충돌 탐지 방법들의 소개

백 낙 훈
동국대학교 컴퓨터·멀티미디어공학과

1. 서 론

컴퓨터 그래픽스나 가상 현실 분야에서와 같이, 실세계에서의 현상들을 컴퓨터 화면 상에 표현해야 하는 경우에는 물체의 외양을 실제와 같이 표현하는 것도 중요하지만, 물체들 간의 상호 작용(interaction)을 사실적으로 표현할 필요가 있다. 기본적인 상호 작용들 중의 하나로는 두 물체들 간의 충돌(collision)을 들 수 있다. 흔히 충돌이라고 표현되지만, 궁극적으로 이 문제는 두 물체들 간의 간섭(interference)이나 교차(intersection)를 찾는 문제로 취급할 수도 있다[1]. 이 글에서는 컴퓨터 그래픽스 분야의 관점에서 충돌을 탐지하는 일반적인 방법들을 소개하고자 한다.

충돌을 처리(collision handling)하는 전체 과정은 크게 두 단계로 분류할 수 있다. 우선, 충돌이 일어나는 지, 아닌지를 알아내는 충돌 탐지(collision detection) 단계가 필요하고, 실제 충돌이 일어나는 경우에는 충돌로 인한 두 물체의 외형이나 운동 궤적에 변화를 가하는 충돌 반응(collision response) 단계가 수행된다[2]. 충돌 탐지 단계에서는 대부분 두 물체의 교차 여부를 판별하는 방식을 택하고 있다. 따라서, 이 단계에서의 여러 알고리즘들은 수치 가공 분야와 같이, 물체들 간의 교차 연산을 수행하는 분야들과도 관계를 가진다. 충돌 반응 단계의 경우는, 물체의 사실적인 변형이나 운동 궤적을 표현할 필요가 있고, 따라서, 궁극적으로 물리 기반 모델링(physically-based modeling) 기법들과 연관된다.

충돌 탐지와 충돌 반응의 관점에서 본다면, 충돌 처리 과정은 전체적으로 그림 1(a)에서와 같이 표현할 수 있다. 충돌 처리의 실제 적용에 있어서는 상당히 많은 수의 물체들을 대상으로 충돌하는 물체들의 쌍을 찾게 되는데, 이 경우에 크게 두 가지 관점에서의 개선 방안이 제기된다. 우선, 아무리 많은 물체를 대상으로 충돌을 탐지하더라도 그 중의 2개 물체만 충돌하게 된다는 점이다. 최초에 충돌하는 두 물체는 충돌 후에 운동 궤적이 바뀔 수 있으므로, 이후에는 전체 충돌 탐지 과정이 다시 수행되어야 할 것이다. 또, 대부분의 응용에서는 충돌 자체가 상당히 드물게 일어난다. 즉, 상대적으로 긴 시간 동안 물체들이 움직인 후에 순간적인 충돌이 한번 일어나게 되는 식이다. 이 때문에, 충돌 탐지 단계는 다시 2단계로 나누어 수행하는 것이 효과적이다.

그림 1(b)는 개선된 충돌 처리 과정의 예이다. 대상이 되는 모든 물체들은 간단한 계산 과정을 통하여, 충돌 가능성이 있는 경우와 없는 경우로 나뉘고, 그들 중에서 충돌 가능성이 있는 물체들에 대해서만 좀더 자세하게 충돌 여부를 탐지한다. 이들을 각각 broad-phase 충돌 탐지(broad-phase collision detection)와 narrow-phase 충돌 탐지(narrow-phase collision detection)라고 분류하기도 한다. broad-phase 충돌 탐지는 충돌 가능성이 전혀 없는 경우들을 최대한 빨리, 최대한 많이 찾아내는 데에 목적이 있다. 이 단계에서의 처리 결과가 좋은 경우에는 실제로 충돌이 일어나는 경우만 남겨질 것이고, narrow-phase 충돌 탐지 단계에서는 꼭 필요한 연산만을 수행함으로써, 전체적인 효율을 높일 수 있다.

이상에서 보는 바와 같이, 전체적인 충돌 처리 과정에서는, broad-phase 충돌 탐지 단계의 효율성에 따라, 이후의 narrow-phase 충돌 탐지 단계나 충돌 반응 단계에서의 계산량이 달라질 수 있다. 따라서, 이제까지의 연구 결과들은 대부분 broad-phase 충돌 탐지 단계의 효율적인 처리를 목적으로

```
while (애니메이션 진행 중) do
  충돌 탐지;
  if (충돌 발생) then 충돌 반응; endif
  화면 출력;
end while
```

(a) 일반적인 충돌 처리 방법

```
while (애니메이션 진행 중) do
  충돌 가능성 탐지;
  if (충돌 가능성 있음) then
    충돌 탐지;
    if (충돌 발생) then 충돌 반응; endif
  end if
  화면 출력;
end while
```

(b) 효율성을 고려한 충돌 처리 방법

그림 1. 충돌 처리 방법의 전체 구조

한다. narrow-phase 충돌 탐지 단계나 충돌 반응 단계에서는 문제의 성격 때문에 택할 수 있는 방법론이 많지 않은 반면에, broad-phase 충돌 탐지 단계의 처리는 다양한 방법들로 접근한 연구 결과들이 발표되었다. 이 글에서는 broad-phase에서의 충돌 탐지 방법들을 우선 설명한 후, narrow-phase에서의 충돌 탐지 방법들을 비롯한 다른 방법들을 살펴보겠다.

