가시화 기반 N-body GPU 충돌 체크 방법

성만규*

계명대학교

Visibility based N-Body GPU Collision Detection

Mankyu Sung*

Keimyung University

E-mail : mksung@kmu.ac.kr

요 약

본 논문은 가시화 기반 LBVH(Linear Bounding Volume Hierarchy))을 이용한 빠른 GPU기반 N-body 충돌 체크 알고리즘을 제안한다. 본 알고리즘은 움직이는 n-body 개체에 대한 수정된 모튼코드(Morton code)를 이용하며, 이 모튼코드는, 일반적으로 사용되는 개체의 위치 정보뿐 아니라 이 개체가 스크린상에 차지하는 가시화 영역 정보를 이용하기 때문에, 카메라의 위치 및 방향에 따라 화면상에 차지하는 영역이 작은 개체에 대한 빠른 GPU기반 정렬(sorting)이 가능하게 된다. 실험을 통해, 본 논문에서 제안하는 방법이 기존 방법보다 15%이상 성능 향상이 있음을 알게 되었다

ABSTRACT

This paper propose a GPU-based N-body collision detection algorithm using LBVH (Linear Bounding Volume Hierarchy) technique. This algorithm introduces a new modified Morton code scheme where the codes use an information about how much each body takes a space in the screen space. This scheme improves the GPU sorting performance of the N-Body because it culls out invisible objects in natural manner. Through the experiments, we verifies that the proposed algorithms can have at least 15% performance improvement over the existing methods

키워드

Morton codes, GPU, Radix sorting, Linear Bounding Volume Hierarchy

Ⅰ. 서 론

움직이는 물체에 대한 빠른 충돌감지는 물리 시뮬레이션을 비롯한 군중 시뮬레이션에서 가장 필요한 기술 중에 하나이다. 본 연구에서는 파티클 N-Body 시뮬레이션과 같이 개체가 스스로 움직이는 애니메이션에서의 개체간 빠른 충돌체크를 하는 기법을 제안한다. 개체들은 동일한 크기 혹은 다른 크기를 가지고 있으며 바운딩 스피어(bounding sphere)를 이용해 근사화 되어 있다고 가정한다. 본 논문의 목표는 10,000개 이상의 바운딩 스피어들간의 충돌 감지를 실시간 수행하는 것을 목표로 하며, 이를 위해 GPU의 병렬 처리 기법을 최대한 활용 한다.

* speaker

Ⅱ. 관련 연구

충돌 감지를 위해 가장 많이 사용되는 기법은 BVH(Bounding Volume Hierarchy)방법으로, 움직이는 복잡한 물체를 계층구조로 나누어 트리 형태로 나타낸 후에, 이 트리에 대한 순회를 통해 빠른 충돌 감지를 수행하는 기법이다. N-Body 각 개체는 독립적으로 움직이고 있으며, 이들에 대한 충돌이 일어날 가능성이 있는 pair를 빠르게 얻는 것이 가장 중요한 부분이다. 일반적으로 GPU 병렬 처리 이용해 성능 향상을 하기 위한 방법으로 LBVH(Linear Bounding Volume Hierarchy)이 제안되었다. LBVH는 Z-order curve라는 Space-filling 기법을 이용하여 움직이는 개체를 빠르게 정렬 한 후 이에 대한 트리 구조를 병렬로 구성하는 방법

이다 [2, 3]. 기본 아이디어는 먼저 leaf 노드 (각각 하나의 객체에 해당)가 트리에 나타나는 순서를 선택한 다음이 이 leaf노드의 비트(bit)를 분석한 후 내부 노드(internl node)를 생성하게 된다. 일반적으로 3D 공간에서 서로 가까이 위치한 개체들은 계층 구조 근처에 위치하기 때문에 공간 채우기 곡선을 따라 정렬하게 되며, 이 때 각 개체의 위치 정보를 space-filling 커브로 바꾸기 위해 모튼 코드(morton code)를 이용하게 된다. 빠른 모튼 코드 생성은 전반적인 성능 향상을 위해 가장 중요한 부분이며, Lookup 테이블, 비트 쉬프트와 같은 방법을 통해 개체별로 할당된다.[1]

III. 제안하는 알고리즘

기본적인 모튼 코드는 단순히 현재 개체의 3차원 위치 정보 (x, y, z)만을 이용하게 되나, 본 논문에서는 카메라의 현재 위치와 개체의 현재 위치를 이용하여 두 단계로 충돌 감지를 수행하도록 제안한다. 첫 단계는 컬링(Culling) 단계로, 화면 상에 해당 개체가 차지하는 영역이 임계 값 이내에 들어올 정도로 작다면 충돌점검에서 제외시키는 단계이다. 두 번째 단계는 임계 값 이상으로 해당 개체가 클 경우더라도, 화면상에 차지하는 면적정보를 모튼 코드의 상위 비트에 지정 함으로서 정렬 시 속도 향상을 하도록 한다. 또한, 변화가 큰 축을 상위 비트에 지정 함으로서, 축 변화에 따른 다른 모튼 코드를 설정하도록 한다. 그림 1은 일반적인 모튼 코드를 이용한 space-filling curve모습을 나타낸다.

