Jurnal
Teknologi

Full Paper

COLLISION DETECTION FOR CLOTH SIMULATION USING BOUNDING SPHERE HIERARCHY

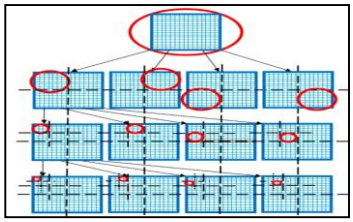
Abdullah Bade*, Ching Sue Ping, Siti Hasnah Tanalol

School of Science & Technology, University Malaysia Sabah, Kota Kinabalu Sabah, Malaysia

Article history
Received
3 December 2013
Received in revised form
2 July 2014
Accepted
25 November 2014

*Corresponding author
abb@ums.edu.my

Graphical abstract



Abstract

For the past 2-decades, the challenges of collision detection on cloth simulation have attracted numerous researchers. Simple mass spring model is used to model the cloth where the movement of the particles within the cloth was controlled by applying the Newton's second law. After the modeling stage, implementation of the collision detection algorithm took place on cloth has been done. The collision detection technique used is bounding sphere hierarchy. Then, quad tree is being used to partitioning the bounding sphere and the collision search was based on the top-down approach. A prototype of the collision detection system is developed on cloth simulation and several experiments were conducted. Time taken for this system to be executed is around 235.258 milliseconds. Then the frame rate is at the average of 22 frames per second which is close to the real time system. Times taken for the collision detection system travels from root to nodes were 23 seconds. As a conclusion, the computational cost for bounding sphere hierarchy is much higher because the bounding sphere required more vertices for generation process, however the execution time for bounding sphere hierarchy is faster than the AABB hierarchy.

Keywords: Component, collision detection, bounding sphere hierarchy, quad tree, cloth

1.0 INTRODUCTION

In this modern era, modeling and simulation using computer has been commonly used since the computational speed and the realism of the model are improving as the time passes. Deformable object are object in which their shape will change whenever there are external forces applying on the object, but the shape of the object will return to its original state when there are no other forces applied [1]. One of the deformable objects which have been widely researched is cloth model. It is important because it adds to the realism of a graphically generated environment. By using mathematical and computational techniques such as Bezier Curve, Splines, Free-form deformation and mass spring model, deformable objects can be formed efficiently [2]. Mass spring model has been widely used for modeling deformable object even though it is not as accurate as the finite element models. The computational cost of this model is not expensive, thus making it more favorable. The simple mass spring model is created using a mesh of points connecting each other with a spring in a regular-shaped structure [3].

Collision detection is one of the most frequently used techniques in computer graphics to detect the intersection between two or more objects. Collision occurs when there is at least one points of the object were in contact with the other object [4]. Most of the collision detection occurs in animation, modeling and simulation. There are many algorithms developed to detect the collision within objects, most of the algorithms which can apply on the deformable object were able to perform on the rigid body [5]. The algorithms purposed for collision detection are Sweep and Prune algorithms, Lin-Canny Bounding Volume algorithm, Gilbert-Johnson-Keerthi (GJK) Distance

algorithm and Bounding Volume Hierarchy (BVHs). Among the algorithms, BVHs has been proven to be one of the most efficient data structures for collision detection [6]. There are many types of bounding volume been studied from the past, the examples are spheres, axis-aligned bounding boxes (AABBs), object-oriented bounding boxes (OBBs) and discrete-oriented polytopes (DOPs) [5].

2.0 METHODOLOGY

A moving ball was used to simulate the collision on the cloth. Whenever the bounding sphere collides with the moving ball, the bounding sphere hierarchy will be partitioned based on quad tree concept. Quad tree is used instead of binary tree in order to set up a faster formation. Once the distance between the objects is smaller than the radius of the bounding sphere, then the bounding sphere will subdivide until the subdivision reaches the forth level which contain the smallest bounding sphere, then the subdivision will stops and the collisions between the objects will be detected. The nodes will continue the division process until each of the child nodes contains only one particle within the cloth. Then the search of the collision was done by top-down approach meaning that the search travelled from the root to the node. The colour of the bounding spheres will change to red when there are collisions detected and remained green if there are no collisions detected.

2.1 Simple Mass Spring Model

Cloth is a highly deformable object which is complex to model and require high demand on computational resources [7]. In this paper, the cloth was created using simple Mass Spring Model where the distance between the particles inside the cloth was control using the spring constraint. The structure of the cloth was formed by arranging the number of particles along the width and height of the cloth from the point (0, 0, 0) to (width, -height, 0). Then the position of each of the particles was calculated based on formula (1).

$$Pos_{Particle} = y \times number\ of\ particles\ within\ width \times x \quad (1)$$

Where x is the column of the particles and y is the row of the particles. Then structure of the cloth was done by forming triangles using 3 particles was shown in Figure 1.

