(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0079884
(43) 공개일자 2012년07월16일

(51) 국제특허분류(Int. Cl.)
G06F 9/44 (2006.01) *G06F 17/10* (2006.01)
(21) 출원번호 10-2011-0001209
(22) 출원일자 2011년01월06일
심사청구일자 2011년01월06일

(71) 출원인
이화여자대학교 산학협력단
서울특별시 서대문구 이화여대길 52 (대현동, 이화여자대학교)
(72) 발명자
김영준
서울특별시 마포구 창전로 45, 서강 한화오벨리스크 102동 1104호 (창전동)
Fuchang Liu
서울특별시 서대문구 이화여대길 52, 이화여대 아산공학관 121 (대현동)
(74) 대리인
김인철

전체 청구항 수 : 총 23 항

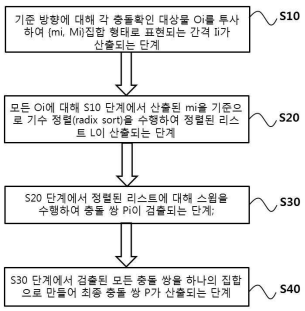
(54) 발명의 명칭 **GPU에서 충돌 감지를 수행하는 방법**

(57) 요 약

본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 기준 방향에 대해 각 충돌확인 대상물 O_i를 투사하여 {m_i, M_i}집합 형태로 표현되는 간격 I_i가 산출되는 S10 단계, 모든 O_i에 대해 S10 단계에서 산출된 m_i을 기준으로 기수 정렬(radix sort)을 수행하여 정렬된 리스트 L이 산출되는 S20 단계, S20 단계에서 정렬된 리스트에 대해 스윕을 수행하여 충돌 쌍 P_i이 검출되는 S30 단계 및 S30 단계에서 검출된 모든 충돌 쌍을 하나의 집합으로 만들어 최종 충돌 쌍 P가 산출되는 S40 단계를 포함한다.

GPU 상에서 다수의 스레드(thread)드의 블록을 이용하여 다수의 충돌확인 대상물에 대한 SaP를 병렬적으로 수행할 수 있다.

대 표 도 - 도1



이 발명을 지원한 국가연구개발사업

과제고유번호	2008-F-033-01
부처명	지식경제부
연구사업명	산업원천기술개발사업
연구과제명	e-Entertainment를 위한 실시간 물리 시뮬레이션 기술 개발
주관기관	이화여자대학교 산학협력단
연구기간	2010.03.01 ~ 2011.02.28

특허청구의 범위

청구항 1

충돌 감지를 수행하는 방법에 있어서,

기준 방향에 대해 각 충돌확인 대상물 O_i를 투사하여 $\{m_i, M_i\}$집합 형태로 표현되는 간격 I_i가 산출되는 S10 단계;

모든 O_i에 대해 상기 S10 단계에서 산출된 m_i을 기준으로 기수 정렬(radix sort)을 수행하여 정렬된 리스트 L이 산출되는 S20 단계;

상기 S20 단계에서 정렬된 리스트에 대해 스윕을 수행하여 충돌 쌍 P_i이 검출되는 S30 단계; 및

상기 S30 단계에서 검출된 모든 충돌 쌍을 하나의 집합으로 만들어 최종 충돌 쌍 P가 산출되는 S40 단계를 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

(여기서, n개의 충돌확인 대상물 중 i번째 충돌확인 대상물을 O_i라고 하고, m_i 및 M_i는 극점의 기준 방향에서의 위치를 나타냄)

청구항 2

제1항에 있어서,

상기 S30 단계는

i<j인 충돌확인 대상물 O_j에 대해서 $M_i < m_j$ 일 때까지 간격 I_i 내에서 L이 스윕(sweep)되는 S30-1 단계; 및

$m_j \in I_i$ 인 O_j에 대해 $O_i \cap O_j \neq \emptyset$라면 (O_i, O_j) 쌍이 P_i에 추가되는 S30-2 단계를 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 3

제1항에 있어서,

상기 각 O_i에 대하여 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 4

제1항에 있어서,

상기 각 O_i에 대하여 같은 충돌 감지 수행 횟수를 갖는 파티션으로 나누어 각 파티션 마다 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 5

제4항에 있어서,

상기 L에 대해 이진 검색을 수행하여 파티션 개수를 결정하는 것으로, 파티션 개수는 $\left\lceil \frac{p-i}{\tau} \right\rceil$ 값인 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

(여기서 $P-i$는 잠재적 충돌 쌍의 개수이고, τ는 하나의 스레드가 할당될 수 있는 충돌 쌍의 최대 개수임)

청구항 6

제1항에 있어서,

상기 S30 단계에서 스윕 방향은 주성분분석법(PCA: Principal Component Analysis)을 이용하여 결정되는 것을

특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 7

제6항에 있어서,

상기 스윕 방향은 아래의 w1 벡터값으로 결정되는 것으로, 상기 w1 벡터값은 PCA의 첫 번째 주요 성분방향과 일치하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

$$\begin{aligned}\mathbf{w}_1 &= \arg\max_{\|\mathbf{w}\|=1} Var\{\mathbf{w}^T\mathbf{X}\} \\ &= \arg\max_{\|\mathbf{w}\|=1} E\{\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\}\end{aligned}$$

(여기서, w_1는 공분산 행렬(covariance matrix) $C = XX^T$의 가장 큰 아이겐 값에 해당하는 아이겐 벡터임)

청구항 8

제1항에 있어서,

상기 S10 단계 전에 작업영역을 분할하는 S5 단계를 더 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 9

제8항에 있어서,

상기 S5 단계는 상기 S30 단계에서 스윕 방향을 d라고 할 때, d에 평행한 평면에 의해 잘리는 m × m 그리드 셀로 분할되는 S5-1 단계; 및

상기 S5-1 단계에서 분할된 셀의 경계를 지나가는 충돌확인 대상물이 있는 경우 상기 간격 I_i는 d 방향인 (j-1)×l 을 이용해 시프팅되는 S5-2 단계를 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

(여기서, $m = \lceil \frac{n}{64K} \rceil$이고 n은 충돌확인 대상물의 개수이고, j는 O_i가 속한 셀 C_j의 인덱스이고, l은 d 방향에 따른 작업영역의 크기임)

청구항 10

제1항에 있어서,

상기 S30 단계 전에

실제 교차하지 않지만 투사된 간격이 겹치는 충돌확인 대상물을 제거하기 위한 셀 분할을 수행하는 단계를 더 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 11

제10항에 있어서,

상기 셀 분할을 수행하는 단계는 하나의 셀이 기준 개수를 갖는 서브셀로 나누어지고, 같은 서브셀을 공유하는 충돌확인 대상물이 아닌 경우 충돌확인 대상물에서 제거하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 12

충돌 감지를 수행하는 방법에 있어서,

기준 방향에 대해 각 충돌확인 대상물 O_i를 투사하여 {m_i, M_i}집합 형태로 표현되는 간격 I_i가 산출되는 S100 단계;

모든 O_i에 대해 상기 S100 단계에서 산출된 m_i을 기준으로 기수 정렬(radix sort)을 수행하여 정렬된 리스트 L이 산출되는 S200 단계;

상기 S200 단계에서 정렬된 리스트에 대해 움직이는 충돌확인 대상물에 병렬적 스윕을 수행하여 충돌 쌍 $P_M(t)$이 검출되는 S300 단계;

정적인 충돌확인 대상물 집합 $O_s(t)$ 내에서 간섭 쌍(interfering pair) $P_s(t)$이 검출되는 S400 단계; 및

상기 S300 단계에서 검출된 $P_M(t)$과 상기 S400 단계에서 검출된 $P_s(t)$를 합집합 연산하여 최종 충돌 쌍 P가 산출되는 S500 단계를 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

(여기서, n개의 충돌확인 대상물 중 i번째 충돌확인 대상물을 O_i라고 하고, m_i 및 M_i는 극점의 기준 방향에서의 위치를 나타냄)

청구항 13

제12항에 있어서,

상기 S300 단계는

$O_s(t)$에 속하는 모든 O_i에 대하여, 병렬적 스위핑을 통해 정적인 충돌확인 대상물 집합 $O_s(t)$과 움직이는 충돌확인 대상물 집합 $O_m(t)$간에 충돌 쌍의 집합 $P_{sm}(t)$이 검출되는 S300-1 단계;

$O_m(t)$에 속하는 모든 O_i에 대하여, 병렬적 스위핑을 통해 움직이는 충돌확인 대상물 집합 $O_m(t)$과 모든 다른 충돌확인 대상물 집합 O를 비교하여 충돌 쌍의 집합 $P_{m*}(t)$이 검출되는 S300-2 단계; 및

상기 $P_{sm}(t)$과 $P_{m*}(t)$을 합집합 연산하는 S300-3 단계를 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 14

제12항에 있어서,

상기 S400 단계는 이전 시간의 충돌 쌍$P(t-1)$ 후에 정적인 충돌확인 대상물 집합 $O_s(t)$ 내에서 간섭 쌍(interfering pair) $P_s(t)$을 검출하는 것으로, $P_s(t)$는 $P(t-1)$에서 아래의 식으로 표현되는 $P'_M(t-1)$을 차집합 연산하여 검출되는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

$P'_M(t-1) \equiv \{\forall(O_i, O_j) \in P(t-1) \mid O_i \in O_m(t) \vee O_j \in O_m(t)\}$

청구항 15

제12항에 있어서,

상기 각 O_i에 대하여 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 16

제12항에 있어서,

상기 각 O_i에 대하여 같은 충돌 감지 수행 횟수를 갖는 파티션으로 나누어 각 파티션 마다 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 17

제16항에 있어서,

상기 L에 대해 이진 검색을 수행하여 파티션 개수를 결정하는 것으로, 파티션 개수는 $\left\lceil \frac{p-i}{\tau} \right\rceil$ 값인 것을 특징

으로 하는 GPU에서 충돌 감지를 수행하는 방법.

