STUDIA INFORMATICA 2010
Volume 31 Number 1 (88)

Ewa LACH, Jacek LACH
Politechnika Śląska, Instytut Informatyki

BOUNDING VOLUME HIERARCHIES FOR COLLISION DETECTION

Summary. The paper describes a problem of generation of bounding volume hierarchies for collision detection in 3D scenes. The effectiveness of two methods is tested followed by discussion on necessary features of an animation framework. The fACT framework is introduced.

Keywords: bounding volume hierarchies, collision detection, 3D animation framework

HIERARCHIE BRYŁ OGRANICZAJĄCYCH W WYKRYWANIU KOLIZJI

Streszczenie. W artykule opisano problem generacji hierarchii brył ograniczających zastosowanych do wykrywania kolizji w scenach trójwymiarowych. Przeprowadzono testy efektywności dwóch metod podsumowane określeniem niezbędnych cech środowiska animacyjnego. Przedstawiono środowisko do animowania postaci 3D: fACT.

Słowa kluczowe: hierarchie brył ograniczających, wykrywanie kolizji, środowisko 3D do animowania postaci

1. Introduction

Collision detection (CD) is a fundamental problem in computer graphics and visualization. It frequently arises in applications such as video games, animation, computer aided design and virtual reality. Very often in addition to detecting whether two objects overlap or reporting the penetration features, CD algorithm is required to do so at real-time rates. The input to a collision detection algorithm is a large number of geometric objects comprising

an environment, together with a set of objects moving within the environment. The computational cost of a collision detection algorithm depends not only on the complexity of the basic interference test used, but also on the number of times this test is applied. Therefore for complex scenes comparing all pairs of primitive geometric elements is inefficient. To address this problem many of the approaches have used hierarchies of bounding volumes (BVHs) or spatial decompositions. These techniques reduce the number of pairs of objects that need to be checked for contact by the approximation of objects with bounding volumes (Fig. 1) or decomposition of the space occupied by objects.

The computation time of bounding volume hierarchy approach can be formulated as follows [2]:

$$T = N \times B + P \times C \tag{1}$$

where T means total execution time, N: number of BV pair overlap tests, B: time for testing a pair of BV, P: number of primitive pair overlap tests, C: time for testing a pair of primitives. This formula indicates that effectiveness of CD algorithm depends on the tightness of the bounding volumes and the simplicity of the overlap test for a pair of BV. The first factor is related to N and P, whereas the second factor is related to B.

The simplest overlap tests provide spheres and axially aligned bounding boxes (AABBs). However, oriented bounding boxes (OBBs) and discrete orientation polytopes (k-DOPs) fit volumes more tightly.

In most 3D graphic applications collision detection methods are limited, adding new CD algorithms is very difficult or impossible and creation of BVHs or space decomposition is automatic. Authors of the paper argue that for different virtual scenes different collision detection methods would yield better solutions. Therefore graphic software should offer more CD methods. For example in the large crowd scene testing collision on mesh level seems inefficient. Different situation occurs when scene of two virtual peoples shaking hands is generated. Additionally, authors believe that in some situation manual creation of bounding volume hierarchies is more cost-effective then automatic one and designers of graphic applications should not dismiss this method as too complex and time-consuming.

In the rest of this paper, first relevant related work is reviewed in Section 2. Then in Section 3 the problem of bounding volume hierarchies is summarized, followed by description of tested collision methods in Section 4. Section 5 introduces animation framework fACT, which allows to add various collision detection techniques, as well as predefined BVHs. The paper is concluded in Section 6 with analysis of conducted experiments and their results.

2. Related work

Interference and collision detection problems have been extensively studied in the literature. Rigid bodies in static poses are tested by the basic type of collision detection algorithms [2, 10, 13, 14, 16]. More complex methods are used in dynamic collision detection [1, 6, 7, 15] or to detect collisions between deformable models [3-5]. Although, these complicated problems seems more challenging then the basic type problem, because of their efficiency CD algorithms solving the latter one are the most popular in animation applications. Additionally, the basic type algorithms serve as the basis for the development of efficient algorithms for more complicated problems. For these reasons, in the paper, we focus on the efficient algorithms for the static cases.