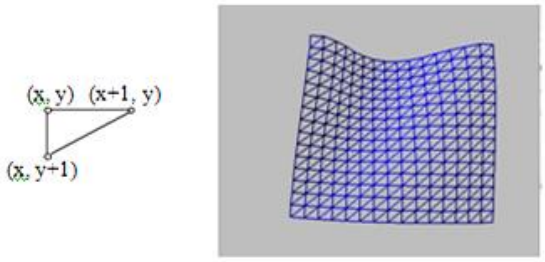


Figure 1 Particles arrangement for cloth structure

According to Newton's second law which state that acceleration is equal to forces divided by mass, therefore this law was applied on each of the particles within the cloth so that when there are any collision, the particles will act accordingly. However, applying just the Newton's second law, on the particles will not create a cloth moving naturally. Therefore, the position of the particles has to be updated in all times. The concept of updating the position is shown in Figure 2.

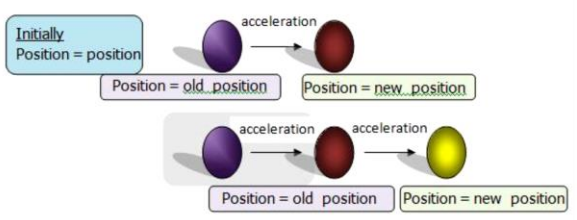


Figure 2 Concepts of updating the position of the particles

Referring to Figure 2, the position of the ball will be updated whenever there are any forces applying on the particles which cause the particles to accelerate. The position of the particles can be calculated using Equation (2).

$$Pos_{new} = Pos_{new} + (Pos_{new} - Pos_{old}) \times (1.0 - D) + acceleration \times T \quad (2)$$

Where Pos_{new} is the current position of the particles, Pos_{old} is the old position of the particles, D is the damping value of the cloth preset in the program, *acceleration* is equal to force divided by mass and is the size of time step to be taken for each frame.

2.2 Collision Detection

The quad tree is used to subdivide the particles within the cloth until there is only one particles contain in each of the child nodes of the tree. In order to detect the collisions, the radius of the bounding sphere has to be calculated and the distance between these bounding spheres and the moving ball has to be updated. Figure 3 shows the formation of the quad tree.

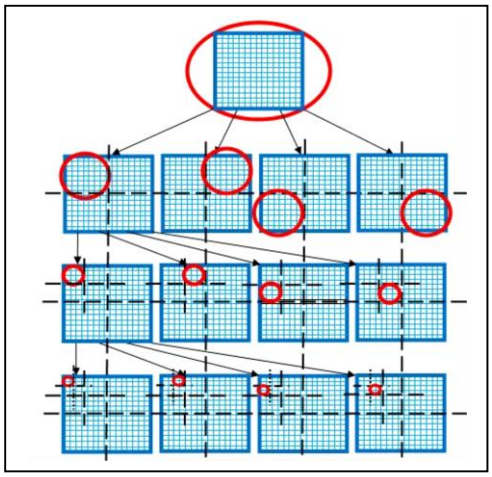


Figure 3 Concepts of updating the position of the particles

Quad tree partition was applied on the cloth in order for each of the particles was surrounded by a sphere for the collision detection. This has to be done in order to detect the collision more accurately.

The bounding sphere was created, where the radius of the sphere is differing for each level of the quad tree. Therefore, the algorithm to getting the radius is shown in Figure 4.

1. Check the longest length of the cloth. (either width or height)
2. Radius of the first sphere = (longest length + 1) ÷2.
3. Radius for the next spheres = radius of previous sphere ÷2

Figure 4 Algorithm to get the radius of the sphere

After the bounding sphere created, the checking for collision was done by calculating the distance between the bounding sphere and the moving ball. Since the ball used to collide with the cloth is a moving object, the position of the ball will be changing at all times. Therefore, the radius of the moving ball can be calculated using Equation (3).

$$R_{ball} = Pos_{ball} + ball\ radius \quad (3)$$

Where R_{ball} is the radius of ball, Pos_{ball} is the position of the ball and *ball radius* is the radius of the ball which has been preset as 1. Then the distance between the cloth and the moving ball is being calculated using Equation (4).

$$Distance = \| R_{spbere} + R_{ball} \| \quad (4)$$

Where, R_{spbere} is the radius of the bounding sphere which will be updated when the ball is moving and R_{ball} is the radius of the moving ball. The next step will be the checks for collision. The checking and updates for the bounding sphere were done. Collisions were detected when the Distance is smaller than the radius of the moving ball.

3.0 RESULTS

The construction of the collision detection for cloth was shown in Figure 5 and Figure 6. Several tests were conducted and the analysis was discussed. The tests conducted are times execution test, frame test, OpenGL functions call test, and CPUs Average Utilization test.

