검사 빈도에 따른 그리드 기반 충돌 검사에서의 동적 공간 분할 기법

이영민, 신병석 *
인하대학교 컴퓨터정보공학과
e-mail : ymlee3218@gmail.com, bsshin@inha.ac.kr

A Dynamic Space Partition Method on Grid-based Collision Detection under Detection Frequencies

Young-Min Lee, Byeong-Seok Shin
Dept. of Computer Science And Information Technology, Inha University

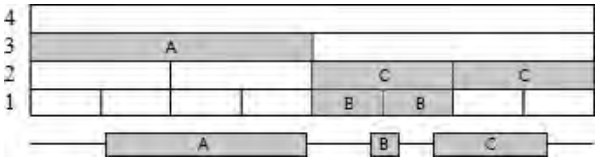
요 약

그리드 기반 공간 분할 방식에서 셀 크기는 성능을 좌우하는 핵심 요소이다. 본 논문에서는 각 셀에서 충돌 검사가 일어나는 빈도에 따라 동적으로 셀의 크기를 제어하는 방식을 제안한다. 각 셀의 충돌 검사 빈도와 인접한 셀과 병합했을 때의 충돌 검사 빈도를 비교하여, 이 차이가 임의의 값보다 작을 때 셀들을 병합하여 하나의 큰 셀로 만든다. 단, 병합할 셀이 여러 개일 경우 충돌 검사 빈도가 작은 셀부터 차례로 병합한다. 이 방식을 통하여 셀의 수 및 업데이트 비용을 줄일 수 있으며 이는 공간상의 객체가 규칙적인 움직임을 보일 때 가장 효율적임을 확인했다.

1. 서론

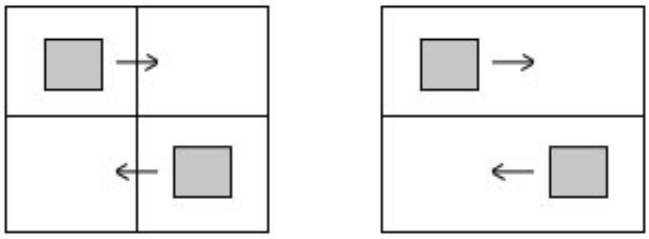
충돌 검사는 장면에서 움직이는 객체들의 상호관계를 확인하는 방법 중 하나이다. 대부분의 게임과 시뮬레이션에서 충돌 검사는 다른 물리 계산의 기본이 되는 필수적인 요소지만, 동시에 수행 시간의 대부분을 차지할 정도로 비용이 큰 부분이다. 이 비용을 줄이기 위해 충돌 가능성이 있는 객체들만 구분하여 검사하는 많은 방법들이 연구되었다. 이러한 방법들 중 하나인 그리드 기반 공간 분할 방식에서는 셀 크기를 얼마로 할지 정하는 것이 중요하다. 셀의 크기가 지나치게 크면 하나의 셀 안에 포함되는 객체가 늘어남에 따라 공간을 분할함으로써 얻는 이점이 줄어들고, 셀의 크기가 지나치게 작으면 메모리 낭비 및 객체의 이동 시 셀 업데이트에 많은 자원이 소비된다[1].

최적의 셀 크기를 정하기 위한 방법으로 계층적 그리드 기반 공간 분할 방식이 있다[2,3]. 이 방법은 객체를 기준으로 셀의 크기를 정하는 것으로서, 셀을 크기에 따라 여러 계층으로 나누어 객체의 크기에 가장 알맞은 레벨의 셀을 할당하는 방식이다.



(그림 1) 1 차원에서의 계층적 그리드 기반 공간 분할의 예시

그림 1 의 아래에는 각각 객체 A, B, C 가 있고, 그 위의 4 단계로 나누어진 계층적 그리드가 있다. 객체가 셀의 경계선에 위치해 있을 때엔 복수의 셀을 차지하게 되는데, 객체의 크기가 셀의 크기보다 작다면 최악의 경우에도 2 차원에서는 객체당 4 개의 셀이, 3 차원에서는 8 개가 매핑된다. 그러므로 각각의 객체가 할당 받게 될 셀은 객체의 크기보다 큰 크기의 셀 중 가장 작은 셀이 된다. 이는 효율적인 충돌 검사가 가능하다는 것을 의미한다.