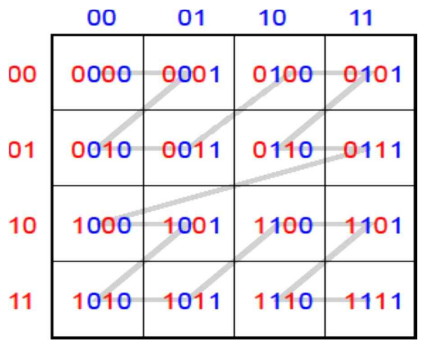

그림 1. 모튼 코드를 이용한 space-filling curve

화면상에 움직이는 개체가 차지하는 면적은 bounding sphere에 투사 함으로 구할 수 있다. 투사된 모습은 ellipse형태를 취하며, 이 ellipse의 장축(*M*)과 단축(*m*)을 아래의 수식을 통해 계산된다.

$$m = \sqrt{\frac{-r^2(r^2 - |o|^2}{(|o|^2 - o_z^2)(r^2 - o_z^2)(r^2 - |o|^2)}} \quad (1)$$

$$M = \sqrt{\frac{-r^2(r^2 - |o|^2}{(|o|^2 - o_z^2)(r^2 - o_z^2)(r^2 - o_z^2)}}$$

장축과 단축을 이용한 면적 계산은 수식 (2)와 같다.

$$a = \pi m M \quad (2)$$

수정된 모튼 코드는 이 면적값을 인코딩하여 사용한다. 수정된 모튼 코드 생성 알고리즘은 Algorithm 1에 나타냈다.

Algorithm 1

```
FUNCTION init
  //환경 크기 : sx,sy,sz
  s_x = s(x) s_y = s(y) s_z = s(z) a = p(a)
  sorting(sx,sy,sz,s0,s1,s2) //
  q0 = 2^8/s0
  q1 = 2^8/s1
  q2 = 2^8/s2
  q3 = 2^8/a
END
FUNCTION expand(int x)
  int v0=0, mask=1
  for (i=0 to 7)
    v = v | (x & mask << (3.i))
    mask = mask << 1
  end
END
FUNCTION Encode(p)
  int v0=q0*p[s0]
  int v1=q1*p[s1]
  int v2=q2*p[s2]
  int v3=q3*p[a]
  int r = expand(v0 << 3) | expand(v1 << 2) |
          expand(v2 << 1) | expand(v3)
  return r
END
```

IV. 실험 결과

제안하는 알고리즘의 성능을 비교하기 위해 멀 multi-agent 시뮬레이션 시스템을 개발하여 실험을 수행 하였다. 실험에 사용한 시뮬레이션 시스템은 그래픽스 API로서 OpenGL를 사용하였으며 GPU상에서 렌더링 및 계산을 수행하기 위해서 Compute shader를 개발하였다. 그림 2는 시스템의 실험 화면을 캡처 한 것이다.

