hence, collision detection has widely used in simulation of virtual environment, animation for realistic environment and robotic where the contact point between object and robot must be measured precisely [21-31].

3. VIRTUAL ENVIRONMENT WITH BOUNDING-VOLUME HIERARCHIES

Virtual environments composed of many objects that could be static or in motion, where each of objects may have thousands of primitives. Testing object interference between those primitives could become troublesome as pairwise intersection tests that need to be performed is countless (which means here, huge) [35]. Hence, spatial data structures becoming one of the solution for accelerating collision detection in massive environments [35].

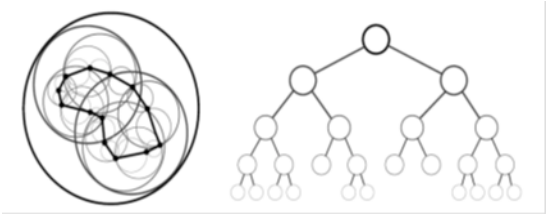

Figure 1. The left hand side image shows a BVH with Sphere BV while on the right hand side image, shows unbalanced hierarchical form using binary type hierarchy.

Spatial data structure can be used in two ways [35]. First, they can be used to reduce the pairwise test among static and moving object in virtual environment. According to [35], for **n** objects there are potentially collided pairs. Next structures searching can be undergoing to reject most of these pairs [48]. Alternatively, they can be used to reduce the pairwise test between two or more primitives or objects with primitives. Both traditions can be built in pre-processing step and are typically motionless.

4. BOUNDING-VOLUME HIERARCHIES

Bounding-Volume Hierarchies (BVH) is a hierarchical representation of 3D object in virtual environments to reduce the computational cost of in various applications such as culling system and collision detection. BVH are simply a tree structure that represents geometric models with specific bounding volumes. It works like a tree that has a root (upper division), a group of leafs (middle division) and a leaf (last division). Each node has it bounding-volumes that cover the children node. The main idea of BVH is to increase the level of the tree where it can create secondary node consists of left and right node. Each node stores a BV as a leaf node. Example of bounding-volume hierarchy is shown in Figure 1.

BVH allows the intersection occurs without searching for non-colliding pairs from the hierarchy tree. For example, given two objects with their BVH, when root of the hierarchies do not intersect, the calculation will not be performed for both objects. However, when the root of both hierarchies intersects, it checks for intersection between one root of the hierarchies' tree and the other children of objects hierarchies' tree. In this case, it recursively checks again whether there is intersection between both objects at middle level until it found the correct intersection.

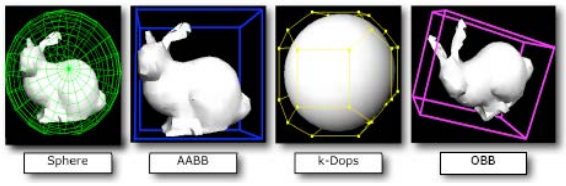

Figure 2. Examples of common BVs as described in [36].

4.1 Bounding-Volume

Bounding-Volume (BV) is an important part of BVH construction. Numerous BV have been developed in the past in order to minimize the computational cost of performing collision detection. Instead of using primitive-primitive checking between intersected 3D objects, BV helps to speed up the process by enclosing bunch of triangles into single BV before proceed with collision checking. This is to reduce the possibility of eliminating set of triangles that does not intersect.

At the present time, there are several famous BVs such as spheres [32], Axis Aligned Bounding Box (AABB) [22, 33, 34], Oriented Bounding Box (OBB) [17, 22, 35], Discrete Oriented Polytope (k-DOP) [28], Oriented Convex Polyhedra [36], and hybrid combination BV [20]. Most large scale 3D simulations used bounding box because of the simplicity, require less storage, fast response of collision, and easy to implement [37]. Figure 2 illustrates most commonly used bounding volume.