(여기서 $P - i$는 잠재적 충돌 쌍의 개수이고, τ는 하나의 스레드가 할당될 수 있는 충돌 쌍의 최대 개수임)

청구항 18

제12항에 있어서,

상기 S100 단계 전에 작업영역을 분할하는 S50 단계를 더 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 19

제18항에 있어서,

상기 S50 단계는 상기 S300 단계에서 스윕 방향을 d라고 할 때, d에 평행한 평면에 의해 잘리는 m × m 그리드 셀로 분할되는 S50-1 단계; 및

상기 S50-1 단계에서 분할된 셀의 경계를 지나가는 충돌확인 대상물이 있는 경우 상기 간격 I_i는 d 방향인 (j-1)×l 을 이용해 시프팅되는 S50-2 단계를 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

(여기서, $m = \lceil \frac{n}{64K} \rceil$이고 n은 충돌확인 대상물의 개수이고, j는 O_i가 속한 셀 C_j의 인덱스이고, l은 d 방향에 따른 작업영역의 크기임)

것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 20

제12항에 있어서,

상기 S300 단계의 스윕 방향은 주성분분석법(PCA: Principal Component Analysis)을 이용하여 결정되는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 21

제20항에 있어서,

상기 스윕 방향은 아래의 w1 벡터값으로 결정되는 것으로, 상기 w1 벡터값은 PCA의 첫 번째 주요 성분방향과 일치하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} Var\{\mathbf{w}^T\mathbf{X}\}$$
$$= \arg\max_{\|\mathbf{w}\|=1} E\{\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\}$$

(여기서, w_1는 공분산 행렬(covariance matrix) $C = XX^T$의 가장 큰 아이겐 값에 해당하는 아이겐 벡터임)

청구항 22

제12항에 있어서,

상기 S300 단계 전에 실제 교차하지 않지만 투사된 간격이 겹치는 충돌확인 대상물을 제거하기 위한 셀 분할을 수행하는 단계를 더 포함하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

청구항 23

제22항에 있어서,

상기 셀 분할을 수행하는 단계는 하나의 셀이 기준 개수를 갖는 서브셀로 나누어지고, 같은 서브셀을 공유하

는 충돌확인 대상물이 아닌 경우 충돌확인 대상물에서 제거하는 것을 특징으로 하는 GPU에서 충돌 감지를 수행하는 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 충돌 감지(Collision Detection)에 관한 것이다. 특히 본 발명은 GPU 상에서 수행가능한 충돌 감지에 관한 것으로, 많은 충돌확인 대상물에 대하여 병렬적으로 충돌 감지 수행하는 방법에 관한 것이다.

배 경 기 술

[0002] 충돌감지는 공간에서 이동하는 충돌확인 대상물(object) 간에 발생하는 겹침 내지 간섭(interference)을 결정하는 영역의 과제이다. 충돌감지는 컴퓨터 그래픽, 컴퓨터 애니메이션, 가상 현실, 형상 모델링(geometric modeling), 햅틱스(haptics), 로봇공학 등에 다양한 분야에서 문제가 되는 분야이다.

[0003] 충돌감지 기술은 크게 광위상(broad-phase) 알고리즘과 협위상(narrow-phase)알고리즘으로 분류된다. 광위상 알고리즘은 n개의 충돌확인 대상물 간에서 가능한 교차 쌍(intersecting pairs)을 결정하는 것이고, 협위상 알고리즘은 광위상 알고리즘에 의해 결정된 충돌확인 대상물들 중에서 한 쌍을 결정하는 것이다. 본 발명은 광위상 알고리즘에 관한 것이고, 광위상 알고리즘은 소위 'n-body 충돌 선택(collision culling)'이라고도 불린다.

[0004] DEM(Distince-element Method), SPH(Smoothed Particle Hydrodynamics) 및 MPS(Moving-particle Semi-implicit)와 같은 입자(particle) 기반 시뮬레이션 영역 등에서 수백만 개의 입자를 대상으로 한 충돌확인 대상물들의 흐름 시뮬레이션이 요구된다. 또한 MMORPG(Massively Multi-Player Online Role-Playing Game)에서는 최소 수 천명의 사용자가 로그인하여 같은 서버에서 활동하게 되면서 수만 개의 캐릭터 및 주위 충돌확인 대상물과 상호작용을 수행한다. 이러한 예를 포함한 많은 응용분야에서 매우 많은 충돌확인 대상물 집합을 대상으로 하면서도, 처리 속도가 빠른 충돌감지 기술이 요구되고 있다.

[0005] 가장 잘 알려진 광위상 충돌감지 알고리즘은 스윕 앤 프룬(SaP: Sweep and Prune) 방법 및 공간분할법(Spatial Subdivision)이 있다. SaP는 충돌확인 대상물이 낮은 차원 공간(일반적으로 단일 차원 공간)에 투사되면, 상기 차원 공간 축을 따라 초평면(hyperplane) 스윕(sweep)하는 겹침 테스트(overlap test)를 수행하여 차원 공간을 줄이는 접근방법이다. 공간분할법은 어떤 형태의 그리드에 등록되어 있는 충돌확인 대상물에 대해 상기 그리드 내부에서 수행되는 지역 교차 테스트(local intersection test)를 수행하는 공간 해싱(spatial hashing) 기법이다.

[0006] SaP는 스윕 동작시 움직이는 충돌확인 대상물이 높은 공간 일관성(spatial coherence)을 갖는 경우에 효과적이나, 반대 경우는 효율이 낮아지는 문제점이 있다. 공간분할법은 같은 크기의 충돌확인 대상물에 대해 효과적이나, 복잡한 체계를 갖는 구조에 대해서는 효율이 낮아지는 문제점이 있다.

[0007] 따라서, 충돌확인 대상물의 크기나 이동의 성향에 따른 제한 없이 적용가능한 빠른 충돌감지 알고리즘이 필요하다고 하겠다.

발명의 내용

해결하려는 과제

[0008] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 다음과 같은 해결과제를 목적으로 한다.

[0009] 첫째, 본 발명은 SaP 알고리즘과 공간분할법의 장점을 취합하여 충돌감지의 효과 및 속도를 높이고자 한다.

[0010] 둘째, 본 발명은 GPU에서 병렬 SaP 알고리즘을 수행하여 충돌감지의 효과 및 속도를 높이고자 한다.

[0011] 셋째, 본 발명은 충돌감지에 있어서, 충돌확인 대상물을 쉽게 추가하거나 제거할 수 있는 알고리즘을 제공하고자 한다.

[0012] 본 발명의 해결과제는 이상에서 언급된 것들에 한정되지 않으며, 언급되지 아니한 다른 해결과제들은 아래의 기재로부터 당업자에게 명확하게 이해되어 질 수 있을 것이다.

과제의 해결 수단

[0013] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 기준 방향에 대해 각 충돌확인 대상물 O_i를 투사하여 $\{m_i, M_i\}$집합 형태로 표현되는 간격 I_i가 산출되는 S10 단계를 포함한다.

[0014] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 모든 O_i에 대해 S10 단계에서 산출된 m_i을 기준으로 기수 정렬(radix sort)을 수행하여 정렬된 리스트 L이 산출되는 S20 단계를 포함한다.

[0015] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S20 단계에서 정렬된 리스트에 대해 스윕을 수행하여 충돌 쌍 P_i이 검출되는 S30 단계를 포함한다.

[0016] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S30 단계에서 검출된 모든 충돌 쌍을 하나의 집합으로 만들어 최종 충돌 쌍 P가 산출되는 S40 단계를 포함한다.

[0017] 여기서, n개의 충돌확인 대상물 중 i번째 충돌확인 대상물을 O_i라고 하고, m_i 및 M_i는 극점의 기준 방향에서의 위치를 나타낸다.

[0018] 본 발명에 따른 S30 단계는i<j인 충돌확인 대상물 O_j에 대해서 $M_i < m_j$ 일 때까지 간격 I_i 내에서 L이 스윕(sweep)되는 S30-1 단계 및 $m_j \in I_i$ 인 O_j에 대해 $O_i \cap O_j \neq \emptyset$라면 (O_i, O_j) 쌍이 P_i에 추가되는 S30-2 단계를 포함한다.

[0019] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 각 O_i에 대하여 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행하는 것을 특징으로 한다.

[0020] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 각 O_i에 대하여 같은 충돌 감지 수행 횟수를 갖는 파티션으로 나누어 각 파티션 마다 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행할 수도 있다.

[0021] 본 발명에 따른 파티션 개수는 L에 대해 이진 검색을 수행하여 파티션 개수를 결정하는 것으로, 파티션 개수는 $\lceil \frac{p-i}{\tau} \rceil$ 값인 것을 특징으로 한다.

[0022] 여기서 $P-i$는 잠재적 충돌 쌍의 개수이고, τ는 하나의 스레드가 할당될 수 있는 충돌 쌍의 최대 개수이다.

[0023] 본 발명에 따른 S30 단계에서 스윕 방향은 주성분분석법(PCA: Principal Component Analysis)을 이용하여 결정되는 것을 특징으로 한다.

[0024] 본 발명에 따른 스윕 방향은 아래의 w1 벡터값으로 결정되는 것으로, w1 벡터값은 PCA의 첫 번째 주요 성분방향과 일치하는 것을 특징으로 한다.

[0025]
$$\begin{aligned} \mathbf{w}_1 &= \arg\max_{\|\mathbf{w}\|=1} Var\{\mathbf{w}^T\mathbf{X}\} \\ &= \arg\max_{\|\mathbf{w}\|=1} E\{\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\} \end{aligned}$$

[0026] 여기서, w_1는 공분산 행렬(covariance matrix) $C = XX^T$의 가장 큰 아이겐 값에 해당하는 아이겐 벡터이다.

[0027] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S10 단계 전에 작업영역을 분할하는 S5 단계를 더 포함할 수 있다.

[0028] 본 발명에 따른 S5 단계는 S30 단계에서 스윕 방향을 d라고 할 때, d에 평행한 평면에 의해 잘리는 m × m 그리드 셀로 분할되는 S5-1 단계 및 S5-1 단계에서 분할된 셀의 경계를 지나가는 충돌확인 대상물이 있는 경우 간격 I_i는 d 방향인 (j-1)×l 을 이용해 시프팅되는 S5-2 단계를 포함한다.

[0029] 여기서, $m = \lceil \frac{n}{64K} \rceil$이고 n은 충돌확인 대상물의 개수이고, j는 O_i가 속한 셀 C_j의 인덱스이고, l은 d 방향에 따른 작업영역의 크기이다.

[0030] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S30 단계 전에 실제 교차하지 않지만 투사된 간격이 겹치는 충돌확인 대상물을 제거하기 위한 셀 분할을 수행하는 단계를 더 포함할 수 있다.

[0031] 본 발명에 따른 셀 분할을 수행하는 단계는 하나의 셀이 기준 개수를 갖는 서브셀로 나누어지고, 같은 서브셀을 공유하는 충돌확인 대상물이 아닌 경우 충돌확인 대상물에서 제거하는 것을 특징으로 한다.

[0032] 본 발명의 다른 측면에서 GPU에서 충돌 감지를 수행하는 방법은 기준 방향에 대해 각 충돌확인 대상물 O_i를 투사하여 $\{m_i, M_i\}$집합 형태로 표현되는 간격 I_i가 산출되는 S100 단계를 포함한다.

[0033] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 모든 O_i에 대해 S100 단계에서 산출된 m_i을 기준으로 기수 정렬(radix sort)을 수행하여 정렬된 리스트 L이 산출되는 S200 단계를 포함한다.