Bounding-volume hierarchies have proven to be among the most efficient data structures for collision detection [1, 8, 12] and a variety of bounding volumes have been investigated. Algorithms using hierarchies of spheres [3, 5, 9, 14] and axis-aligned bounding boxes AABB [10, 11] do very well performing 'rejection tests', whenever two objects are far apart. However, when the two objects are in close proximity and can have multiple contacts, their performance slows down considerably and they become a major bottleneck in the animation generation. Another bounding volume that has become popular is the oriented bounding box OBB [2,13,14], which surrounds an object with a bounding box whose orientation is arbitrary with respect to the coordinate axes (Fig. 1(b)). OBB approximates an object tighter then previous two volumes, thanks to the fact that its orientation can be changed to make the volume as small as possible. Regrettably its overlap tests are more complex then the ones for spheres and AABBs. Seeking a compromise between the relatively poor tightness of bounding spheres and AABBs, and the relatively high costs of overlap tests of OBBs, k-DOPs [16,17] were proposed. Discrete orientation polytope (k-dop) is a convex polytope whose facets are determined by half-spaces whose outward normals come from a small fixed set of k orientations (Fig. 1(d)).

Different types of collision detection algorithms are based on the spatial decomposition, which divides the space occupied by the objects. To detect collision only those pairs of objects (or parts of objects) are checked, that are in the same or nearby cells of the decomposition. By creating hierarchy of decompositions it is possible to further accelerate CD process. Several techniques are discussed in the literature. Among them octrees [19,20], BSP-trees [18,21,22], uniform grids [23] and spatial hashing [24] are the most popular.

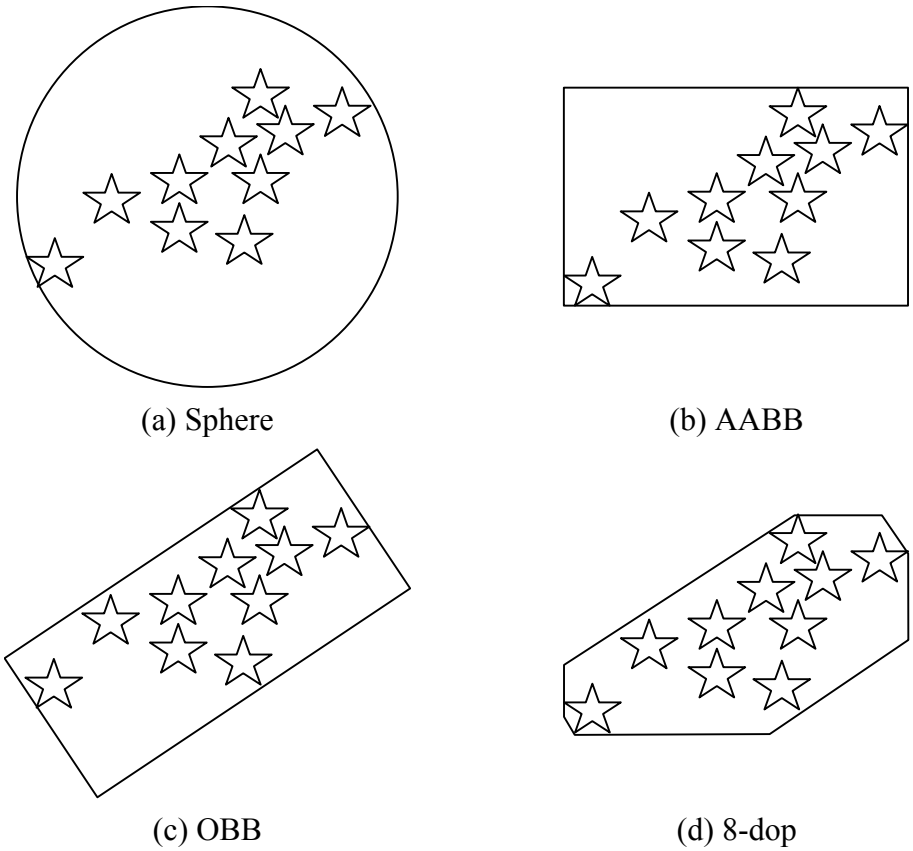


Fig. 1. Approximations of objects by four bounding volumes: sphere, axis-aligned bounding boxes (AABB), oriented bounding box (OBB) and k-dop (where k=8)

Rys. 1. Cztery, przybliżające kształt obiektów, bryły ograniczające: sfera, AABB, OBB i k-dop (gdzie k=8)

3. Bounding volume hierarchies

The goal of any hierarchical BV algorithm used for collision detection is to discard quickly pairs of the primitives which cannot intersect. Because any hierarchical algorithm is as good as the hierarchical data structure it uses, creating BV trees was considerably examined [2-5,13,14, 16,17]. "Good" BV trees for CD are defined as ones with a low number of primitive intersection tests. Several desired features of BV hierarchy were proposed [26]:

- The subtree's nodes should be relatively near each other. The lower down the tree the nearer the nodes ought to be.
- The nodes should be of minimal volume.
- Minimal number of all BVs should be used.
- Nodes near the root should remove maximal number of objects from further tests.
- Sibling nodes should overlap by minimal volume.

