한국컴퓨터정보학회 동계학술대회 논문집 제27권 제1호 (2019. 1)

입자 기반 시스템에서 동적인 부채꼴 영역을 이용한 2차원 충돌 검사의 가속화 기법

김종현O
O강남대학교 소프트웨어응용학부
e-mail: jonghyunkim@kangnam.ac.krO

Acceleration Method of 2D Collision Detection with Dynamic Cone Area in Particle-based System

Jong-Hyun KimO
ODept. of Software Application, Kangnam University

● 요 약 ●

본 논문에서는 많은 개체와의 충돌검사를 요구하는 입자 기반 시스템에서 부채꼴 영역의 동적인 변화를 이용하여 효율적으로 충돌검사를 가속화시킬 수 있는 프레임워크를 제안한다. 부채꼴 영역의 동적인 변화를 계산하기 위해 입자의 위치와 속도를 이용하여 부채꼴의 영역을 결정하였으며, 이 영역 내에 있는 입자들만을 이용하여 충돌 검사를 빠르게 수행한다. 본 연구에서 제안하는 가속화 방법은 트리 자료구조를 명시적으로 만들지 않고, 닫힌 형태 방정식(Closed form equation)으로 실행되기 때문에 간단하게 구현되며 모든 결과에서 충돌검사 성능이 3배 정도 개선되었다.

키워드: 충돌 검사(Collision detection), 동적인 부채꼴 영역(Dynamic cone area), 입자 기반 시스템(Particle-based system), 가속화 기법(Acceleration method)

I. Introduction

입자 기반 시스템은 물리 시뮬레이션, 렌더링, 게임, 영상특수효과 등 다양한 분야에서 사용되고 있다[1,2]. 일반적인 고체와는 다르게 입자 기반 시스템은 충돌 검사하는 개체의 수가 굉장히 많기 때문에 이를 효율적으로 다루고자 팔진/쿼드 트리[3], K-d 트리[4], 해시 테이블[5] 등 다양한 최적화 방법들이 사용되고 있다 (Fig. 1 참조).

하지만, 이러한 가속화 자료구조는 입자의 위치가 변경되면 매번 자료구조를 업데이트해야 되기 때문에 메모리와 계산속도 측면에서 비효율적이다. 이 문제는 실시간을 요구하는 게임이나 가상현실 콘텐츠 분야에서 적합하지 않고, 충돌 검사할 개체의 수가 적다면 오히려 앞에서 언급한 가속화 자료구조를 생성하는 과정에서 계산 속도가 더 오래 걸리는 상황이 발생하기도 한다.

Fig. 1. Examples of particle-based system.

본 논문에서는 이러한 문제를 해결위해 입자의 위치와 속도를 기반으로 부채꼴의 영역을 동적으로 변경하고 이를 이용하여 충돌 검사를 가속화하는 새로운 자료구조를 제안한다.

II. The Proposed Scheme

1. Collision Detection Between Particle and Static Cone Area

제안하는 방법은 두 가지 단계로 구분되어 실행된다. 첫 번째로 정적인 부채꼴 영역과 입자와의 충돌 검사이며, 두 번째로 입자의 위치와 속도를 고려하여 부채꼴의 영역을 업데이트한다.

첫 번째 조건을 모델링하기 위해 본 연구에서는 입자를 구와 같은 형태라고 가정하고, 아래와 같이 세 가지 종류의 충돌 검사를 수행한다 (Fig. 2 참조).

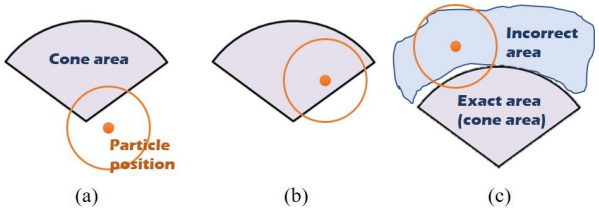


Fig. 2. Various types of collision detection.

1) 부채꼴 영역의 중심 위치가 입자의 내부에 있을 경우 (Fig. 2a 참조)

이 같은 경우는 구의 방정식을 활용하여 쉽게 계산 할 수 있다 (수식 1 참조).

$$(x_p - x_c)^2 + (y_p - y_c)^2 < r_p^2 \quad (1)$$

여기서 (x_p, y_p)와 (x_c, y_c)는 각각 입자와 부채꼴의 중심 좌표이며, r_p는 입자의 반지름이다. 여기서 반지름은 입자와 부채꼴 영역간의 충돌 검사를 수행할 때 반경이며, 결과적으로 부채꼴 영역의 중심이 입자의 반경 내부에 존재하면 수식1을 만족하기 때문에 충돌이 발생된 상황으로 판단한다.