[0034] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S200 단계에서 정렬된 리스트에 대해 움직이는 충돌확인 대상물에 병렬적 스윕을 수행하여 충돌 쌍 $P_M(t)$이 검출되는 S300 단계를 포함한다.

[0035] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은정적인 충돌확인 대상물 집합 $O_s(t)$ 내에서 간섭 쌍(interfering pair) $P_s(t)$이 검출되는 S400 단계를 포함한다.

[0036] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S300 단계에서 검출된 $P_M(t)$과 S400 단계에서 검출된 $P_s(t)$를 합집합 연산하여 최종 충돌 쌍 P가 산출되는 S500 단계를 포함한다.

[0037] 여기서, n개의 충돌확인 대상물 중 i번째 충돌확인 대상물을 O_i라고 하고, m_i 및 M_i는 극점의 기준 방향에서의 위치를 나타낸다.

[0038] 본 발명에 따른 S300 단계는 $O_s(t)$에 속하는 모든 O_i에 대하여, 병렬적 스위핑을 통해 정적인 충돌확인 대상물 집합 $O_s(t)$과 움직이는 충돌확인 대상물 집합 $O_m(t)$간에 충돌 쌍의 집합 $P_{sm}(t)$이 검출되는 S300-1 단계, $O_m(t)$에 속하는 모든 O_i에 대하여, 병렬적 스위핑을 통해 움직이는 충돌확인 대상물 집합 $O_m(t)$과 모든 다른 충돌확인 대상물 집합 O를 비교하여 충돌 쌍의 집합 $P_{m*}(t)$이 검출되는 S300-2 단계 및 $P_{sm}(t)$과 $P_{m*}(t)$을 합집합 연산하는 S300-3 단계를 포함한다.

[0039] 본 발명에 따른 S400 단계는 이전 시간의 충돌 쌍$P(t-1)$ 후에 정적인 충돌확인 대상물 집합 $O_s(t)$ 내에서 간섭 쌍(interfering pair) $P_s(t)$을 검출하는 것으로, $P_s(t)$는 $P(t-1)$에서 아래의 식으로 표현되는 $P'_M(t-1)$을 차집합 연산하여 검출되는 것을 특징으로 한다.

[0040] $$P'_M(t-1) \equiv \{\forall(O_i, O_j) \in P(t-1) \mid O_i \in O_m(t) \vee O_j \in O_m(t)\}$$

[0041] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 각 O_i에 대하여 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행하는 것을 특징으로 한다.

[0042] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 각 O_i에 대하여 같은 충돌 감지 수행 횟수를 갖는 파티션으로 나누어 각 파티션 마다 GPU 스레드가 하나씩 할당되어 충돌 감지를 수행하는 것을 특징으로 한다.

[0043] 본 발명에 따른 파티션 개수는 L에 대해 이진 검색을 수행하여 파티션 개수를 결정하는 것으로, 파티션 개수는 $\left\lceil \frac{p-i}{\tau} \right\rceil$ 값인 것을 특징으로 한다.

[0044] 여기서 $P-i$는 잠재적 충돌 쌍의 개수이고, τ는 하나의 스레드가 할당될 수 있는 충돌 쌍의 최대 개수이다.

[0045] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S100 단계 전에 작업영역을 분할하는 S50 단계를 더 포함할 수 있다.

[0046] 본 발명에 따른 S50 단계는 S300 단계에서 스윕 방향을 d라고 할 때, d에 평행한 평면에 의해 잘리는 m × m 그리드 셀로 분할되는 S50-1 단계 및 S50-1 단계에서 분할된 셀의 경계를 지나가는 충돌확인 대상물이 있는 경우 간격 I_i는 d 방향인 $(j-1)\times l$ 을 이용해 시프팅되는 S50-2 단계를 포함한다.

[0047] 여기서, $m = \lceil \frac{n}{64K} \rceil$이고 n은 충돌확인 대상물의 개수이고, j는 O_i가 속한 셀 C_j의 인덱스이고, l은 d 방향에 따른 작업영역의 크기이다.

[0048] 본 발명에 따른 S300 단계의 스윕 방향은 주성분분석법(PCA: Principal Component Analysis)을 이용하여 결정되는 것을 특징으로 한다.

[0049] 본 발명에 따른 스윕 방향은 아래의 w1 벡터값으로 결정되는 것으로, w1 벡터값은 PCA의 첫 번째 주요 성분방향과 일치하는 것을 특징으로 한다.

[0050]
$$\begin{aligned} \mathbf{w}_1 &= \arg\max_{\|\mathbf{w}\|=1} Var\{\mathbf{w}^T\mathbf{X}\} \\ &= \arg\max_{\|\mathbf{w}\|=1} E\{\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\} \end{aligned}$$

[0051] 여기서, w_1는 공분산 행렬(covariance matrix) $C = XX^T$의 가장 큰 아이겐 값에 해당하는 아이겐 벡터이다.

[0052] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 S300 단계 전에 실제 교차하지 않지만 투사된 간격이 겹치는 충돌확인 대상물을 제거하기 위한 셀 분할을 수행하는 단계를 더 포함할 수 있다.

[0053] 본 발명에 따른 셀 분할을 수행하는 단계는 하나의 셀이 기준 개수를 갖는 서브셀로 나누어지고, 같은 서브셀을 공유하는 충돌확인 대상물이 아닌 경우 충돌확인 대상물에서 제거하는 것을 특징으로 한다.

발명의 효과

[0054] 본 발명에 따른 GPU에서 충돌 감지를 수행하는 방법은 다음과 같은 효과를 갖는다.

[0055] 첫째, GPU 상에서 다수의 스레드(thread)의 블록을 이용하여 다수의 충돌확인 대상물에 대한 SaP를 병렬적으로 수행할 수 있다.

[0056] 둘째, PCA(Principal Component Analysis)를 이용한 최적의 스윕 방향을 결정하여 SaP 알고리즘의 효율이 높다.

[0057] 셋째, 병렬 SaP에 2 단계 공간분할법을 병합하여 공간 제거에서 발생 가능한 밀집된 투사 간격의 문제가 완화된다.

[0058] 넷째, 시뮬레이션 환경에서 충돌대상이 되는 충돌확인 대상물을 추가하거나, 제거하여 특정 목적에 맞는 충돌감지가 수행된다.

[0059] 본 발명의 효과는 이상에서 언급된 것들에 한정되지 않으며, 언급되지 아니한 다른 효과들은 아래의 기재로부터 당업자에게 명확하게 이해되어 질 수 있을 것이다.

도면의 간단한 설명

[0060] 도 1은 본 발명에 따른 병렬 SaP 알고리즘의 일 예에 대한 순서도이다.

도 2는 본 발명에 따른 병렬 SaP 알고리즘에 의해 충돌 여부에 대한 테스트가 수행되는 충돌 쌍을 도시한다.

도 3은 파티션을 기준으로 불균형한 충돌 쌍 테스트를 분배하는 과정의 일 예를 도시한 것이다.

도 4는 최적의 스윕 방향을 설명하기 위한 예시도이다.

도 5는 움직임 일관성에 설명된 단계를 포함하는 또 다른 충돌 감지 방법이다.

도 6(a)는 2 × 2 작업영역 분할을 나타내는 그래프이고, 도 6(b)는 SaP를 수행하기 위하여 셀이 이동하는 과정에 대한 일 예를 도시한 그래프이다.

도 7은 본 발명에 따른 셀 분할의 일 예를 도시한 그래프이다.

도 8은 본 발명의 방법과 다른 충돌 감지 알고리즘의 수행 성능을 비교한 막대그래프이다.

도 9는 충돌확인 대상물 중 일부가 움직이는 경우 본 발명의 방법에 따른 충돌 감지 수행 결과를 도시한 것이

다.

도 10은 본 발명의 최적 스윕 방향을 사용한 수행 결과와 x축을 일괄적으로 스윕 방향으로 설정한 경우를 비교한 막대 그래프이다.

도 11은 본 발명의 스레드 할당을 적용한 경우와 적용하지 않은 경우의 충돌 감지 알고리즘 시간을 비교한 것이다.

발명을 실시하기 위한 구체적인 내용

[0061] 본 명세서에서 사용되는 용어에서 단수의 표현은 문맥상 명백하게 다르게 해석되지 않는 한 복수의 표현을 포함하는 것으로 이해되어야 하고, "포함한다" 등의 용어는 설시된 특징, 개수, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함을 의미하는 것이지, 하나 또는 그 이상의 다른 특징들이나 개수, 단계 동작 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 배제하지 않는 것으로 이해되어야 한다.

[0062] **일반적인 SaP 알고리즘**

[0063] SaP(Sweep and Prune) 방법은 정렬기반의 알고리즘으로서 특히 높은 공간 일관성(spatial coherence)을 갖는 경우 효과적이다. SaP 알고리즘의 복잡도는 $O(n+s)$로 알려져 있다. 여기서 n은 특정 모양을 갖는 충돌확인 대상물의 개수이고, s는 충돌확인 대상물의 정렬된 순서를 유지하기 위한 교환(swapping) 동작의 개수이다.

[0064] 3D 환경에서 n개의 충돌확인 대상물 중 i번째 충돌확인 대상물을 O_i라고 하면, SaP의 목표는 충돌확인 대상물 중 공간상에서 겹쳐진 모든 충돌확인 대상물의 쌍 P를 탐색하는 것이다. 따라서, $P = \{(O_i, O_j) \mid O_i \cap O_j \neq \emptyset, 1 \leq i \neq j \leq n\}$이고, 충돌확인 대상물 O_i는 축에 평행한 박스형태(AABB: Axis-Aligned Bounding Box) 또는 구(sphere)와 같은 단순한 형태를 갖는다고 하고, 따라서 $O_i \cap O_j \neq \emptyset$인지 여부 결정이 상수 시간 내로 수행된다고 가정한다.

[0065] 종래의 SaP 알고리즘은 AABB와 같은 충돌확인 대상물이 3 방향 차원 공간에서 투사된 간격이 겹치는지 여부를 결정하기 위한 것이다. 종래 일반적인 SaP 알고리즘(Baraff 992)은 다음과 같은 단계로 설명될 수 있다.

[0066] 종래 제1 단계: 기준 축에 대해 각 충돌확인 대상물 O_i의 크기를 투사한다. 예컨대, x축에 대하여 극점의 1D 간격을 생성한다. 간격은 $I_i = \{m_i, M_i\}$로 표현된다. 이때 m_i 및 M_i는 극점의 기준 방향(여기서는 x축 방향)의 위치를 나타낸다.

[0067] 종래 제2 단계: 모든 i에 대하여 m_i 및 M_i을 정렬하여 정렬된 리스트 L을 산출한다.