- The hierarchy should be balanced with respect to both: its node structure and its content. Balancing allows removal from further tests as much of the scene's objects as possible whenever a branch is not traversed into.

The number of hierarchies that can be built for any scene is exponential in the number of scene objects [25]. There are various design choices in constructing bounding volume trees. The degree δ of the tree (binary, ternary, etc.), top-down, bottom-up or insertion (objects are inserted into the tree one at a time) construction, the choice of the bounding volume and splitting rules are among the most important ones.

In constructing effective BVHs, our goal is to assign subsets of objects to each child of a node in such a way as to minimize some chosen value:

- the sum of the volumes (or surface areas) of the child volumes,
- the maximum volume (surface area) of the child volumes,
- the volume (surface area) of the intersection of the child volumes,
- the separation of child volumes.

Another strategy for splitting a child volume is dividing it in equal-size parts with respect to their projection along the selected axis, resulting in the most balanced hierarchy possible. Sometimes combinations of the previous strategies are used.

When generating BVH it is important to remember that no hierarchical representation gives the best performance in all cases. Furthermore, given two models, the total cost of interference detection varies considerably with the proximity and relative orientation of the models.

Collision detection algorithm depends not only on how the hierarchies are built but also on how the hierarchies are descended, when their top-level volumes overlap. Given two hierarchies: A and B there are several possible traversal alternatives:

- Descend A before B is descended or descend B before A is descended. This strategy can end up testing many leaves of B (A) against the whole of A (B).
- Descend first the volume dynamically determined larger.
- Descend A and B simultaneously.
- Descend A and B alternatingly.
- Descend based on overlap. Parts in which the hierarchies overlap more are first.

4. Collision detection methods

The paper describes tests of two CD methods that check if collision occurs between static objects of 3D scene and virtual characters. The algorithms only detect whether two objects overlap. They find first intersecting pair of objects and they do not report all penetration

features. Both collision detection methods apply orientated bounding boxes hierarchies. Algorithms traverse the scene tree (S-T) and s virtual character tree (VCH-T), descending along scene hierarchy first. The algorithms stop their execution when the level in scene tree is found, at which all nodes are separated from VCH-T or when a collision is detected between S-T and VCH-T leaves. In the first method ABVHCD bounding volume hierarchy of scene objects is generated automatically (ABVH), in the second TBVHCD we propose BV hierarchy (TBVH) generated according to a template, which was defined beforehand.

In the ABVHCD method, the degree of the tree δ can be assigned value: 2, 4 or 8. During BVH generation parent volume is split in δ equal-size parts with respect to their projection along the selected axis. Axis used for splitting are defined in configuration file of the CD method. For $\delta=2$ the axis along which the OBB is longest is chosen. As it is not possible to guarantee selecting a splitting plane that does not cut across any primitives, any straddling primitives must be dealt with when partitioning the set. In ABVHCD the position of primitive center with respect to the splitting plane or planes determines which subset it goes in. If the number of primitive objects is equal or less then δ leaves nodes for primitives are constructed. ABVHCD method ensure that minimum two children nodes define new BVs. If all primitives are assigned to one child, the biggest object is moved to another node-leaf. Finally, created BVs are adapted to fit primitives more tightly. Because virtual characters are placed within the scene, testing them against whole scene seems unproductive. Therefore, to speed up the test, ABVHCD algorithm starts checking for collisions from the second level of the scene tree.