두 번째와 세 번째 조건은 부채꼴 영역을 구성하는 두 벡터의 내분을 활용하여 계산한다.

2) 입자의 위치가 부채꼴 영역의 내부에 있을 경우 (Fig. 2b 참조)
3) 입자의 위치가 부채꼴 영역을 형성하는 두 벡터 사이에 있고, 입자의 위치로부터 부채꼴 중심까지의 거리가 "입자의 반경+부채꼴 영역의 반지름"보다 작다면 충돌 (Fig. 2c 참조)

입자가 부채꼴 영역 내에 있다면, 이 가정은 부채꼴 영역을 구성하는 두 벡터 v_1과 v_2 사이에 있다고 말할 수 있고, 본 연구에서는 이러한 조건을 아래와 같이 계산한다 (수식 2 참조).

$$P - C = \alpha v_1 + \beta v_2 \quad (2)$$

여기서 P는 입자의 위치 (x_p, y_p)이고, C는 부채꼴의 중심위치인 (x_c, y_c)이다. α와 β는 0보다 큰 가중치이고 벡터의 내분을 계산할 때 사용되는 값이다. 수식 2는 벡터의 내분을 활용한 수식이며, 결과적으로 두 벡터 v_1과 v_2 사이를 $t : 1-t$ 비율로 구하는 수식이다. 본 연구에서 가중치인 $1-t$는 α이며 t는 β이다. 두 벡터 공간에 입자가 위치한다면 α와 β는 0보다 크거나 같고, 둘의 합은 1이 되며, 1보다 큰 경우 부채꼴 영역 외부에 있다고 판단한다. 하지만, 이 조건만 적용할 경우 Fig. 2c에서 보듯이 두 벡터를 포함하는 모든 공간에 대해서 충돌되었다고 잘못 판단된다. 우리가 원하는 부채꼴의 영역은 연한 분홍색 부분이지만, 결과적으로 하늘색 영역까지 포함된다 (Fig. 2c 참조). 본 연구에서는 이 문제를 피하기 위해 Fig. 2b조건을 추가한다. 부채꼴의 중심에서 입자의 중심으로의 방향 벡터의 크기를 비교하여 이 길이가 v_1의 크기보다 작은 경우에만 충돌영역이라고 판단한다 (수식 3 참조).

$$\| P - C \| < \| v_1 \| \quad (3)$$

본 논문에서 부채꼴 영역을 구성하는 v_1과 v_2는 같은 크기이기 때문에 위 수식에서 v_1이라고 했지, v_2로 계산해도 같은 결과가 나온다. 앞에서 언급한 세 가지 조건을 모두 만족하는 경우에 입자와 정적인 부채꼴 영역은 충돌한다.

2. Calculation of Dynamic Cone Area

본 논문에서는 부채꼴 영역에 동적인 변화를 추가하여 입자의 움직임에 따라 부채꼴 영역을 자동으로 업데이트시킨다. 이 방법을 구현하기 위해 입자의 위치와 속도를 이용하였으며, 이 속성들로부터 부채꼴의 위치, 각도, 길이를 제어한다 (Fig. 3 참조).

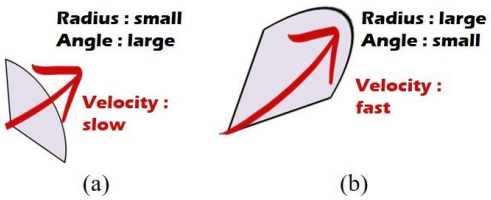


Fig. 3. Dynamic search area of cone.

부채꼴의 위치 C는 입자의 위치 P와 동일하게 하고 부채꼴의 반지름과 각도와 아래와 같이 계산한다.

$$r_c = \| v \| \delta \quad (4)$$

여기서 r_c와 v는 부채꼴의 반지름과 입자의 속도이며, δ는 반지름을 조절하는 가중치이다. 각도 a_c는 최소와 최대 각도를 두어 그 사이 값을 입자의 속도에 따라 선형 보간하여 업데이트 한다 (수식 6 참조).