(그림 2) 움직이는 객체가 포함된 공간을 각각 4 개의 셀과 2 개의 셀로 나눈 경우

하지만 이처럼 계층적 그리드 기반 공간 분할 방식은 객체가 차지하는 셀의 개수에 초점을 맞춘 방식이기 때문에, 그림 2 에서와 같이 해당 객체가 다른 객체가 존재하지 않는 셀로 넘나드는 경우에는 여전히 셀을 업데이트하기 위해 불필요한 비용이 발생한다.

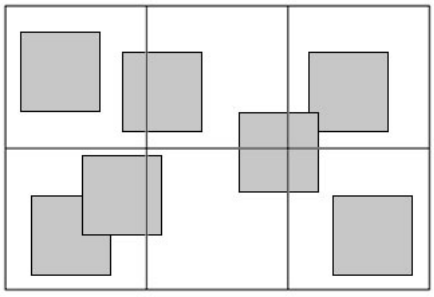
* 교신저자

이는 각 셀에서 충돌 검사가 발생하는 빈도를 고려하지 않았기 때문이다. 본 논문에서는 이처럼 셀에서 충돌 검사가 일어나는 빈도를 고려하여 동적으로 셀의 크기를 제어함으로써 효율적으로 공간을 분할하는 기법을 제안한다.

2. 관련 연구

일반적인 충돌 검사의 경우, 모든 객체들에 대해 두 개의 객체가 겹쳤는지를 확인하는 방식으로 검사를 수행한다. 하나의 공간 안에 있는 N 개의 객체가 있을 때 충돌 검사 횟수는 $O(N^2)$이 된다[4].

충돌 검사의 대상이 되는 객체를 최대한 줄이면 렌더링할 때 연산량을 획기적으로 줄일 수 있다. 이를 위해 충돌검사를 실제 충돌 여부를 검사하는 미세 충돌 검사 단계(narrow CD phase)와, 그 전에 충돌할 가능성이 높은 객체들만을 따로 판단하는 광역 충돌 검사 단계(broad CD phase)로 나누어 진행한다. 일반적으로 미세 충돌 검사 비용이 광역 충돌 검사 비용에 비해 크므로, 광역 충돌 검사를 통해 충돌 가능성이 없는 객체를 제외하는 것만으로도 처리속도를 높일 수 있다[5].

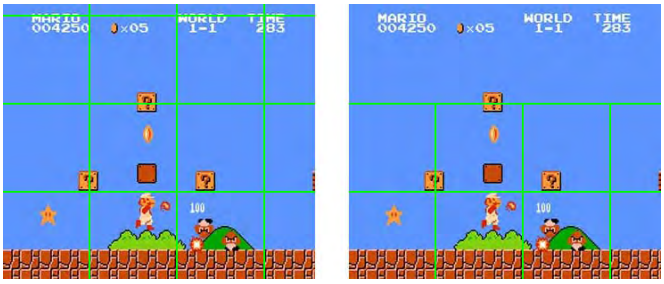


(그림 3) 그리드 기반 공간 분할 방식

광역 충돌 검사의 가장 기본적인 방법 중 하나는 그리드 기반 공간 분할 방식이다. 그리드 기반 공간 분할 방식은 공간을 일정한 크기의 공간인 몇 개의 셀로 나눈 다음, 객체가 다른 객체와 같은 셀에 포함되었을 때에만 미세 충돌 검사를 수행하는 방식이다. 이를 통하여 충돌 검사의 대상이 되는 객체를 여러 그룹으로 나눔으로써 전체 충돌 검사 횟수를 줄일 수 있다. 그림 3 의 경우 광역 충돌 검사를 수행하지 않았을 때에는 7 개의 객체에 대해 각각 미세 충돌 검사를 수행하여야 하므로 총 21 번의 검사가 필요하다. 하지만 공간을 여섯 개의 셀로 나눴을 경우에는 같은 셀에 포함된 객체끼리만 미세 충돌 검사를 수행하면 되므로 6 번만 검사하면 된다.