The second CD method utilizes BVH defined in advance. In our cases BVH templates are based on animators knowledge of the scene. There are no restrictions on a degree of the scene tree. It can vary with the S-T levels. Similarly as in ABVHCD method, to prevent testing VCH-T against whole scene, the root of S-T can be omitted. To decrease the number of the pair tested for collision, we propose that some primitives can be defined in template as “use only”. This kind of primitive objects are applied to build parent bounding volumes, but they are not used individually to check for collision. Primitives of this type should be used with care, so they do not reduce precision of the CD method. The obvious fault of TBVH is its need for animator’s time and knowledge. However, it is important to remember that for static objects like walls or trees, scene hierarchy is created only once and used during every change of character position. When time of character’s strategy execution is critical the time spent on creating the TBVH seems cost-effective.

5. Collision detection with behavioral animation framework *fACT*

In this section the framework *fACT*, which allows for generation of virtual characters' animation, is surveyed. The significant advantage of using this application is an easy way in which an animator can add, change or remove different collision detection algorithms. The main goal of the framework is to provide an environment for testing different methods for behavioral strategies generation, which are understood, as sequences of operations run successively or in parallel, according to character's states or states of his environment to obtain character's goals. The *fACT* framework integrates various modules that can be added, replaced or specialized independently (Fig. 2). At the core of the system is the Virtual Character (VCH) module, composed of several components, which is responsible for managing characters. Information about the virtual environment is controlled by the Virtual Environment (VENV) module, which stores information about size, position and orientation of three-dimensional objects and characters situated in the environment. It also performs on these objects geometric transformations. Strategy Generation (SG) module is responsible for the generation of character's behaviors, Groups Manager (GM) module manages groups of virtual characters and characters actions are synchronized by Synchronization (SYN) module. Initialization (INIT) module is used to create and initialize other *fACT* modules and components. It is also responsible for parsing XML configuration files. XML format was chosen, because it is flexible, standardized, readable by humans and easily editable. Detailed information on all fACT modules can be found in [27]. In this paper we focus on modules responsible for CD.

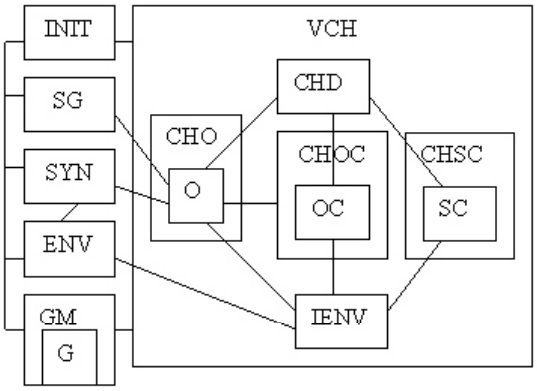
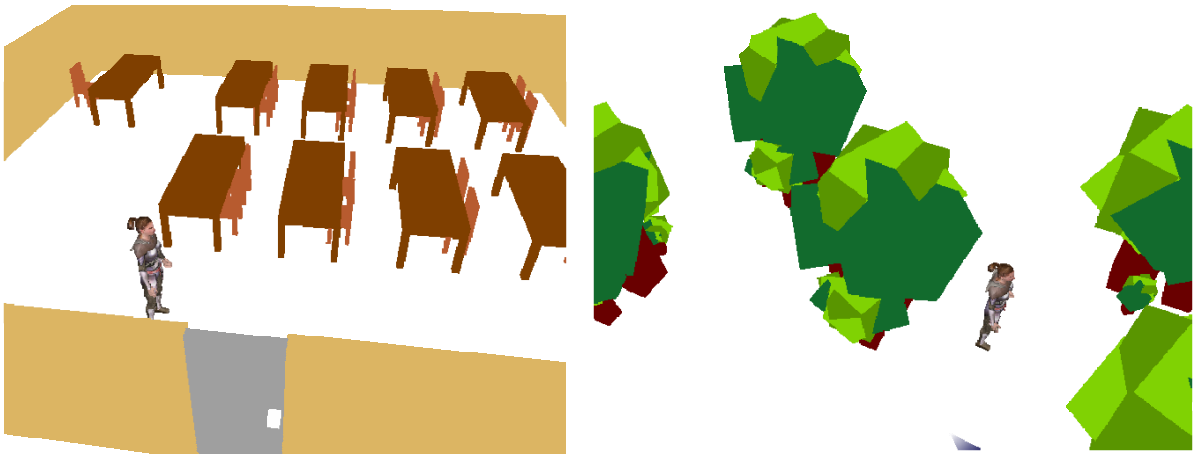


Fig. 2. Architecture of the *fACT* framework
Rys. 2. Schemat budowy środowiska *fACT*