[0068] 종래 제3 단계: L을 스윕하고 활동 리스트 A를 유지한다. (a)L로부터 m_i가 검색되면 O_i를 A에 추가하고 A에 있는 모든 O_i를 충돌 쌍의 집합 P_x에 추가한다. (b)L로부터 M_i가 검색되면 O_i를 A에서 제거한다.

[0069] 종래 제4 단계: 상기 제1 단계 내지 제3 단계를 y 및 z 축에 대해서도 반복하여, 충돌 쌍 P_y 및 P_z를 산출한다. 최종 충돌 쌍 $P = P_x \cap P_y \cap P_z$을 검출한다.

[0070] 충돌확인 대상물의 움직임이 매우 일관적(highly coherent)이라면, 상기 제2단계 알고리즘이 삽입 정렬(insertion sort)로 수행되면 효과적이고(Cohen 1995), 상기 제3단계는 근처 O_i간에 교환 동작으로 대체될 수 있다. 충돌확인 대상물이 구형태인 경우, 구의 중앙으로부터의 거리를 비교하면 겹쳐지는지 여부를 확인할 수 있기 때문에 종래 제4 단계는 불필요하다. 나아가, 간격 I_i 는 투사된 중심위치에서 구의 지름에 해당한다.

[0071] 전술한 종래 SaP 알고리즘은 임의의 움직임을 갖는 매우 많은 충돌확인 대상물에 대해서 적합하지 않고, 전술한 종래 SaP 알고리즘은 기수 정렬(radix sort)을 사용하여 GPU(Graphics Processing Unit)에서 종래 제1 단계가 수행될 수 있다고 하더라도 전체 알고리즘의 병렬 수행에는 적합하지 않다.

[0072] 적합하지 않은 이유는 첫째, 스위핑 순서에 따라 결정되기 때문에 활동 리스트 A 유지는 병렬로 처리될 수 없

고, 둘째, 전술한 종래 제3 단계 및 종래 제4 단계는 현재의 GPU 구조에서 메모리 접근 비용을 줄이기에 적합하지 않으며, 셋째, 충돌확인 대상물이 매우 큰 경우, 높은 일관도를 갖는 움직임(highly coherent motion)에 대한 추정 및 연관된 교환 동작이 기하급수적으로 많아지기 때문이다.

[0073] 이하에서는 도면을 참조하면서 GPU에서 충돌 감지를 수행하는 방법에 관하여 구체적으로 설명하겠다.

[0074] **병렬 SaP 알고리즘**

[0075] 본 발명은 GPU 상에서 SaP 알고리즘을 병렬 수행하기 위한 것이다. GPU(Graphics Processing Units)는 단일 칩으로 구성된 프로세서로서, 본래 3 차원 애플리케이션을 위해 사용되었는데, 광원 효과와 객체의 변형 등 매번 다시 그려지는 3차원 장면을 만들어준다. 이런 것들은 수학적 계산이 집약되어 있는 작업으로서, 만약 GPU가 없다면 CPU에 많은 부담을 줄 수 있는 작업들이다. GPU는 이런 부담으로부터 CPU를 해방시키고, CPU의 사이클들을 다른 작업을 위해 사용될 수 있게 한다.

[0076] 전술한 종래 SaP 알고리즘과 비교하여 주요한 특징은 다음과 같다. 본 발명의 병렬 SaP 알고리즘은 종래 SaP 알고리즘과 달리 활동 리스트가 필요 없고, 전술한 일반 SaP 알고리즘에서의 종래 제2 단계에 해당하는 정렬된 리스트 L을 산출하는 과정은 m_i 하나만을 기준으로 O_i를 정렬하여 수행된다. 그리고 L을 스윕할 때 i<j인 충돌확인 대상물 O_j에 대해서만 $m_j \in I_i$ 인지 여부를 확인한다. 이는 (O_j, O_i)가 (O_i, O_j)로부터 유추될 수 있기 때문이다. 종래 제2단계에 해당하는 과정은 GPU 기반 기수 정렬에 의해 수행되어 교환 동작이 필요 없게 된다. 종래 제3 단계에 해당하는 스위핑 단계는 각 O_i 충돌확인 대상물을 각각의 간격 L_i 내에서 독립적으로 스위핑하여 수행된다.

[0077] 도 1은 본 발명에 따른 병렬 SaP 알고리즘의 일 예에 대한 순서도이다. 본 발명에 따른 병렬 SaP 알고리즘은 다음과 같은 단계로 구성된다.

[0078] S10 단계: 한 방향 축에 대해 각 충돌확인 대상물 O_i를 투사하여 간격 I_i를 산출한다. S10 단계의 기준 방향은 기본적으로 x,y 및 z 축 방향 중 하나이다.

[0079] S20 단계: 모든 O_i를 m_i을 기준으로, 기수 정렬(radix sort)을 이용하여 정렬된 리스트 L을 산출한다.

[0080] S30 단계: 각 O_i에 대하여 다음 각 소단계를 병렬로 수행한다. (a) i<j인 충돌확인 대상물 O_j에 대해서 $M_i < m_j$ 일 때까지 간격 I_i 내에서 L을 스윕(sweep)하는 S30-1 단계 및 (b) 어떤 O_j에 대해 $m_j \in I_i$ 인 경우, $O_i \cap O_j \neq \emptyset$라면 (O_i, O_j) 쌍을 P_i에 추가하는 S30-2 단계를 수행한다.

[0081] S40 단계: $P \equiv \cup P_i$, S3 단계에서 구해진 P_i를 모두 합집합하여 최종적인 충돌 쌍 P를 산출한다.

[0082] 각 충돌확인 대상물 O_i에 대하여 GPU 스레드를 하나씩 할당하고, 정렬된 리스트 L의 부분집합을 스윕하여 충돌하는 쌍 $P_i = \{(O_i, O_j) \mid i < j, m_j \in I_i\}$ (이때 $\exists\ j > i$ 이고 $M_i \geq m_j$ 인 조건을 만족함)에 대한 로컬 리스트를 산출한다.

[0083] 도 2는 본 발명에 따른 병렬 SaP 알고리즘에 의해 충돌 여부에 대한 테스트가 수행되는 충돌 쌍을 도시한다. O_1은 스레드$_1$에 의해 핸들링 되고, 충돌 쌍은 $\{(O_1, O_2), (O_1, O_3), (O_1, O_4), (O_1, O_5), (O_1, O_6)\}$이다. 이 충돌 쌍에 대해 충돌 여부를 테스트하게 된다.

[0084] 본 발명은 병렬 SaP 알고리즘의 산술 집중도(Arithmetic intensity)를 높이기 위하여, 종래 제3 단계 및 종래 제4 단계를 병합하여 수행한다. 여기서 산술 집중도란 메모리에 전송되는 워드(word) 마다 수행되는 수학적 연산의 수의 비를 의미한다. 종래 제3 단계와 종래 제4 단계의 병합은 모든 차원 공간에 대해 $O_i \cap O_j \neq \emptyset$ 인 각각의 충돌 쌍 $(O_i, O_j) \in P_i$ 에 대해 겹침 테스트(overlap test)를 수행하여 병합된다.

[0085] 충돌감지 대상이 AABB 형태가 아닌 구(sphere)인 경우, 종래 제3 단계와 종래 제4 단계의 병합은 더 단순해

진다. 2개의 구의 중심과 2개의 구의 반지름 합산값을 비교하면 병합이 가능하다.

[0086] 최종적으로 모든 스레드가 작업이 종료되면, 로컬 충돌 쌍을 합집합 처리하여 최종 리스트 P를 산출한다. 최근에 개발된 GPU를 이용하면 수만 개의 스레드가 병렬적으로 처리된다.

[0087] 나아가 본 발명의 병렬 SaP 알고리즘은 사전 처리 단계나 추가적인 질의 구조(querying structure)가 필요 없기 때문에, 충돌확인 대상물을 쉽게 추가하거나 제거하여 활용할 수 있다.

[0088] **스레드 할당**

[0089] 본 발명에 따른 병렬 SaP 알고리즘은 GPU 상에서 매우 많은 수의 연산 스레드를 이용하여 처리가 가능하다. 그러나 충돌확인 대상물이 서로 다른 크기를 갖는 경우, 큰 크기의 충돌확인 대상물은 작은 대상물 보다 많은 충돌 쌍(colliding pair)을 포함할 수 있다. 이러한 경우, 이 큰 크기의 충돌확인 대상물에 할당되는 하나의 스레드는 작은 충돌확인 대상물에 할당되는 스레드보다 많은 작업을 수행하고 작은 충돌확인 대상물에 할당되는 스레드는 수행할 작업이 없는(idle) 상태에 있게 된다. 결국 태스크 분배에 있어 불균형을 가져오고, GPU 연산을 위한 전력 소모를 일으킨다.

[0090] 이러한 문제는 더 많은 충돌 쌍 테스트를 수행하게 되는 충돌확인 대상물을 위해 더 많은 스레드를 할당하면 해결된다. 충돌 쌍의 정확한 개수를 충돌 테스트 이전에 알 수 없으므로, 충돌확인 대상물의 사이즈에 비례하여 충돌 쌍 테스트의 수행 횟수가 늘어난다고 가정할 수 있다.

[0091] 본 발명에서는 각 충돌확인 대상물에 대하여 충돌 쌍 테스트를 같은 테스트 횟수를 갖는 파티션(partition)으로 나누어 각 파티션 마다 하나의 스레드를 할당한다. 도 3은 파티션을 기준으로 불균형한 충돌 쌍 테스트를 분배하는 과정의 일 예를 도시한 것이다. 도 3을 살펴보면, O_1은 8개의 충돌확인 대상물과 충돌 테스트를 수행해야하고, O_4는 O_5와 하나의 충돌 테스트를 수행하면 된다. 그러나 O_1을 2개의 파티션으로 나눈다면 각 스레드는 단지 4번의 충돌 테스트를 수행하면 된다.

[0092] 구체적으로, 충돌확인 대상물 O_i가 충돌 테스트를 위한 쌍의 집합 $S_i = \{(O_i, On_1), \cdots, (O_i, On_j)\}$를 가진다고 가정한다. 이 집합은 부분집합 $S_i = Pi_1 \cup Pi_2 \cup \cdots \cup Pi_k$으로 나뉘는데 여기서, $Pi_k = \{(Oi, On_{t-1}), \cdots, (O_1, On_{t-1})\}$이고 $|S_i| = \sum |Pi_k|$이다. 나아가 $|Pi_k| < \tau$라는 제한을 더 두게되며 여기서, τ는 하나의 스레드에 할당될 수 있는 충돌 쌍의 최대 개수이다.

[0093] 본 발명은 각 파티션 Pi에서 충돌 테스트를 수행하는 하나의 스레드 T_i를 할당한다. 실제 τ는 GPU에서 동작할 수 있는 활동 스레드의 최대 개수로 결정할 수 있다. 활동 스레드의 개수가 많아질수록 τ는 작아지게 된다.