3. 충돌 검사 빈도에 따른 동적 공간 분할 기법

대부분의 응용에서 구역별 객체 출현 빈도는 일정하지 않다. 비행 시뮬레이션처럼 지상과 공중의 구분이 없는 경우가 아닌 일반적인 3D 게임이나 사이드뷰 기반 2D 게임 등에서 대부분의 객체는 지면과 가까운 공간에 위치해 있다. 따라서 객체의 수가 적은 공간에는 그렇지 않은 공간보다 셀의 크기를 늘려도 충돌 검사가 일어나는 횟수에 별 다른 영향을 미치지 않는다(그림 4).



(그림 4) 기존의 그리드 기반 방식과 객체의 출현 빈도가 낮은 셀을 병합한 그리드 기반 방식

이를 위하여 각 셀에서 평균적으로 발생하는 충돌 검사의 횟수(빈도)를 계산하여 인접한 두 셀이 모두 충돌 검사 빈도가 낮을 때 이 셀들을 병합하면 해당 부분에는 셀의 크기를 늘리는 것과 같은 효과를 얻을 수 있다. 이 때, 충돌 검사 빈도는 각 프레임 별 충돌 검사 횟수의 평균값으로 정의한다.

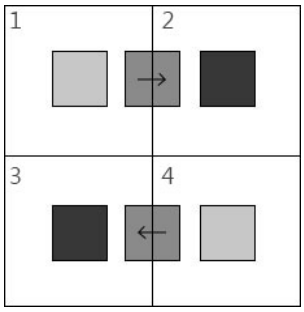
셀 병합이 이루어졌을 때 기존에 없었던 충돌 검사가 발생하는 경우는 광역 충돌 검사를 수행하는 의미가 없으므로 바람직한 상황이 아니다. 따라서 병합하고자 하는 셀들의 충돌 검사 빈도 f 의 합계를 그 셀들을 병합할 경우의 충돌 검사 빈도 g에서 뺀 값이 임의의 값 ε보다 작은 경우에만 병합을 수행한다. 이 과정을 나타내는 함수 P 를 수식으로 나타내면 수식 (1)과 같다.

$$P(f,g)=\begin{cases} if\ g-\sum f<\varepsilon\ then\ 1 \\ if\ g-\sum f<\varepsilon\ then\ 0 \end{cases} \quad (\varepsilon \geq 0) \quad (1)$$

함수 P 의 결과가 1 일 때 병합을 수행한다.

g 가 f 의 합계보다 크거나 같으므로 병합된 셀의 충돌 검사 빈도 f_m 을 g 값으로 설정한다면 f_m 은 f 의 합계보다 g 에서 f 의 합계를 뺀 만큼 커진다. 병합이 반복될 경우 이 값이 누적되어서 최종적으로 병합된 셀의 충돌 검사 빈도와 해당 셀을 이루고 있는 셀들의 본래 충돌 검사 빈도의 합계가 ε 이상 차이가 날 수 있다. 그러므로 f_m의 값은 f의 합계로 한다.

셀 병합 시 임계값 ε를 0 보다 크게 하면 병합 후보가 된 셀들의 충돌 검사 빈도에도 차이가 생긴다. 이 때, 이 셀들을 충돌 검사의 빈도 및 순서의 구분 없이 병합할 경우 효율이 떨어질 수 있다. 그러므로 충돌 검사의 빈도가 낮은 셀부터 우선적으로 병합해야 한다.