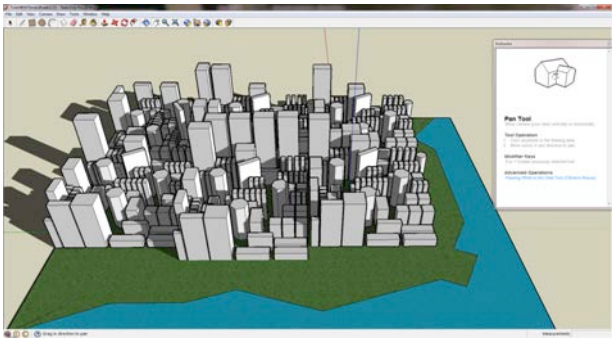

Figure 3. Urban Simulation creation.

5. VIRTUAL ENVIONMENT CREATION

The construction of urban environment begins by using Google SketchUp version 7.0 and Deep Exploration version 5.0 as tools to draw and viewing 3D models. Various type of building is developed using Google SketchUp not limiting the shape of buildings. The objective is to develop an effective enough urban environment to meet the main research objective of using hierarchical representation to detect object interference in urban simulation. Thus, the development intends to abandon few non-important elements such as textures, lighting, and culling system. The research just intends to test the capabilities of using BVH in urban environment effectively. Urban environment is not necessarily too large or too small (Figure 3). As long as the urban environment meets the objective to become supporting tools to proof the new optimization of BVH traversal algorithm.

5.1 Integration Spatial Object Median Technique for Fast Construction of BVH

The implementation of BVH in Urban simulation starts by first loading the corresponding 3DS file into OpenGL programming. Then, the process begins by parsing parameters such as total vertex, vertex points, total faces, and face points. When the urban simulation successfully loaded into virtual environment, all related parameters will be passed into BVH load function. BVH function will handle all the BVH construction and its rendering. For this construction, we used Spatial Median Technique for fast construction of BVH [38]. The process is illustrated as follows in Figure 4:-

1. Start Create BV for the objects
2. Calculate all midpoints of the objects
    a) Create Midpoints BV
    b) Create BV space for all midpoints
    c) Find Minimum and Maximum points for the midpoint space
3. Splitting Process
    a) Determine the longest axis for separating plane
    b) Split the BV (Midpoints BV) according to their spatial object median (SOMS)
4. Create Left and Right BV for the objects using midpoint separating plane.
5. Repeat Procedure until Stopping Criteria is met.

Figure 4. BVH Creation with SOMS implementation

In our experiment, total of 5672 triangles is used to build a complete urban simulation. Figure 5 shows the corresponding urban simulation with BVH.

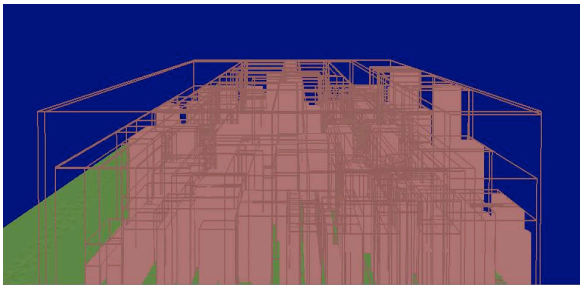

Figure 5. Complex environment of urban simulation with hierarchical representation (texture off)

Table 1. Comparison of BVH Construction

Object	Spatial Median (ms)	SOMS(ms)
Urban Environment (BVH Level 6)	5.15625	4.765625
Urban Environment (BVH level 7)	7.890625	7.4765625

6. CONCLUSION

In this paper, we have described the performance of improved splitting rule called SOMS. SOMS technique help reducing the problem of splitting parent nodes into two nodes that contains fewer triangles and in balanced mode. SOMS had an advantages of splitting each node until the lowest level with 1 triangle 1 BV and almost do not need to use splitting heuristic to determine non-splitting condition (Case where same triangle is exists and computer floating points problems that rounded the number into closes floating number). The implementation of SOMS technique is successfully been done in urban simulation. SOMS could produce fast and efficient BVH tree especially in urban simulation where there is large set of polygons that need to be divided accordingly. However, this technique still has limitation where it can only be used for large set of polygons or large scale objects that might contains millions of triangles. This research is still on going where it is going to be implemented in continuous collision detection technique.