Collision Detection methods are implemented as Operations Controllers (OC), which are the components of VCH module. The operations controllers monitor, restrict and modify operations during their execution. Operations are responsible for modifying the character or the environment, performing transformations on three-dimensional objects and determining the state of the character, the environment, objects and other characters placed within

the environment. Operations are used to build behavioral strategies. Examples of operations are: 'move character forward by 5 units', 'rotate character's right arm by 15^{o} ' or 'test if character is far from the door'. The OC can use information from the environment (from VENV module using environment interface IENV) and the Character Data (CHD) component, which stores various data on virtual character, that can be dynamically added, removed, modified, reset or accessed by different modules. Operations controllers can modify an operation, stop its execution or stop the execution of the whole strategy. OCs are added dynamically during initialization of the application according to the *fACT* configuration file. Additionally each controller can use its own configuration file.



(a) *Class* scene (b) *Field* scene

Fig. 3. 3D scenes for collision detection (class and field)
Rys. 3. Trójwymiarowe sceny zastosowane do wykrywania kolizji (klasa i pole)

6. Experiments, results and analysis

The experiments were performed for two different 3D scenes. In the first *Class* scene, a virtual human moves in a school class, with desks and chairs, restricted by four walls (Fig. 3a). In the second *Field* scene a virtual character wonders about a field with randomly placed clusters of trees (Fig. 3b). The first scene is more ordered. Therefore it is easer to find patterns that can be used to create scene template. It was uncertain if it was even possible to find hierarchy for the second, more random scene, that can be significantly better than the one generated automatically by splitting the scene. The virtual human model (avatar), used in the experiments, borrows its biomechanical characteristics from robotics. The avatar has a set of joints whose movements can be either prismatic or revolute and are ruled by the forward kinematics. Bounding volume hierarchy of the virtual character is constructed according to the template defined in a configuration file and is the same for all algorithms.

The hierarchies were tested for three different virtual character's trajectories with different amount of collisions.

Table 1

Results of detecting collisions in two scenes for three different character trajectories

	Class scene			Fild scene		
	T1	**T2**	**T3**	**T1**	**T2**	**T3**
Number of collision algorithm runs	151	301	449	582	286	48
Detected collisions	133	151	133	145	18	0
Amount of detected collisions in collision runs (%)	88.1	50.2	29.6	24.9	6.29	0

Table 1 shows how often collision algorithms were run for various walks of the virtual character and in what proportions collisions were detected. For example in scene *Class* for trajectory T1 in 88.1% of collision's checks the overlaps between primitive objects were detected.

The results of the experiments are presented in table 2. Algorithms are compared by number of times collision tests are applied to pairs of bounding volumes. The algorithm ABVHCD was tested for δ=2, δ=4 and δ=8. In the literature we found statements suggesting supremacy of binary (δ=2) trees over δ-ary trees [16], but performed experiments do not confirm it. In *Class* scene 4-ary trees behave significantly better for all trajectories. The next good ABVHCD algorithm needs to perform minimally 30% more collision tests, maximally – 74,2%. However, binary trees are better in more chaotic *Field* scene. Although, the difference between the best ABVHCD algorithm and the next best one is not as great as in previous case (min.:1,7%, max:14,4 %).

Table 2

Number of BV collision tests for investigated algorithms and their results in proportion (%) to TBVHCD

	Class scene			Field scene		
	T1	**T2**	**T3**	**T1**	**T2**	**T3**
ABVHCD- δ2	3760 (186)	5093 (180.1)	7742 (217.4)	4796 (115.1)	6473 (109.7)	326 (98.2)
ABVHCD- δ4	2158 (107)	3878 (137.1)	5668 (159.2)	5443 (130.7)	6583 (111.5)	373 (112)
ABVHCD- δ8	4875 (241)	5011 (177.2)	8698 (244.3)	8113 (194.7)	7477 (126.7)	574 (173)
TBVHCD	2022 (100)	2828 (100)	3561 (100)	4166 (100)	5903 (100)	332 (100)

The TBVHCD algorithm worked better then ABVHCD for all trajectories in school class and for most trajectories in the *Field* scene. In the one case it behaved worse, it needed to perform only 1,82% more tests than the best ABVHCD algorithm. Manually created BVHs are better for ordered scenes, but they can be also used in more random ones.