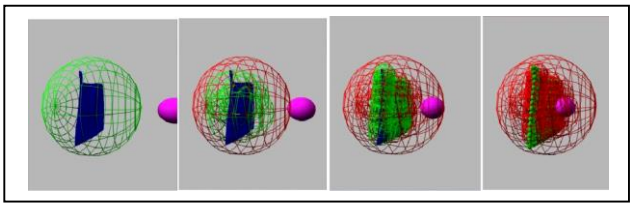


Figure 5 The bounding sphere hierarchy

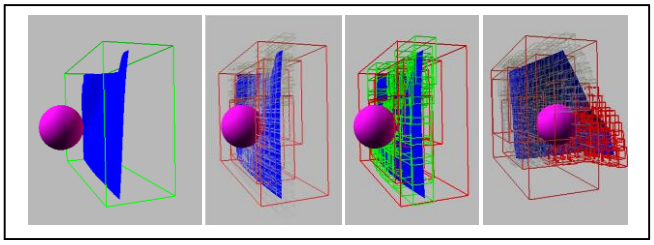


Figure 6 The Axis Aligned Bounding Box (AABB)

The times execution test was the calculation of time to execute the system. This test was conducted 20 times in order to get the average time of execution and the results were shown in Table 1.

Table 1 20 sets of time execution test for collision detection using bounding sphere hierarchy and AABB hierarchy

Experiment	Time Execution (ms) for Bounding Sphere Hierarchy	Time Execution (ms) for AABB Hierarchy
1	215.877	1233.01
2	188.131	1259.05
3	220.172	1267.02
4	216.726	1203.98
5	236.049	1263.00
6	223.576	1256.01
7	215.311	1177.10
8	233.671	1211.98
9	253.518	1260.03
10	259.710	1242.00
11	249.192	1259.02
12	245.186	1240.99
13	253.523	1259.00
14	232.061	1241.01
15	229.389	1243.03

Experiment	Time Execution (ms) for Bounding Sphere Hierarchy	Time Execution (ms) for AABB Hierarchy
16	274.904	1249.02
17	237.714	1272.03
18	226.821	1262.04
19	235.760	1271.02
20	257.875	1224.01
Average	**235.2583**	**1244.718**

From Table 1, the average time execution for AABB hierarchy (1244.72 ms) is more than of using bounding sphere hierarchy (235.26 ms) which indicates that computational cost for AABB is higher than bounding sphere. Figure 7 shows the comparison of time execution for bounding sphere hierarchy and AABB.

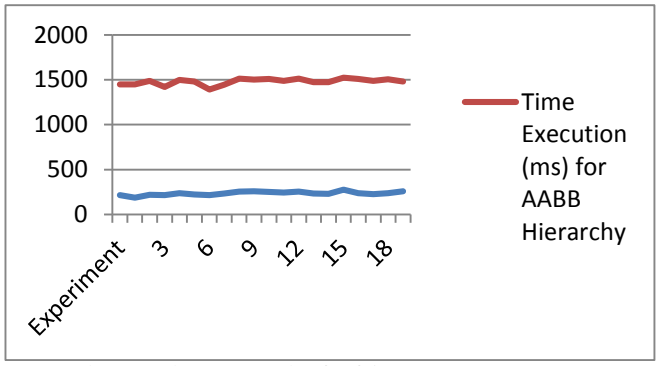


Figure 7 Time execution(ms) for BSH and AABB

The second experiment conducted was the frame test. The dramatic fall of the frame rate for bounding sphere as shown in Figure 8 is due to the huge number of OpenGL calls functions. The average frame per second for bounding sphere hierarchy during the simulation was 13.41 whilst AABB was 21.17 seconds which is roughly closer to the real time. This shows that AABB hierarchy is formed faster than the bounding sphere hierarchy. Figure 8 shows the comparison of frame test for bounding sphere is and AABB.

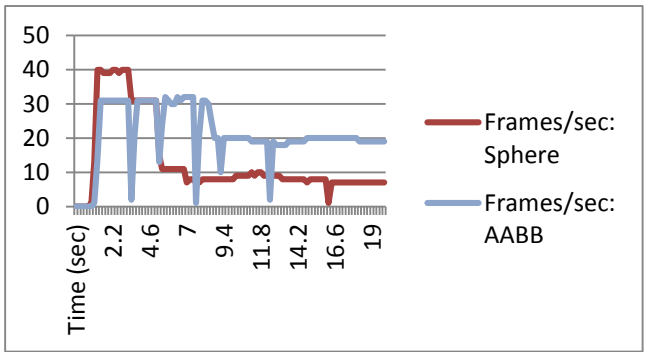


Figure 8 Comparison of frame test for BSH and AABB

The OpenGL (OGL) functions call test indicates the number of functions called when executing the program for both bounding sphere hierarchy and AABB hierarchy.