[0094] 현재 개발된 GPU 장치가 동적으로 스레드를 할당할 수 없기 때문에, 필요로하는 GPU 스레드의 개수를 사전에 결정해야 한다. L에서 $m_p \le M_i \le m_{p+1}$와 같은 M_i의 위치를 찾는 이진 검색(binary search)을 이용하여 충돌확인 대상물 O_i에 대한 파티션 개수를 먼저 산출한다. O_i에 대한 파티션 개수는 $\left\lceil \frac{p-i}{\tau} \right\rceil$ 값이다. 여기서 p-i는 잠재적 충돌 쌍의 개수이다. 최종적으로 모든 충돌확인 대상물에 대한 전체 파티션 개수가 할당에 필요한 전체 GPU 스레드 개수가 된다.

[0095] **스윕 방향 결정**

[0096] 전술한 알고리즘은 스윕 방향(sweep direction)을 3개의 축 중 하나의 방향이라고 가정한 것이다. 그러나 SaP 알고리즘에서 이러한 가정은 효율적이지 않은 결과를 가져 올 수 있다. 예컨대, 도 4에 도시된 예에서 x 축 방향을 스윕방향으로 선택하게 되면 불필요하게 많은 횟수의 충돌 테스트를 수행하게 된다.

[0097] 최적의 스윕 방향은 가능한 많은 데이터가 나누어지도록 투사되는 하나의 방향이 된다. 충돌확인 대상물이 투사된 후 충돌확인 대상물의 변화량을 최대화하는 스윕 방향을 찾기 위해 주성분분석법(PCA: Principal Component Analysis)이용할 수 있다.

[0098] 주성분 분석은 데이터를 한개의 축으로 사상시켰을 때 그 분산이 가장 커지는 축이 첫 번째 좌표축으로 선택

되고, 두 번째로 커지는 축이 두 번째로 선택되는 방식으로 차례로 놓이도록 새로운 좌표계로 데이터를 선형 변환한다. 이와 같이 각각의 축에 데이터의 "가장 중요한" 성분을 위치시킴으로써 여러 가지 응용이 가능하다. 주성분은 아이겐 벡터로 설명되나, 다른 선형 변환과 달리 사전에 정해진 아이겐 벡터를 갖지 않으며, 아이겐 벡터는 데이터의 특성에 따라 달라진다.

[0099] PCA를 이용한 스윕 방향 결정에서 스윕 방향은 첫 번째 주요 성분의 방향과 일치한다. 데이터 집합 X의 첫 번째 주요 성분의 방향 w_1은 아래의 수학식 1과 같이 정의된다. 이때 데이터의 MSE(Mean Squqare Error)를 최소화하기 위하여 평균 0(mean of zero) 값으로 평균 차감을 수행한다.

수학식 1

[0100]
$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} Var\{\mathbf{w}^T\mathbf{X}\} = \arg\max_{\|\mathbf{w}\|=1} E\{\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\}$$

[0101] 여기서, w_1는 공분산 행렬(covariance matrix) $C = XX^T$의 가장 큰 아이겐 값에 해당하는 아이겐 벡터이다.

[0102] PCA는 GPU를 이용하여 구현된 바 있으나, 이 알고리즘은 인터랙티브 응용프로그램에서 매우 느리다는 단점이 있다. 이를 극복하기 위하여 본 발명에서는 (a) 감소 병렬 스캔 알고리즘(reduced parallel scan algotithm)을 이용하여 X의 평균값을 계산한다. 이를 통해 아이겐 벡터 $C = XX^T$ 연산이 단순해진다. X는 3×n 크기이고, C는 3×3 대칭행렬이 된다. (b) 최종적으로 대칭 행렬 C의 아이겐 값을 연산하기 위하여 Jacobi 방법을 사용한다. 이를 통해 스윕 방향 결정을 인터랙티브 프로그램에서도 빠르게 수행할 수 있다.

[0103] **움직임 일관성(Motion Coherence)**

[0104] 종래 SaP 알고리즘의 종래 제3 단계는 교환 동작을 통해 대체될 수 있음은 전술한 바와 같다.

[0105] 종래 SaP 알고리즘에서의 교환 동작(swapping)은 움직임 일관성을 이용한다. 특히 작은 개수의 충돌확인 대상물이 움직이는 경우 매우 효과적이다. 그러나 본 발명의 알고리즘에서 종래의 교환 동작과 같은 과정은 GPU 상에서 수행되는 연산보다 더욱 많은 비용이 소모되는 메모리 접근이 필요하기 때문에 적합하지 않다.

[0106] 따라서 본 발명에서도 종래 제3 단계를 대신하여 수행가능한 GPU-맞춤 기술을 제안하고자 한다. 즉 움직임 일관성을 이용하면 전술한 S10 단계 내지 S40 단계와는 다른 충돌 감지 방법이 도출된다. 도 5는 움직임 일관성에 설명된 단계를 포함하는 또 다른 충돌 감지 방법이다.

[0107] 움직임 일관성은 시간 t에서 충돌확인 대상물 집합 $O = \{O_i\}$을 서로 차집합인 2개의 시간 의존적 부분집합으로 분류하여 활용된다. 2 개의 부분집합은 움직이는 충돌확인 대상물에 대한 집합 $O_m(t)$과 정적인 충돌확인 대상물에 대한 집합 $O_s(t)$이다. 따라서 $O = O_m(t) \cap O_s(t)$이다.

[0108] 본 발명은 $O_m(t)$ 및 $O_s(t)$에 대해 각각 충돌 감지를 수행하여 시간 의존적 충돌 쌍 $P(t)$을 검출한다. 이와 같은 방법은 $O_s(t)$의 충돌 쌍이 $O_s(t-1)$의 충돌 쌍과 큰 차이가 없기 때문에 $|O_m(t)| \ll |O_s(t)|$인 경우에 효과적이다. 따라서 $P(t-1)$로부터 $P(t)$에 변화는 약간의 업데이트만 필요하다.

[0109] 구체적으로 충돌 쌍 $P(t-1)$로부터 $P(t)$의 업데이트는 다음과 같은 단계를 통해 수행된다.

[0110] S100 단계: 기준 방향에 대해 각 충돌확인 대상물 O_i를 투사하여 $\{m_i, M_i\}$집합 형태로 표현되는 간격 I_i가 산출된다.

[0111] S200 단계: 모든 O_i에 대해 상기 S10 단계에서 산출된 m_i을 기준으로 기수 정렬(radix sort)을 수행하여 정렬

된 리스트 L이 산출된다.

[0112] S300 단계: $O_m(t)$에서 움직이는 충돌확인 대상물에 의해 발생하는 충돌 쌍 $P_M(t)$를 다음과 같은 소단계를 통해 산출한다.

[0113] (a) $O_s(t)$에 속하는 모든 O_i에 대하여, 병렬적 스위핑을 통해 정적인 충돌확인 대상물 집합 $O_s(t)$과 움직이는 충돌확인 대상물 집합 $O_m(t)$간에 충돌 쌍의 집합 $P_{sm}(t)$을 검출한다. 즉, $P_{sm}(t) = \{(O_i, O_j) \mid O_i \cap O_j \neq \emptyset \wedge O_i \in O_s(t), O_j \in O_m(t) \wedge i<j\}$이다.

[0114] (b) $O_m(t)$에 속하는 모든 O_i에 대하여, 병렬적 스위핑을 통해 움직이는 충돌확인 대상물 집합 $O_m(t)$과 모든 다른 충돌확인 대상물 집합 O를 비교하여 충돌 쌍의 집합 $P_{m*}(t)$을 검출한다. 즉, $P_{m*}(t) = \{(O_i, O_j) \mid O_i \cap O_j \neq \emptyset \wedge O_i \in O_m(t), O_j \in O \wedge i<j\}$이다.

[0115] (c) $P_M(t) = P_{sm}(t) \cap P_{m*}(t)$로 산출된다.

[0116] S400 단계: 이전 시간의 충돌 쌍$P(t-1)$ 후에 정적인 충돌확인 대상물 집합 $O_s(t)$ 내에서 간섭 쌍(interfering pair) $P_s(t)$을 검출한다. 이 과정은 $P_s(t) = P(t-1) - P'_M(t-1)$로 표현되며, 여기서, 각 쌍에서 하나의 충돌확인 대상물이 t 시간에 있는 $O_m(t)$일지라도 $P'_M(t-1)$는 t-1 시간에 충돌 쌍의 집합 $P(t-1)$이다. 즉, $P'_M(t-1) \equiv \{\forall(O_i, O_j) \in P(t-1) \mid O_i \in O_m(t) \vee O_j \in O_m(t)\}$이다.

[0117] S500 단계: 최종적으로 $P(t) = P_M(t) \cap P_s(t)$인 충돌 쌍($P(t)$)을 산출한다.

[0118] SaP 알고리즘을 매우 많은 수의 충돌확인 대상물에 적용하면 스윕 방향에서 투사된 간격 간에 많은 겹침이 발생할 수 있다. 이 문제는 전술한 PCA 기술을 이용하여 최적의 스윕 방향을 결정한 경우에도 나타날 수 있다. 이 문제를 해결하기 위하여 작업영역 분할(workspace subdivision)과 셀 분할(cell subdivision)을 병합한 하이브리드 SaP 기법을 제안하고자 한다.

[0119] **작업영역 분할(Workspace Subdivision)**

[0120] 스윕 축 상에 투사되는 충돌확인 대상물 간격의 밀집도를 낮추기 위해, SaP 알고리즘이 수행되기 전에 첫 번째 분할인 작업영역 분할이 수행된다. 이 분할은 GPU 상에서 알고리즘이 효율적으로 수행되도록 만들기 위한 균일 분할(uniform subdivision)이다.

[0121] 도 6(a)는 2 × 2 작업영역 분할을 나타내는 그래프이고, 도 6(b)는 SaP를 수행하기 위하여 셀이 시프팅하는 과정에 대한 일 예를 도시한 그래프이다.

[0122] SaP 알고리즘을 위한 스윕 방향을 d라고 하면, 3차원 작업영역은 d에 평행한 평면에 의해 잘리는 m × m 그리드 셀로 분할될 수 있다. 도 6(a)는 x가 스윕 방향일 때 2 × 2 분할의 z 축 단면에 해당한다. 최초 모든 충돌확인 대상물을 2개의 부분집합에 배치하였다. 2개의 부분 집합은 $O_{in} = \{O_i \in O \mid O_i$는 어떤 C_j 내부에 모두 존재함$\}$ 및 $O_{bd} \equiv O - O_{in}$ 이다. 나중에 O_{bd}는 확장된다.

