Deformable Model Collision Detection using A-Buffer

Hanyoung Jang[*] JungHyun Han[†]

Game Research Center, College of Information and Communications
Korea University, Seoul, Korea

Abstract

This paper presents a new image-space algorithm for real-time collision detection, where the GPU computes the potentially colliding sets (PCSs), and the CPU performs the standard triangle/triangle intersection test. When the bounding boxes of two objects intersect, the intersection is passed to the GPU. By rendering the objects in the intersection region, the GPU saves the object surfaces in A-buffer about 8 times faster than the ordinary approaches. As surfaces are rendered, their depth values are stored in textures. If the depth disparity between two surfaces is small enough, a PCS is obtained, which consists of two triangle, with a triangle from an object. The PCSs are read back to the CPU, and the CPU computes the intersection points between triangles. The proposed algorithm overcomes a variety of weaknesses which are revealed in the existing collision detection techniques: it does not require any preprocessing, it can handle dynamic models including deformable and fracturing meshes, it can compute self-collisions, etc. Its performance proves its usefulness in real-time applications such as 3D games.

Introduction The mainstream of collision detection algorithms is based on object-space computation, which usually uses the bounding volume hierarchies (BVHs). On the other hand, the image-space collision detection algorithms do not require additional data structures such as BVHs. They are accelerated by GPU's rasterization capability and programable shaders. Therefore, the performance of the image-space collision detection algorithms grows upon the improvement of the GPU. The algorithms can also effectively handle deformable objects and dynamic environments. However, the image-space algorithms often suffer from the overhead of rendering and readback, and reveal the inaccuracy problem of the collision detection results. Our method improves such weaknesses of the image-space algorithms.

Features The major strengths of the proposed algorithm can be listed as follows: no special data structure is required, no preprocessing is required, dynamic objects including deforming and even fracturing objects can be handled, self-collision is detected, rendering and readback overheads are significantly reduced, and the CPU overhead is also reduced due to the small size of a PCS.

Framework This paper proposes to compute PCSs using GPU and leave the primitive-level intersection test to CPU. Each object is associated with an axis-aligned bounding box (AABB). First of all, the AABB overlap test is performed between two objects. If the AABBs overlap, the intersection is passed to the GPU as a *region of interest* (*ROI*). Given an ROI, the GPU performs three steps: surface accumulation, PCS generation, and stream reduction. A PCS consists of two triangles.

Surface accumulation In the graphics rendering fields, the effort for saving the multi-layer depth image has been proposed [Myers and Bavoil 2007]. The approach captures up to 8 layers in a pass. Our approach adopts A-buffer in order to save the surface information for collision detection. The GPU renders the object's surface of the ROI. When the objects are rendered, two values are saved into two color channels, red and green. The red channel stores the *depth* value of the fragment, and the green channel stores the *triangle ID* of the owner triangle of the fragment. Thanks to DirectX 10, triangle IDs are allocated by using *SV_PrimitiveID* semantic in shader with no additional effort.

PCS generation As the result of surface accumulation, a texel of A-buffer may contain surface information of two objects. In order to compute PCSs, a simple test is invoked. If the distance between a surface from one object and a surface from the other object is less than the threshold ϵ, the triangle IDs are retrieved from the surfaces, and the IDs of the two triangles are saved into a render target as a PCS.

Stream reduction The texels of the render target which contain PCSs should be read back to the CPU. Some texels contain the information of PCSs, but most of them have no information. This sparse data is processed to reduce the readback overhead by using *stream reduction*. Fortunately, recent GPUs such as NVidia's G80 support geometry shader, which can trivially be used to remove unwanted elements from a stream of input, and the reduced elements are written to a memory resource which can be copied to a staging resource for readback to the CPU. Finally, given the texels, the CPU computes the intersection points by performing triangle-level intersection test.

Conclusion Utilizing the functionalities of the recent GPU, the proposed algorithm can handle deformable and fracturing objects, and can perform self-collision detection. The experimental results show the feasibility of the shader-based collision detection and its performance gain, which must be useful in real-time applications such as 3D games.

Acknowledgement

This research was supported by MIC, Korea under ITRC IITA-2007-(C1090-0701-0046).

References

MYERS, K., AND BAVOIL, L. 2007. Stencil routed a-buffer. In *ACM SIGGRAPH 2007 sketches*, ACM, NY, USA, 21.

[*]e-mail: jhymail@gmail.com
[†]e-mail: jhan@korea.ac.kr