Referring to Figure 9, the graph of OGL calls per frame for bounding sphere hierarchy shows that the highest number of OGL calls can reach up to 27,000 calls which is a huge number of OGL calls. This will highly affect the frame test which proves that the frame test was affected by the OGL call as indicated in Figure 9.

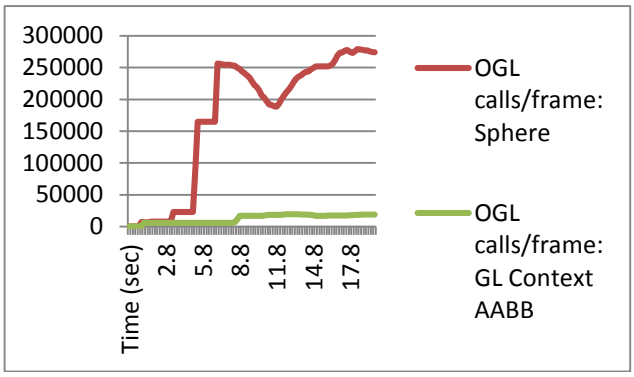


Figure 9 Comparison of OpenGL function cal per frame for bounding sphere hierarchy and AABB hierarchy

CPUs Average Utilization test is to test on the percentage of the memory usage for CPUs when both bounding sphere hierarchy and AABB hierarchy programs are being executed. Figure 10 shows the percentage of CPUs average utilization for bounding sphere hierarchy is more than the AABB hierarchy for about 15%. Since the OGL calls function for bounding sphere hierarchy is higher than AABB, the CPUs utilization for bounding sphere hierarchy will also be higher than AABB. Hence, the CPUs average utilization is directly proportional to OGL calls.

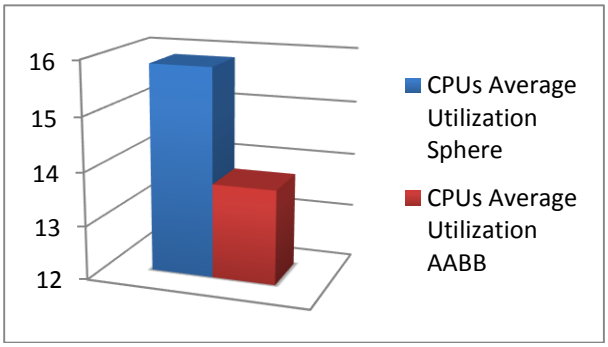


Figure 10 Graph of CPUs Average Utilization test for bounding sphere hierarchy and AABB hierarchy

4.0 CONCLUSION

A series of tests has been conducted on the prototype for cloth simulation using bounding sphere hierarchy.

Meanwhile, the cloth simulation using AABB hierarchy acted as the control experiments for a comparison with the bounding sphere hierarchy.

After running the tests, the processes of building and updating the bounding sphere hierarchy is affected by the number of generated vertices. This will cause a huge number of OGL functions call which will lower down the frame per second generated and CPUs utilization. Mean while, the number of OGL functions calls also had a little bit of effect on the time generation in which the bounding sphere require more time in detecting the collisions. However, the execution time test showed that bounding sphere hierarchy required lesser time to execute comparing with AABB hierarchy.

Besides using bounding sphere hierarchy to detect collision on cloth, other bounding volume can be used for more efficient and accurate collision detection, for example the oriented bounding boxes and k-DOP. Other than collision detection on the surface, experiments for the volumetric collision detection can be done too.

References

[1] Sulaiman, H. A. and A. Bade. 2011. The Construction of Balanced Bounding-Volume Hierarchies using Spatial Object Median Splitting Method for Collision Detection. *International Journal of New Computer Architecture and Their Application (IJNCAA)*. 2: 396-403.

[2] Gibson, S. F. and B. Mirtich. 1997. *A Survey of Deformable Modeling in Computer Graphic*. MERL.

[3] Rajiv, P. 2011. Cloth *Simulation using Mass-Spring Technique*. NCCA CGIT.

[4] Jiménez, P., F. Thomas and C. Torras. 2001. 3D Collision Detection: A Survey. *Computers and Graphics*. 25: 269-285.

[5] Teschner, M. E., S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser and P. Volino. 2004. Collision Detection for Deformable Objects. In Proceedings of Eurographics 2004, State-of-the-Art Report. 119-135.

[6] Andersen, K. A. and C. Bay. 2006. A Survey of Algorithms for Construction of Optimal Heterogeneous Bounding Volume Hierarchies. [Online]. From: http://image.diku.dk/projects/media/christian.bay.kasper.andersen.06B.pdf. [Accessed on 27 June 2013].

[7] Simnett, T. 2012. *Real-Time Simulation and Visualisation of Cloth using Edge-based Adaptive Meshes*. University of East Anglia.