[0123] 실제 실험적으로 m 값은 $m = \lceil \frac{n}{64K} \rceil$로 결정되어 사용되었다. 여기서, n은 충돌확인 대상물의 개수이다

[0124] 어떤 셀의 경계를 지나는 충돌확인 대상물(O_{bd})가 없는 경우(즉, $O_{bd} = \emptyset$), 그리드 셀의 행이나 열을 지나는 평행선에서 본 발명의 SaP를 수행할 수 있다. 이를 위해서 도 6(a)에 도시된 바와 같이, 각 셀이 식별할 수 있는 인덱스(도 6(a)에서는 C_1 및 C_2로 표시됨)를 갖는다. 도 5(a)를 살펴보면, $\{O_1, O_3, O_4\} \subset C_1$, $\{O_2, O_3, O_5, O_6\} \subset C_2$, $\{O_2, O_3, O_5\} \subset Cr_1$. $\{O_1, O_2, O_4, O_5, O_6\} \subset O_{in}$, $\{O_2, O_3, O_5\} \subset O_{bd}$인 것을 알 수 있다.

[0125] 이 후 $O_i \in O_{in}$ 인 각 충돌확인 대상물에 의해 투사되는 간격 I_i는 d 방향인 $(j-1)\times l$ 을 이용해 변경될 수 있

다. 도 6(b)에 도시된 바와 같이, 여기서, O_i가 속한 셀이 C_j가 되고, l은 d 방향에 따른 작업영역의 크기를 의미한다. 이러한 변경을 시프팅(shifting) 기술이라 하고, 시프팅 기술은 스윕 방향에서 충돌확인 대상물이 겹쳐지는 개수를 줄이고, SaP 알고리즘의 효과를 높인다. 도 6(b)를 살펴보면, $\{O_1, O_4\}$ 및 $\{O_2, O_5, O_6\}$가 각각 C_1및 C_2과 시프트되었는데, 이는 $\{O_1, O_4\} = C_1 \cap O_{in}$ 이고 $\{O_2, O_5, O_6\} = C_2 \cap O_{in}$ 이기 때문이다. $\{O_2, O_3, O_5\} = Cr_1 \cap O_{bd}$ 이기 때문에 $\{O_2, O_3, O_5\}$는 Cr_1와 시프트되었다.

[0126] 실제 어떤 충돌확인 대상물은 셀 경계를 지나기도 한다($O_{bd} \neq \emptyset$). 잠재적으로 O_{bd}에 있는 충돌확인 대상물과 충돌할 가능성 있는 충돌확인 대상물을 검출할 수 있다. 이러한 충돌확인 대상물은 경계에 의해 영향을 받는다고 할 수 있다.

[0127] 이러한 충돌확인 대상물을 검출하기 위하여 O_{bd}에 있는 충돌확인 대상물의 최대 크기를 계산해야 한다. 예컨대, 구의 최대 크기는 구의 지름이고 AABB의 최대 크기는 AABB의 가장 긴 3차원 길이에 해당한다.

[0128] O_{bd}에 있는 충돌확인 대상물의 최대 크기를 계산한 후 O_{bd}에 영향을 받는 영역 $R = \{C_{R1}, \cdots, C_{Rn}\}$을 추정할 수 있다. 예컨대, 도 6에서는 R이 C_{r1}이다. 그리고 R과 겹치는 모든 충돌확인 대상물을 검출할 수 있다. R과 겹치는 충돌확인 대상물은 O_{bd}에 추가한다. 각 C_{ri}에 새로운 셀 인덱스를 부여하여 새로운 셀로 취급하고, O_{bd}에 있는 각 충돌확인 대상물을 각 충돌확인 대상물에 겹치는 C_{ri}셀에 연관시킨다. 예컨대, 도 6을 살펴보면, 초기 $O_{bd} = \{O_3\}$이지만 C_{r1}때문에 $\{O_2, O_3, O_5\}$으로 확장된다.

[0129] 어떤 충돌확인 대상물은 도 6에서의 O_2 및 O_5와 같이, O_{in}과 O_{bd}에 동시에 속하여 R에 있는 하나 이상의 셀에 영향을 받을 수도 있다.

[0130] **셀 분할(Cell subdivision)**

[0131] 작업영역 분할에서 서로 다른 셀에 속한 충돌확인 대상물들은 서로 충돌할 수 없기 때문에, 서로 다른 셀에 속한 충돌확인 대상물들은 충돌 테스트를 수행할 필요가 없다. 따라서 병렬 SaP의 연산 오버헤드를 크게 줄일 수 있다. 그러나 어떤 셀에서는 연관된 충돌확인 대상물이 실제로 교차하지 않지만 투사된 간격이 겹치는 충돌확인 대상물을 많이 포함할 수 있다. 이와 같은 잘못된 판단(false positive)을 줄이기 위하여, 두 번째 분할인 셀 내(intra-cell) 분할을 SaP 동작 중에 수행하는 것이 바람직하다.

[0132] 도 7은 본 발명에 따른 셀 분할의 일 예를 도시한 그래프이다. 각 셀은 4×4 서브셀로 나누어진 것이고, 우측에 확대 도시된 것은 좌측은 C_1 영역의 단면도이다.

[0133] 본 발명의 셀 분할은 스윕 방향에 평행인 작업영역 분할의 셀을 나누기 때문에 2차원 분할에 해당한다. 이하 서브셀에 대한 새로운 맵핑(mapping) 방법을 소개하고자 한다. m-비트 주소를 사용하는데, m은 서브셀 전체 개수의 로그(log) 값이고, 도 7에 도시된 바와 같이, 상기 비트 값의 절반은 각 차원(dimension)에 대응된다.

[0134] 이와 같은 셀 분할은 같은 서브셀을 공유하는 충돌확인 대상물에 대해서만 SaP가 수행되도록 한다. 도 7은 16개의 서브셀로 하나의 셀이 분할된 예로서, O_1 과 O_4 에 있는 충돌확인 대상물 간의 충돌 테스트는 수행되지 않는 예를 보여준다. 어떤 서브셀이 어떤 충돌확인 대상물을 포함하고 있는지 검출할 수 있다.

[0135] 서브셀의 수는 GPU 장치에서 충분히 수행가능한 개수로 나누는 것이 바람직하다.

[0136] 도 7에서 O_4의 극단점은 1010 및 1011로 맵핑된다. 같은 서브셀을 공유하고 있는지 여부는 비트별로 AND 연산을 수행하면 확인이 가능하다. 최종적으로 각 O_i에 대해 병렬수행되는 SaP는 같은 서브셀을 갖는 충돌확인 대상물에 대해서만 수행되면 된다. 이를 통해 후보 집합(candidate set)을 크게 줄일 수 있어 알고리즘의 효율이 높아진다.

[0137] **효과를 입증하는 실험 및 데이터**

[0138] 이하 전술한 본 발명의 광위상(broad-phase) 충돌감지 알고리즘의 수행 효과를 증명하는 실험결과를 설명하고자 한다.

[0139] 본 실험에서는 Visual Studio C++ 및 NVIDIA CUDA 프로그래밍 언어가 사용되었고, 실험에 사용된 컴퓨터는 Intel Quad-core 2.66GHZ CPU 및 2.8GB 메인 메모리를 갖고, 4Gb 메모리를 갖는 NVDIA Tesla C1060 그래픽 카드를 장착한 것이었다.

[0140] 초기 충돌확인 대상물 및 최종 충돌 결과는 GPU의 광역 메모리에 저장하였다. 실험은 다양한 형태를 갖는 충돌확인 대상물을 사용하였다. 랜덤한 형상(Random Configuration), 파티클 시뮬레이션(Particle Simulation) 및 강체 움직임(Rigid-body Dynamics)에 대하여 본 발명의 방법과 CPU 또는 GPU를 기반으로 하는 다른 알고리즘의 성능을 비교하였다.

[0141] 이중 랜덤한 형상을 대상으로 한 실험을 대표적으로 설명한다. 밴치마킹을 위한 대상물은 Bullet collision library와 유사한 것을 사용하였다. AABB가 불균형하게 배치되고 랜덤하게 이동하는 대상물을 사용하였고, AABB의 개수는 16K, 128K 및 960K 3가지로 설정하였다. 본 발명의 방법과 BoxPruning, ArraySap 및 AABB dynamic tree 방법을 비교하였다. 도 8에 도시된 바와 같이 본 발명의 방법이 종래의 다른 방법에 비해 월등히 빠른 속도 결과를 나타내었다.

[0142] 또한 일부 충돌확인 대상물이 움직이는 경우에 본 발명의 방법의 수행 성능을 실험하였다. 대상물은 서로 다른 크기를 갖는 1백만 개의 AABB이고, 전체에서 5%에서 25%까지 이동하는 것으로 설정하였다.

[0143] 도 9는 충돌확인 대상물 중 일부가 움직이는 경우 본 발명의 방법에 따른 충돌 감지 수행 결과를 도시한 것이다. 'CD Pairs'는 움직이는 대상물로 인해 발생한 충돌의 비율을 의미하고, 'CD Timing'는 움직이는 모든 대상물에 대한 연산 시간을 의미한다.

[0144] 도 9에 도시된 바와 같이, 움직이는 대상물로 인해 새로운 충돌이 발생하지만 이를 위한 연산 시간도 선형적으로만 증가하였음을 알 수 있다. 이러한 결과는 본 발명의 방법이 정적인 대상물에 의해 유발되는 충돌 결과를 효율적으로 활용한다는 것을 의미한다.

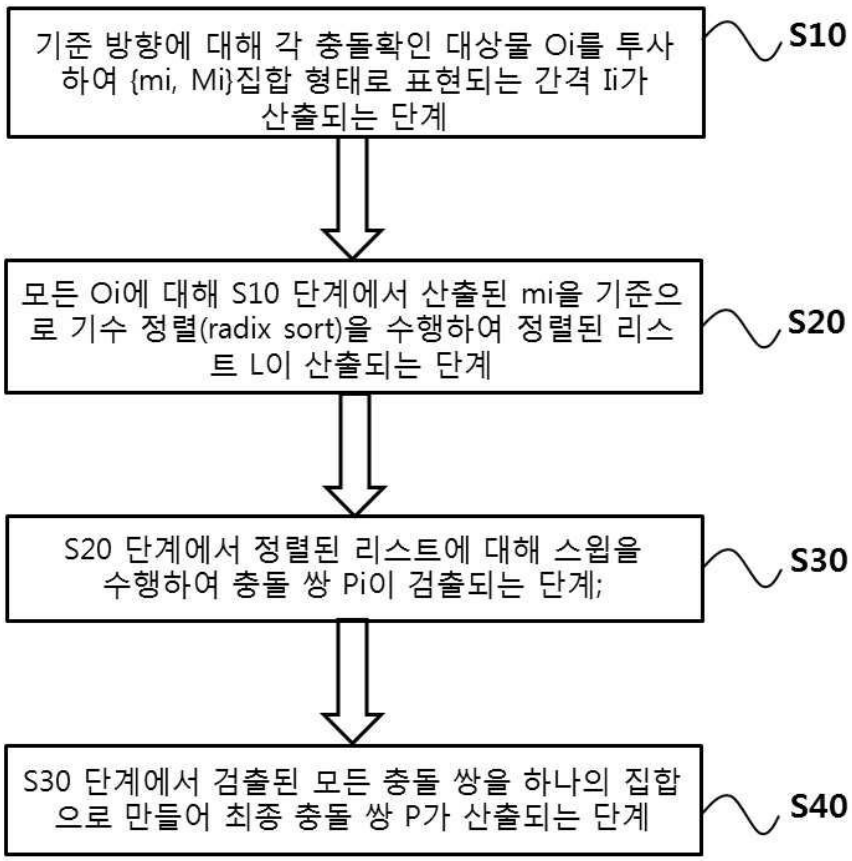
[0145] 도 10은 본 발명의 최적 스윕 방향(Best Axis)을 사용한 수행 결과와 x축을 일괄적으로 스윕 방향으로 설정한 경우를 비교한 막대 그래프이다. AABB 사이를 움직이는 구를 대상으로 실험을 수행하였는데 본 발명의 PCA를 사용한 방법이 3배 이상 빠른 것을 확인할 수 있다.