7. ACKNOLEDGMENT

The author(s) would like to thanks to the members of Graphics and Visualization Group (GRAVS – www.gravslab.com) for the research collaboration and technical help. This paper also supported by Universiti Teknikal Malaysia Melaka PJP Grant VOT S00903.

8. REFERENCES

[1] M. A. M. Azahar, M. S. Sunar, A. Bade, and D. Daman, "Crowd Simulation for Ancient Malacca Virtual Walkthrough," in The 4th International Conference on Information & Communication Technology and Systems, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, 2008, pp. 511 - 516.

[2] N. M. Suaib, A. Bade, and D. Mohamad, "Collision Detection Using Bounding-Volume for avatars in Virtual Environment applications," in The 4th International Conference on Information & Communication Technology and Systems, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, 2008, pp. 486 - 491.

[3] G. Jianhong, H. Hanwu, Z. Wenxuan, and L. Yanfei, "Research on real-time collision detection for vehicle driving in the virtual environment," in International Conference on Information and Automation, 2008. ICIA 2008. , 2008, pp. 1834-1839.

[4] J. Hamill and O. S. Carol, "Virtual Dublin - A Framework for Real-Time Urban Simulation," WSCG, vol. 11, pp. 221-225, 2003.

[5] J. Willmott, L. I. Wright, D. B. Arnold, and A. M. Day, "Rendering of large and complex urban environments for real time heritage reconstructions," presented at the Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage, Glyfada, Greece, 2001.

[6] T. Manninen, "Interaction in networked virtual environments as communicative action: social theory and multi-player games," in Groupware, 2000. CRIWG 2000. Proceedings. Sixth International Workshop on, 2000, pp. 154-157.

[7] U. S. Team. (2010). Virtual Los Angeles. Available: http://www.ust.ucla.edu/ustweb/projects.html

[8] M. Dikaiakou, A. Efthymiou, and Y. Chrysanthou, "Modelling the Walled City of Nicosia," in 4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage, 2003.

[9] Z. Ruilin, G. Weijie, and Z. Xianghui, "Approach of Geometric Texture Mapping Based on Discrete Gradient Searching," in Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on, 2010, pp. 315-318.

[10] M. H. Yoon, D. O. Kim, R. H. Park, and S. W. Lee, "Geometry-dependent texture map compression," Electronics Letters, vol. 46, pp. 43-44, 2010.

[11] L. Tong-Yee, Y. Shao-Wei, and I. C. Yeh, "Texture Mapping with Hard Constraints Using Warping Scheme," Visualization and Computer Graphics, IEEE Transactions on, vol. 14, pp. 382-395, 2008.

[12] A. Maki, H. Watanabe, and C. Wiles, "Geotensity: combining motion and lighting for 3D surface reconstruction," in Computer Vision, 1998. Sixth International Conference on, 1998, pp. 1053-1060.

[13] R. Dosselmann and Y. Xue Dong, "Mean shift point-mass level-of-detail," in Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on, 2008, pp. 000037-000042.

[14] [14] C. Yi, W. Kim, and T. Kim, "Improvement of Collision Detection Performance of Hierarchies by Using Dynamic-Density of 3D Objects Based on LOD (Level-of-Detail)," presented at the Proceedings of the Computer Graphics, Imaging and Visualisation, 2007.

[15] H. Tan Kim and D. Daman, "A review on level of detail," in Computer Graphics, Imaging and Visualization, 2004. CGIV 2004. Proceedings. International Conference on, 2004, pp. 70-75.

[16] A. E. W. Mason and E. H. Blake, "A graphical representation of the state spaces of hierarchical level-of-detail scene descriptions," Visualization and Computer Graphics, IEEE Transactions on, vol. 7, pp. 70-75, 2001.

[17] [J.W. Chang, W. Wang, and M.-S. Kim, "Efficient collision detection using a dual OBB-sphere bounding volume hierarchy," Computer-Aided Design, vol. In Press, Corrected Proof, 2009.

[18] V. R. Kamat and J. C. Martinez, "Interactive collision detection in three-dimensional visualizations of simulated construction operations," Engineering with Computers, vol. 23, pp. 79-91, 2007.

[19] S. Trenkel, R. Weller, and G. Zachmann, "A Benchmarking Suite for Static Collision Detection Algorithms," presented at the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Plzen, Czech Republic, 2007.