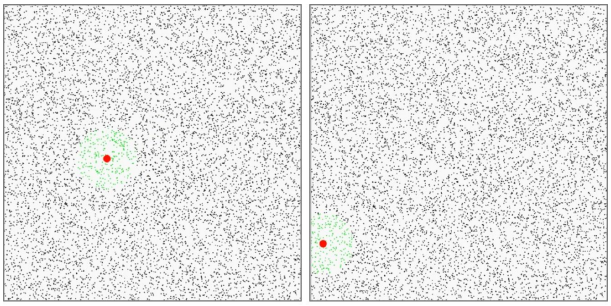
$$w = \frac{\min(\max(r_c, \| v^{\min} \|), \| v^{\max} \|)}{\| v^{\max} - v^{\min} \|} \quad (5)$$

$$a_c = (1-w)a^{\max} + wa^{\min} \quad (6)$$

III. Results

아래 그림은 2차원에서 다수의 입자를 배치하여 충돌 검사를 적용한 결과이다 (Fig. 4 참조). 빨간색 구를 중심으로 충돌된 녹색 입자를 찾는 과정에서 입자의 속도에 관계없이 전체 입자들에 대해 충돌 검사를 수행하기 때문에 불필요한 계산이 추가된다. 2차원에서 500,000개의 입자를 실험하는 과정은 빠르게 수행될 수 있기에 가속화

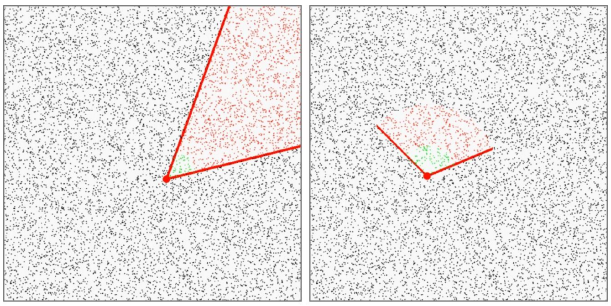
자료구조를 사용할 경우 오히려 쿼드 트리나 해싱 테이블을 구성하는 데 시간이 더 오래 걸린다. Fig. 4의 결과는 평균 0.003초가 소요됐다.



(a) Fast velocity (b) Slow velocity

Fig. 4. Collision detection with previous approach (green particle : collided particle).

본 연구의 방법은 Fig. 4와 동일한 환경에서 실험되었으며 본 논문에서 제안한 기법은 입자의 개수가 많지 않음에도 불구하고 0.001초가 소요되었으며, 평균 3배 정도의 성능이 향상되었다 (Fig. 5 참조). 그림에서 빨간색 입자들은 충돌 검사를 수행할 입자들이며 모든 입자들에 대해서 충돌 검사를 수행한 Fig. 4에 비해 그 개수가 월등히 줄어들었다. 뿐만 아니라 안정적으로 충돌 검사를 수행하기 위해 입자의 속도에 따라 영역이 자동으로 업데이트 되었으며, 2장에서 설명했듯이 부채꼴 영역이 다르게 나타나는 것을 쉽게 볼 수 있다 (Fig. 5 참조).



(a) Fast velocity (b) Slow velocity

Fig. 5. Collision detection with our method (red particle : candidate particle, green particle : collided particle).

IV. Conclusions

본 논문에서는 입자의 속도에 따라 부채꼴 영역을 변화시켜 충돌 검사에 필요한 입자들의 개수를 줄이는 새로운 프레임워크를 제안했다. 2차원 입자 기반 시스템에서 수행한 결과 충돌 검사 성능이 3배 개선되었으며, 향후 인접 입자들의 정보를 매번 필요로 하는 입자 기반 유체 시뮬레이션에 적용하여 시뮬레이션 성능을 고속화 할 수 있도록 연구할 계획이다.

REFERENCES

[1] Premžoe, Simon, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker. "Particle-based simulation of fluids." In Computer Graphics Forum, vol. 22, no. 3, pp. 401-410. Oxford, UK: Blackwell Publishing, Inc, 2003.

[2] Sakamoto, Naohisa, Jorji Nonaka, Koji Koyamada, and Satoshi Tanaka. "Particle-based volume rendering." In Visualization, 2007. APVIS'07. 2007 6th International Asia-Pacific Symposium on, pp. 129-132. IEEE, 2007.

[3] Lucchesi, Benjamin J. "A parallel linear octree collision detection algorithm." PhD diss., University of Nevada, Reno, 2002.

[4] Horn, Daniel Reiter, Jeremy Sugerman, Mike Houston, and Pat Hanrahan. "Interactive kd tree GPU raytracing." In Proceedings of the 2007 symposium on Interactive 3D graphics and games, pp. 167-174. ACM, 2007.

[5] Lefebvre, Sylvain, and Hugues Hoppe. "Perfect spatial hashing." In ACM Transactions on Graphics (TOG), vol. 25, no. 3, pp. 579-588. ACM, 2006.