The experiments show interesting dependency. In the *Class* scene the more collisions occur the less efficient TBVHCD is in relation to the best ABVHCD algorithm. In the *Field* scene on the contrary. The cause of this inconsistency can be the way TBVHs are constructed. In a school class, there are more patterns on global levels. On the other hand, in the field it is difficult to find the best split planes for randomly placed cluster of trees. It is easer, however, to notice them within the clusters.

The conducted experiments demonstrate the need for various BVHs in different scenes with different proximity and relative orientation between models checked for collisions. They confirm that no hierarchy gives the best performance all the times. Therefore, it was reinforced that animation frameworks should provide animators with various collision algorithms, as well as allow appliance of new ones.

The analysis of the results prove also that for the most scenes it is possible to design BVH that will perform better than automatic ones. If the time of animation generation is crucial, for same scenes it can be advised to use manually created BVHs.

Summarizing, an animation software that is capable of adding, editing and applying various collision algorithms by simply means is needed. It should also provide methods that allow to use manually created BVHs. The fACT framework, applied to execute described experiments, offers all these features.

BIBLIOGRAPHY

1. Eckstein J., Schömer E.: Dynamic Collision Detection in Virtual Reality Applications. University of West Bohemia, 1999, s. 71÷78.
2. Gottschalk S., Lin M., Manocha D.: OBB-tree: A hierarchical structure for rapid interference detection. In: Proc. SIGGRAPH, 1996, s. 171÷180.
3. Spillmann J., Becker M., Teschner M.: Efficient updates of bounding sphere hierarchies for geometrically deformable models. Journal of Visual Communication and Image Representation, Orlando, FL, USA 2007, Vol. 18, No. 2.
4. Larsson T. Akenine-Möller T.: A dynamic bounding volume hierarchy for generalized collision detection. Computers & Graphics, 2006, Vol. 30, No. 3, s. 451÷460.
5. Garcia M., Bayona S., Toharia P., Mendoza C.: Comparing Sphere-Tree Generators and Hierarchy Updates for Deformable Objects Collision Detection. Advances in Visual Computing, 2005, Vol. 3804, s. 167÷174.

6. Govindaraju N., Kabul I., Lin M., Manocha D.: Fast continuous collision detection among deformable models using graphics processors. Computers & Graphics, 2007, Vol. 31, No. 1, s. 5÷14.
7. Redon S., Kheddar A., Coquillart S.: Fast continuous collision detection between rigid bodies. Computer & Graphics, 2002; Vol. 21, No. 3, s. 279÷287.
8. Teschner M., Kimmerle S., Heidelberger B., Zachmann G., Raghupathi L., Fuhrmann A., Cani M.-P., Faure F., Magnenat-Thalmann N., Strasser W., Volino P.: Collision Detection for Deformable Objects. Computer Graphics Forum, 2005, Vol. 24, No. 1, s. 61÷81.
9. Larsson T.: Fast and Tight Fitting Bounding Spheres. Proceedings of the Annual SIGRAD Conference, Stockholm, 2008, s. 27÷30.
10. Ernst M., Schneider M., Greiner G.: Collision Detection with Early Split Clipping. IEEE VGTC Pacific Visualization Symposium, Kyoto 2008.
11. Li C., Feng Y., Owen D.: SMB: Collision detection based on temporal coherence. Computer methods in applied mechanics and engineering, 2006, Vol. 195, No. 19-22, s. 2252÷2269.
12. Tropp O., Tal A., Shimshoni I., Dobkin D.: Temporal Coherence in Bounding Volume Hierarchies for Collision Detection. International Journal of Shape Modeling, 2006, Vol. 12, No. 2, s. 159÷178
13. Liu L., Wang Z.; Xia S.: A Volumetric Bounding Volume Hierarchy for Collision Detection. Proceedings of 2007 10th IEEE International Conference on Computer Aided Design and Computer Graphics, Beijing, 2007, s. 485÷488.
14. Chang J.-W., Wangb W., Kim M.-S.: Efficient collision detection using a dual OBB-sphere bounding volume hierarchy. Computer-Aided Design, 2009.
15. Choi Y.-K., Chang J.-W., Wang W., Kim M.-S., Elber G.: Continuous Collision Detection for Ellipsoids. IEEE Transactions on Visualization and Computer Graphics, 2009, Vol. 15, No. 2, s. 311÷325.
16. Klosowski J., Held M., Mitchell J., Sowizral H., Zikan K.: Efficient collision detection using bounding volume hierarchies of k-dops. IEEE Transactions on Visualization and Computer Graphics, 1998; Vol. 4, No. 1, s. 21÷37.
17. Zachmann G. Rapid collision detection by dynamically aligned dop-trees. In: Proc. of IEEE virtual reality annual international symposium, Atlanta, 1998, s. 90÷97.
18. Luque R., Comba J., Freitas C.: Broad-phase collision detection using semi-adjusting BSP-trees. Proceedings of the 2005 symposium on Interactive 3D graphics and games, Washington, 2005, s. 179÷186.
19. Thatcher U.: Loose octrees. In Game Programming Gems, Charles River Media, 2005, s. 444÷453.