[0146] 도 11은 본 발명의 스레드 할당을 적용한 경우와 적용하지 않은 경우의 충돌 감지 알고리즘 시간을 비교한 도표이다. 시간 단위는 ms(밀리 세컨드)이고, 좌측 64K, 128K, 256K는 대상물의 개수이다. 스레드 할당을 통한 알고리즘이 2배 이상 성능이 좋은 것을 확인할 수 있다.

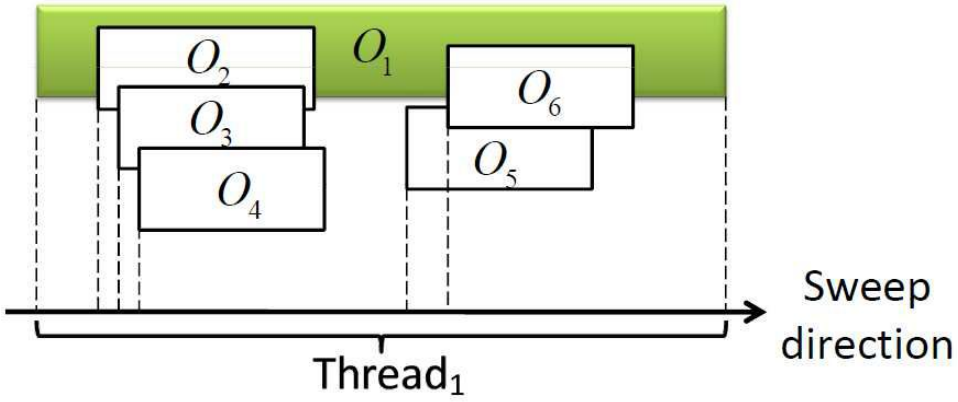
[0147] 본 실시예 및 본 명세서에 첨부된 도면은 본 발명에 포함되는 기술적 사상의 일부를 명확하게 나타내고 있는 것에 불과하며, 본 발명의 명세서 및 도면에 포함된 기술적 사상의 범위 내에서 당업자가 용이하게 유추할 수 있는 변형 예와 구체적인 실시예는 모두 본 발명의 권리범위에 포함되는 것이 자명하다고 할 것이다.

도면

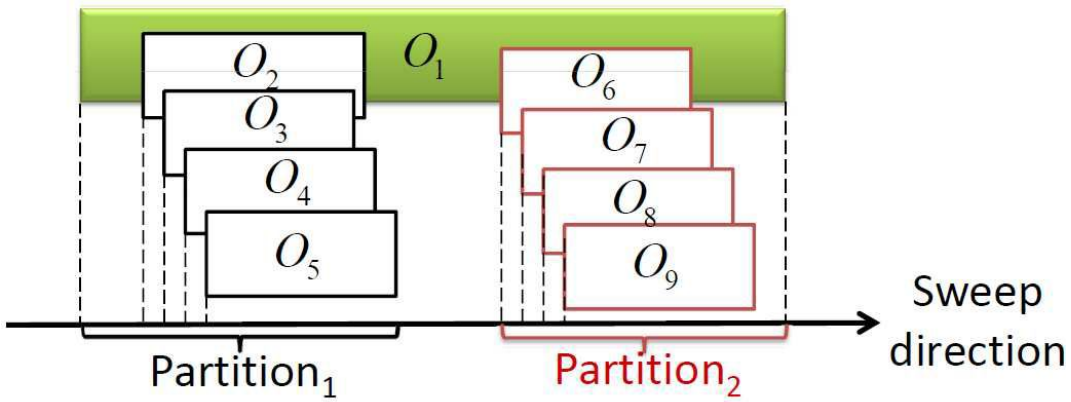
도면1


기준 방향에 대해 각 충돌확인 대상물 Oi를 투사하여 {mi, Mi}집합 형태로 표현되는 간격 Ii가 산출되는 단계
S10
모든 Oi에 대해 S10 단계에서 산출된 mi을 기준으로 기수 정렬(radix sort)을 수행하여 정렬된 리스트 L이 산출되는 단계
S20
S20 단계에서 정렬된 리스트에 대해 스윕을 수행하여 충돌 쌍 Pi이 검출되는 단계;
S30
S30 단계에서 검출된 모든 충돌 쌍을 하나의 집합으로 만들어 최종 충돌 쌍 P가 산출되는 단계
S40