(그림 5) 규칙적인 움직임을 보이는 두 객체

그림 5 에서 두 객체는 매 프레임마다 밝은 색으로 표시된 위치에서 어두운 색으로 표시된 위치로 이동하며, ε 의 값은 0.4 라고 가정한다. 1 번 셀과 2 번 셀을 병합할 시 g 에서 f 의 합계를 뺀 값은 0 이므로 2 번 셀은 병합 후보가 된다. 반면 1 번 셀을 3 번 셀과 병합할 시 두 번째 프레임에서 충돌 검사가 1 번 발생하므로 g 에서 f 의 합계를 뺀 값은 1/3 이다. 즉 3 번 셀도 마찬가지로 병합 후보가 된다. 그런데 1 번 셀과 3 번 셀을 먼저 병합한다면 새로 만들어진 셀과 2 번 셀은 병합 시 g 에서 f 의 합계를 뺀 값이 2/3 가 되므로 병합이 이루어지지 않는다. 따라서 최종적으로는 1, 3 번 셀과 2, 4 번 셀이 병합된 형태가 된다. 이 경우 두 셀의 충돌 검사 빈도는 1/3 로, 1, 2 번 셀과 3, 4 번 셀을 병합했을 때 두 셀의 빈도인 0 보다 높다. 이러한 경우는 병합 후보인 두 셀이 둘 다 병합될 수는 없는 상황에서 발생한다. 그러므로 병합 후보 셀이 여러 개일 때는 낮은 빈도의 셀부터 우선적으로 병합해야 한다.

4. 결과

실험은 가로 8 개, 세로 6 개로 이루어진 그리드 위에서 이루어졌고, 해당 그리드에는 5 개의 규칙적으로 움직이는 객체와 2 개의 고정된 객체가 포함되었다. 인텔 i5 2.60GHz CPU 와 Nvidia GTX765M 을 장착한 하드웨어에서 수행되었고, 프로그램은 DirectX 11 을 사용하여 구현하였다.

움직이는 객체가 똑같은 주기로 움직일 경우, 기존의 그리드 기반 공간 분할 방식을 이용하였을 때에는 48 개의 셀에서 프레임당 평균 21 번의 셀 업데이트가 이루어졌다. 반면 본 논문의 방식을 적용하였을 때에는 12 개의 셀에서 프레임당 평균 8.5 번의 셀 업데이트가 이루어졌으며 추가적인 충돌 검사는 발생하지 않았다. 이는 본 논문의 방식을 적용하기 가장 최적의 상태로서, 각 셀의 빈도가 항상 일정하기에 가장 최적화된 상태의 셀 병합을 이룰 수 있었기 때문이다. 반면 객체가 한 쪽에 몰려있는 경우에는 기존의 방식과 비교하여 셀 개수의 경우 약 50%, 셀 업데이트 비용의 경우 약 10%의 감소율을 보였다. 객체들이 밀집한 부분에서는 충돌 검사 빈도가 높아 병합이 일어나지 않았으며, 대부분의 셀 업데이트 또한 병합이 일어나지 않은 부분에서 일어났기에 앞서의 경우에 비해 셀 업데이트 비용 감소 폭이 작았던 것으로 분석된다.

5. 결론

본 논문에서는 각 셀에서 일어나는 충돌 검사의 빈도를 고려하여 추가적인 충돌 검사를 최소화 하면서 셀을 통합하는 방식을 제안하였다. 대부분의 객체들이 규칙적인 움직임을 보일 경우, 적은 횟수의 추가적인 충돌 검사만으로 셀 업데이트 비용을 확연히 낮출 수 있었다. 그러므로 함정과 같이 규칙적인 움직임을 보이는 객체가 많거나, 구역별로 객체 출현 빈도의 차이가 큰 응용 등에서 효율적일 것이다.

6. 감사의 글

논문은 2015 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2015R1A2A2A01008248).

참고문헌

[1] Christer Ericson(2004), "Real-Time Collision Detection", CRC Press

[2] Eitz, M., & Lixu, G. (2007, June). Hierarchical spatial hashing for real-time collision detection. In *Shape Modeling and Applications, 2007. SMI'07. IEEE International Conference on* (pp. 61-70). IEEE.

[3] Collision Detection Part 3: Hierarchical Collision Grids, https://ductomaniac.wordpress.com/2012/03/15/collision-detection-part-3-hierarchical-collision-grids/

[4] Mario Zechner, Robert Green(2012), "Beginning Android Games", Apress Access

[5] Broad Phase Collision Detection Using Spatial Partitioning, http://buildnewgames.com/broad-phase-collision-detection/