20. Ganovelli F., Dingliana J., O'Sullivan C.: BucketTree: improving collision detection between deformable objects. In: Spring Conference in Computer Graphics (SCCG2000); 2000, s. 156÷63.
21. Csaba T.: Binary Space Partitions: Recent Developments. Combinatorial and Computational Geometry, 2005, Vol. 52.
22. Sigal A., Gil M., Ayellet T.: Deferred, Self-Organizing BSP Trees. Comput. Graph. Forum, 2002, Vol. 21, No. 3.
23. Turk G.: Interactive collision detection for molecular graphics. Master's thesis, Computer Science Department, University of North Carolina at Chapel Hill; 1989.
24. Teschner M., Heidelberger B., Mueller M., Pomeranets D., Gross M.: Optimized spatial hashing for collision detection of deformable objects. In: Proceedings of Vision, Modeling and Visualization; 2003, s. 47÷54.
25. Ng K., Trifonov B.: Automatic Bounding Volume Hierarchy Generation Using Stochastic Search Methods. Stochastic Search Algorithms, Vancouver, Canada, 2003.
26. Ericson Ch.: Real-Time Collision Detection. Elsevier Inc., 2005.
27. Lach E.: fACT - Animation Framework for Generation of Virtual Characters Behaviours, 1st International Conference on Information Technology, Gdańsk, 2008, s. 325÷328.

Recenzent: Dr hab. inż. Bogdan Smołka, prof. Pol. Śląskiej

Wpłynęło do Redakcji 20 listopada 2009 r.

Omówienie

W artykule opisano problem generacji hierarchii brył ograniczających (BVH), zastosowanych do wykrywania kolizji w scenach trójwymiarowych. Ponieważ obliczeniowy koszt algorytmu wykrywania kolizji zależy nie tylko od złożoności podstawowego testu na przecinanie się brył, ale także od liczby wywołań pojedynczego testu, zastosowanie BVH pozwala na istotne ograniczenie liczby testowanych par, a co za tym idzie przyspieszenie działania algorytmu. W pracy poruszono problem tworzenia BVH oraz eksperymentalnie sprawdzono efektywność dwóch metod. W ramach pierwszej metody automatycznie generowano dwu-, cztero- i ośmiostopniowe drzewa, w ramach drugiej hierarchie były tworzone na podstawie wzorca przygotowanego przy wykorzystaniu wiedzy animatora. Algorytmy testowane były dla scen różniących się stopniem uporządkowania oraz trajektorii ruchu animowanej postaci o zróżnicowanej kolizyjności. Analiza otrzymanych wyników

potwierdziła, że żadna hierarchia nie dostarcza optymalnych wyników dla każdej sytuacji. Algorytmy zachowywały się różnie w zależności od rodzaju sceny oraz bliskości i względnej orientacji sprawdzanych modeli. Z obserwacją tą wiążą się wymagania stawiane środowiskom animacyjnym, które powinny zapewnić możliwość stosowania różnych algorytmów wykrywania kolizji, a także ich edytowanie oraz dodawanie nowych.

Dodatkowo, wykonane eksperymenty wykazały, że prawie dla każdej sceny można skonstruować BVH, która będzie zachowywała się lepiej niż hierarchia wygenerowana automatycznie. W sytuacjach, w których czas działania jest podstawowym wyznacznikiem jakości algorytmu wykrywania kolizji, oprogramowanie animacyjne powinno umożliwiać wprowadzanie BVH tworzonych ręcznie.

Artykuł przedstawia środowisko do animowania postaci 3D: fACT, które spełnia wszystkie z wymagań zdefiniowanych w ramach pracy.

Addresses

Ewa Lach: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, ewa.lach@polsl.pl.
Jacek Lach: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, jacek.lach@polsl.pl.