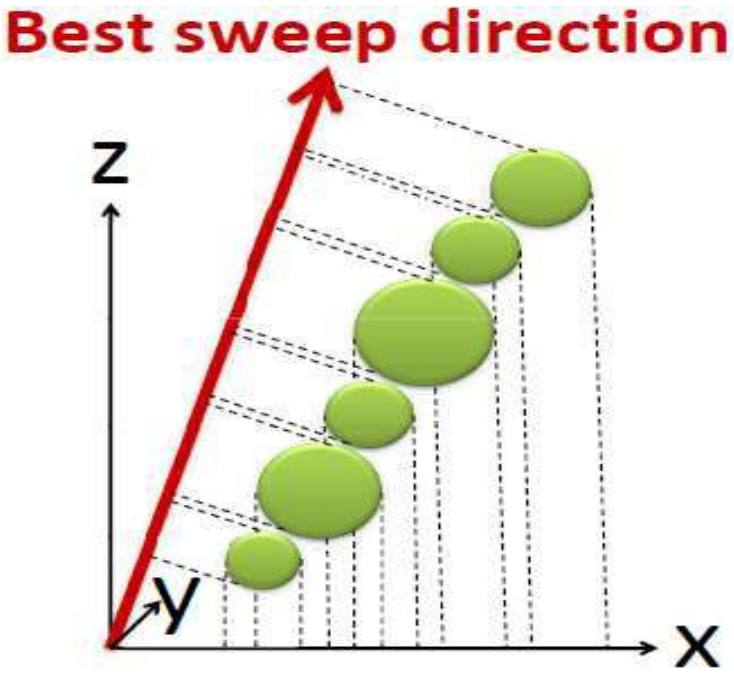
도면2


O_1
O_2
O_3
O_4
O_5
O_6
Sweep direction
Thread$_1$

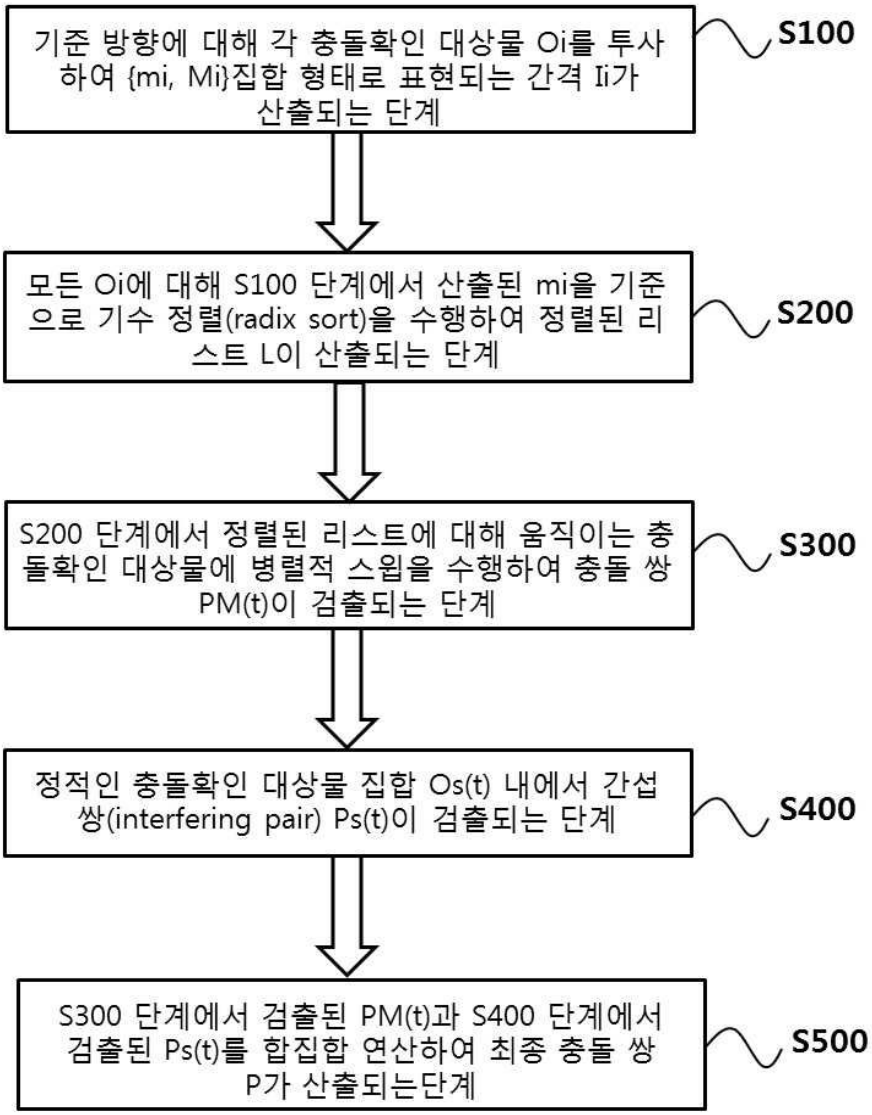
도면3


O_1
O_2
O_3
O_4
O_5
O_6
O_7
O_8
O_9
Sweep direction
Partition$_1$
Partition$_2$

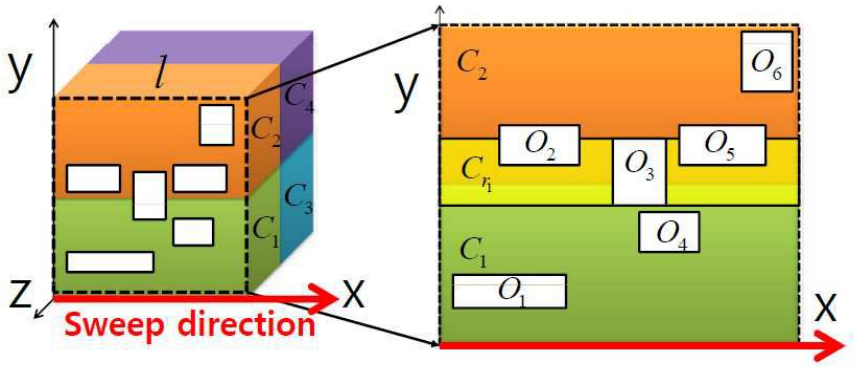
도면4



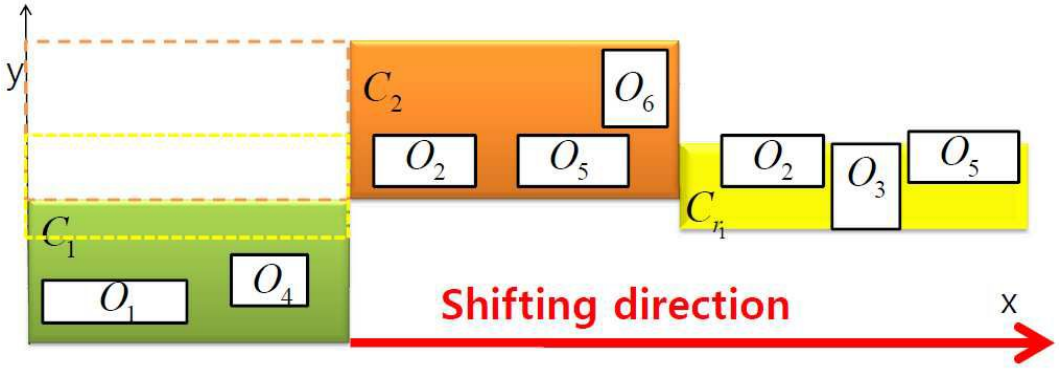
도면5


기준 방향에 대해 각 충돌확인 대상물 Oi를 투사하여 {mi, Mi}집합 형태로 표현되는 간격 Ii가 산출되는 단계
S100
모든 Oi에 대해 S100 단계에서 산출된 mi을 기준으로 기수 정렬(radix sort)을 수행하여 정렬된 리스트 L이 산출되는 단계
S200
S200 단계에서 정렬된 리스트에 대해 움직이는 충돌확인 대상물에 병렬적 스윕을 수행하여 충돌 쌍 PM(t)이 검출되는 단계
S300
정적인 충돌확인 대상물 집합 Os(t) 내에서 간섭쌍(interfering pair) Ps(t)이 검출되는 단계
S400
S300 단계에서 검출된 PM(t)과 S400 단계에서 검출된 Ps(t)를 합집합 연산하여 최종 충돌 쌍 P가 산출되는단계
S500

도면6

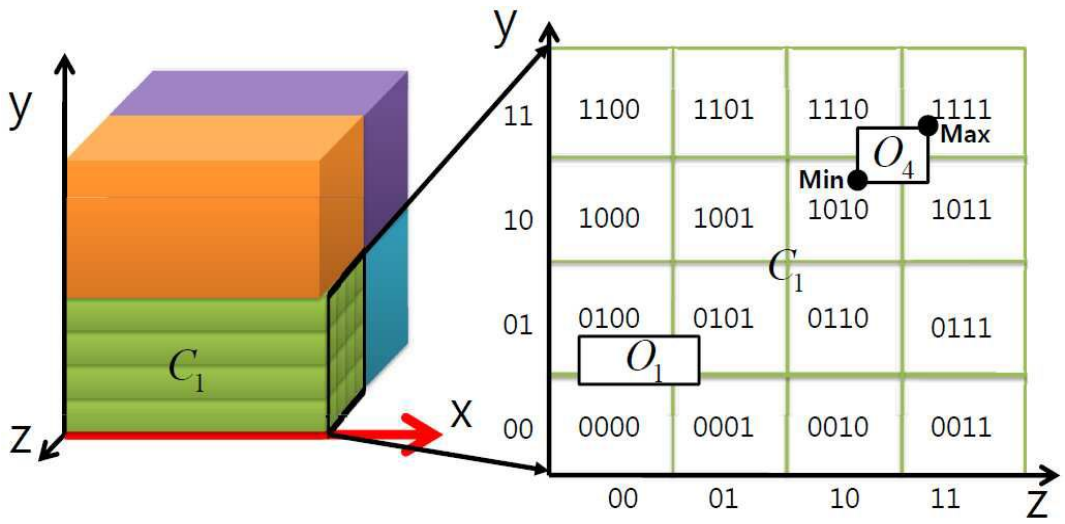

y
l
C4
C2
C3
C1
z
x
Sweep direction
y
C2
O6
O2
O5
Cri
O3
O4
C1
O1
x

(a)

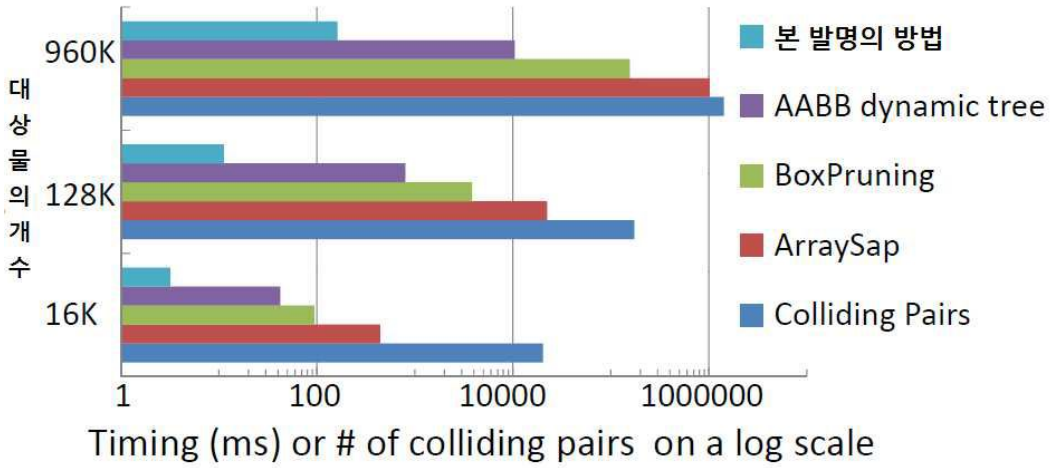

y
C2
O6
O2
O5
O2
O3
O5
Cri
C1
O1
O4
Shifting direction
x

(b)

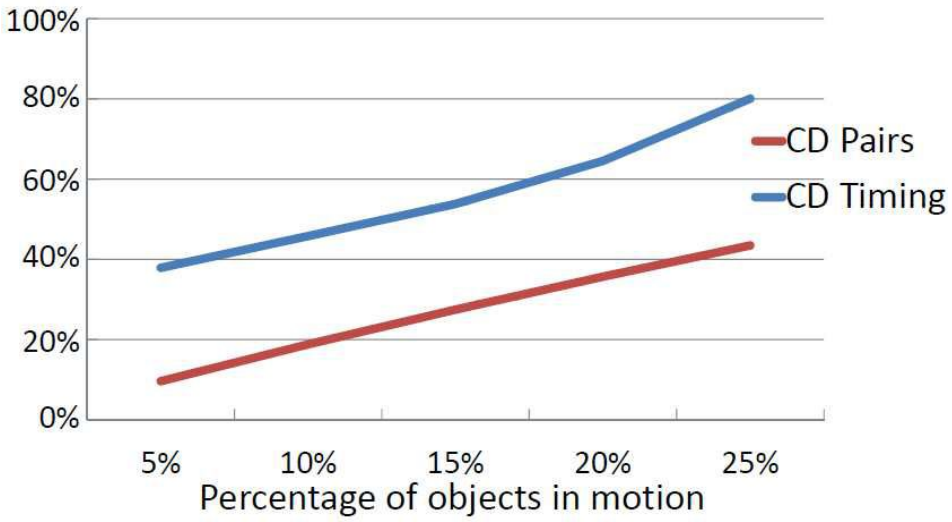
도면7


y
z
x
C_1
y
11
10
01
00
1100
1101
1110
1111
Max
O_4
Min
1000
1001
1010
1011
C_1
0100
0101
0110
0111
O_1
0000
0001
0010
0011
00
01
10
11
z

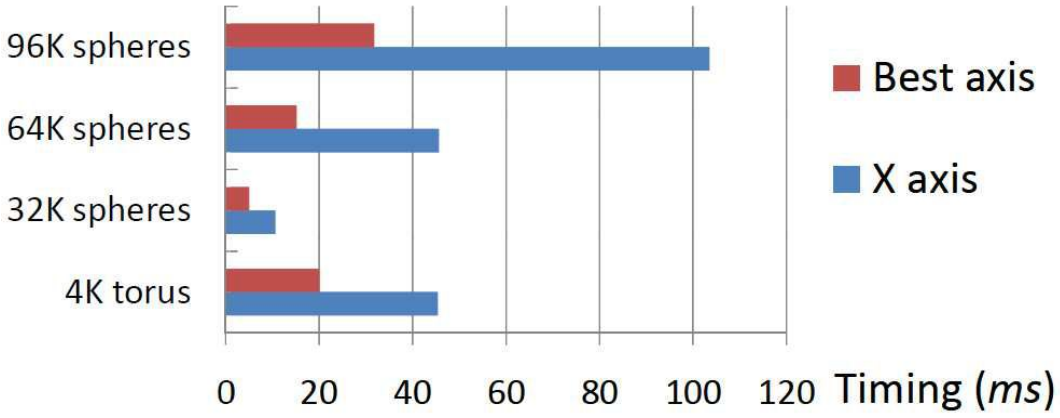
도면8


960K
128K
16K
대상물의개수
1
100
10000
1000000
Timing (ms) or # of colliding pairs on a log scale
본 발명의 방법
AABB dynamic tree
BoxPruning
ArraySap
Colliding Pairs

도면9


100%
80%
60%
40%
20%
0%
5%
10%
15%
20%
25%
Percentage of objects in motion
CD Pairs
CD Timing

도면10


96K spheres
64K spheres
32K spheres
4K torus
Best axis
X axis
0
20
40
60
80
100
120
Timing (ms)

도면11

스레드 할당	적용	미적용
64K	13.71	30.05
128K	48.16	124.11
256K	241.51	467.78