[20] S. H. Kockara, T.; Iqbal, K.; Bayrak, C.; Rowe, Richard;, "Collision Detection - A Survey," presented at the IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC., 2007.

[21] F. A. Madera, A. M. Day, and S. D. Laycock, "A Hybrid Bounding Volume Algorithm to Detect Collisions between Deformable Objects," in Second International Conferences on Advances in Computer-Human Interactions, 2009. ACHI '09. , 2009, pp. 136-141.

[22] C. Tu and L. Yu, "Research on Collision Detection Algorithm Based on AABB-OBB Bounding Volume," in First International Workshop on Education Technology and Computer Science, 2009. ETCS '09. , 2009, pp. 331-333.

[23] H. A. Sulaiman, A. Bade, D. Daman, and N. M. Suaib, "Collision Detection using Bounding-Volume Hierarchies in Urban Simulation," presented at the The 5th Postgraduate Annual Research Seminar, Faculty of Computer Science & Information System, UTM, 2009.

[24] G. Zachmann, "Collision Detection as a Fundamental Technology in VR Based Product Engineering," presented at the 2nd Advanced Study Institute "Product Engineering: Tools and Methods Based on Virtual Reality, Chania, Crete, 2008.

[25] T. Min, C. Sean, Y. Sung-Eui, and M. Dinesh, "Interactive continuous collision detection between deformable models using connectivity-based culling," presented at the Proceedings of the 2008 ACM symposium on Solid and physical modeling, Stony Brook, New York, 2008.

[26] S. Redon, A. Kheddar, and S. Coquillart, "Fast Continuous Collision Detection between Rigid Bodies," Computer Graphics Forum, 2002.

[27] G. Zachmann, "Virtual Reality in Assembly Simulation - Collision Detection, Simulation Algorithms, and Interaction Techniques," Department of Computer Science, Darmstadt University of Technology, Germany, 2000.

[28] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," IEEE Transactions on Visualization and Computer Graphics, vol. 4, pp. 21-36, 1998.

[29] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-COLLIDE: an interactive and exact collision detection system for large-scale environments," presented at the Proceedings of the 1995 symposium on Interactive 3D graphics, Monterey, California, United States, 1995.

[30] A. Garcia-Alonso, Nicol, s. Serrano, and J. Flaquer, "Solving the Collision Detection Problem," IEEE Comput. Graph. Appl., vol. 14, pp. 36-43, 1994.

[31] [31] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," Robotics and Automation, IEEE Journal of, vol. 4, pp. 193-203, 1988.

[32] [32] L. Liu, Z.-q. Wang, and S.-h. Xia, "A Volumetric Bounding Volume Hierarchy for Collision Detection," in 10th IEEE International Conference on Computer-Aided Design and Computer Graphics, 2007 2007, pp. 485-488.

[33] [33] X. Zhang and Y. J. Kim, "Interactive Collision Detection for Deformable Models Using Streaming AABBs," IEEE Transactions on Visualization and Computer Graphics, vol. 13, pp. 318-329, 2007.

[34] [34] R. e. Weller, J. Klein, and G. Zachmann, "A Model for the Expected Running Time of Collision Detection using AABB Trees," in Eurographics Symposium on Virtual Environments (EGVE), Lisbon, Portugal, 2006.

[35] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: a hierarchical structure for rapid interference detection," presented at the Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996.

[36] A. Bade, N. Suaib, M. Z. A, and T. S. T. M, "Oriented convex polyhedra for collision detection in 3D computer animation," presented at the Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, Kuala Lumpur, Malaysia, 2006.

[37] M. C. Lin and D. Manocha, "Collision and Proximity Queries," in In Handbook of Discrete and Computational

Geometry, 2nd Ed. vol. 35, Boca Raton, FL: CRC Press LLC, 2004, pp. 787-807.

[38] Sulaman H.A, and Bade A. Balanced Hierarchical Method for Collision Detection in Virtual Environment. *Communications in Computer and Information Science, 2011,* Volume 181, Part 5, 493-501, DOI: 10.1007/978-3-642-22203-0_43