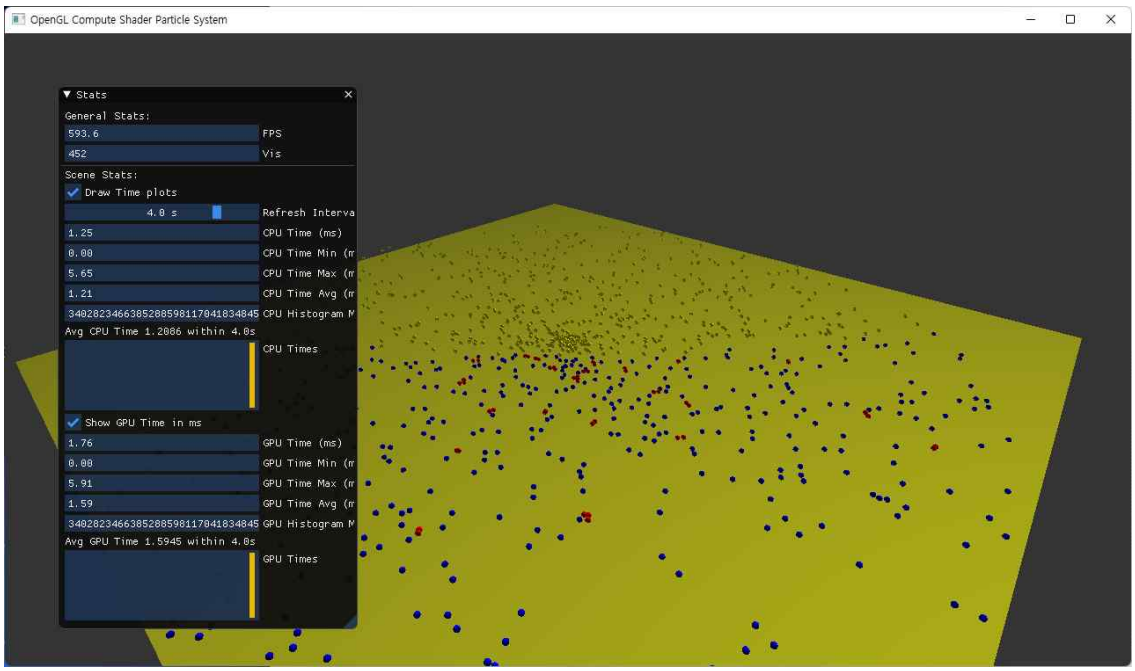

그림 2. 개발 시스템

그림 2에서 sphere의 전체 갯수는 2000개 이며, 이들은 원을 그리며 랜덤 속도로 움직이 움직이도록 하였다. 빨간색으로 처리된 sphere들은 충돌 가능성이 있는 두 pair를 나타낸 것이며 카메라의 위치 및 해당 개체의 위치에 따라 화면상에 차지 하는 면적이 임계값 보다 작은 경우 노란색으로 나타내 었다. 그림 3은 제안한 두 단계 알고리즘과 원래 위치정보만을 이용하여 모든 코드를 생성해 시뮬레이션 했었을 때를 비교 한 그래프 이다. 본 그래프에서 x축은 sphere의 개수를 의미 하며 y축은 frame rate를 의미한다. 본 그래프에서 나타나 있듯이 제안한 알고리즘이 대략적으로 15%정도의 성능 향상이 있음을 밝혔다.

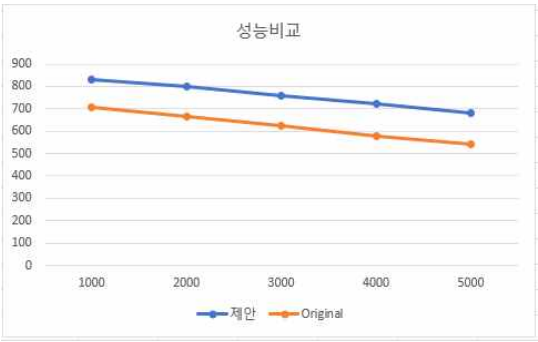

그림 3. 성능 비교

Ⅴ. 결 론

본 논문에서는 가시화 기반 수정된 모든 코드를 이용한 N개의 움직이는 물체에 대한 빠른 충돌 감지 기법을 제안하였다. 본 알고리즘의 핵심은 움직이는 물체에 bounding sphere를 씌운 후 이를 화면상에 투사하여, 화면상에 차지하는 면적을 계산 한 후 이 면적값을 모튼코드의 상위 비트에 할당하여 빠른 정렬이 가능하도록 하였다. 추 후 연구를 통해 100,000이상의 움직이는 물체에 대해 30프레임 이상의 성능이 나올 수 있도록 최적화를 수행 할 예정이다.

Acknowledgement

이 논문은 이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2021R1A2C1012316).

References

[1] M. Vinkler, J. Bittner, V. Harvran, “Extended Morton codes for High Performance Bounding Volume Hierachy Construction,” In *Proceeding of HPG ’17, Los Angeles, CA, USA, 2017*

[2] J. Pantaleoni and D. Luekbe, “HLBVH: Hierarhical LBVH Construction for Real-Time Ray Tracing of Dynamic Geometry” in *Proceeding of HPG 10,* 2010.
[3] J. Jakob and M Guthe, “Optimizing LBVH-Construction and Hierarchy-Traversal to Accelerate kNN Query on Point Clouds using the GPU”, *Computer Graphics Forum, Vol 40. No. 1, 2021*