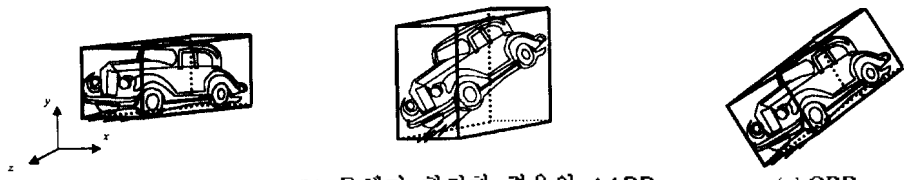
2. Broad-phase 충돌 탐지 방법들

앞에서도 언급했듯이, 광범위한 충돌 탐지 방법들의 목적은 충돌 가능성이 없는 물체들을 처리 과정에서 제거하는 것이다. 이를 위해서는 bounding volume을 이용한 처리가 가장 효과적이고, 실제로 많은 연구 결과들이 다양한 종류의 bounding volume들을 사용하고 있다.

Bounding volume 들 중에서 가장 직관적인 형태로는 bounding box를 들 수 있다. 이 방법은 x, y, z 축을 따라 각 좌표축에서 해당 물체가 차지하는 구간을 표현하는데, 결과적으로는 해당 물체를 둘러싸는, 좌표축에 평행한 직육면체 형태를 표현한다(그림 2(a)). 좌표축에 평행한 특성 때문에, AABB(axis-aligned bounding box)라고도 불린다. 직교 좌표계에서 정의된 물체의 AABB는 단순 작업으로 구해지고, AABB 들 간의 교차 여부 역시 간단하게 검사할 수 있다.

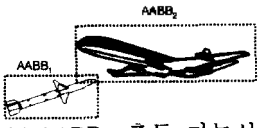
AABB를 사용하는 경우에는 그 효율을 높이기 위해, 추가로 좌표축 상에서의 정렬(axis-sorting) 기법을 사용하는 것이 일반적이다. n개의 물체에 대하여 단순하게 AABB 기법을 적용한다면, $O(n^2)$개의 물체 쌍들에 대하여 AABB의 교차 여부를 검사하여야 한다. 반면에, AABB 들을 x, y, z 축에 투영시킨 결과들을 미리 정렬한 후에, 투영 구간들이 겹치는 것 만을 찾는다면, 시간을 절약할 수 있다. 2개의 AABB가 서로 교차한다면, 각각의 x, y, z 축 투영 결과도 모두 교차하여야 하고, 그 역도 성립한다. 따라서, AABB 들을 각 좌표축을 따라 정렬하는 방식에서는 $O(n \log n)$ 시간의 정렬과 $O(n)$ 시간의 검사로 충돌 가능성이 있는 AABB 들을 모두 찾아낼 수 있다[3].

AABB는 계산하기 쉽다는 장점이 있지만, 일부 물체들에서는 AABB가 물체에 비해 너무 커지는 단점이 있다(그림 2(b)). 이 때문에, 좌표축에 평행한 직육면체를 구하는 대신, 해당 물체를 둘러싸는 최소한의 직육면체를 사용하기도 한다(그림 2(c)). 이 경우를 OBB(oriented bounding box)라고 하는데, 물체에 가장 적합한 bounding box를 계산하기 때문에, 충돌 탐지의 효율을 높일 수 있다. 반면에, 일반적인 물체의 OBB를 계산하는 과정은 꽤 복잡한 과정을 거쳐야 하는 단점이 있다[4,5]. OBB를 사용하는 경우에도 좌표축 정렬(axis-sorting) 기법의 아이디어는 여전히 좋은 효과를 얻을 수 있다. 다만, 정렬의 기준이 될 수 있는 축을 찾기가 애매

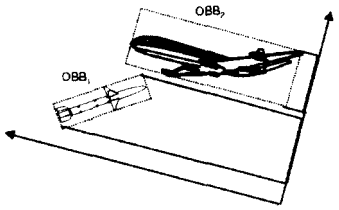


(a) AABB (b) 물체가 회전한 경우의 AABB (c) OBB

그림 2. AABB와 OBB



(a) AABB : 충돌 가능성 있음



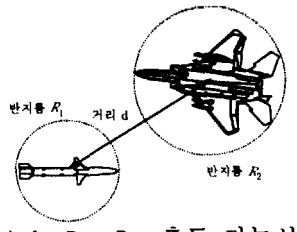
(b) OBB : 충돌 가능성 없음

그림 3. AABB와 OBB에서의 충돌 탐지

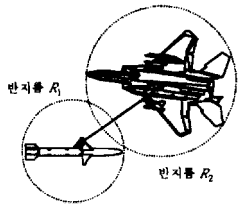
하다는 문제점 때문에, 2개의 OBB에 대한 교차 여부를 판별하는 방식이 주로 사용된다. 그림 3에서와 같이, AABB로는 충돌 가능성이 있다고 판별되는 경우에도 OBB에서는 충돌 가능성이 없음을 정확히 판별하는 것이 가능하다.

AABB 나 OBB는 모두 직육면체 형태로 정의되기 때문에, 물체가 회전 운동을 할 수 있는 경우에는 매번 AABB 나 OBB를 새로 계산해야 한다는 문제점을 가지고 있다. 이 단점을 해결할 수 있는 bounding volume으로는 bounding sphere가 있다. 주어진 물체를 둘러싸는 최소 지름의 구를 사용하는 이 방법은 직관적이고, 충돌 탐지 계산 과정이 단순한 장점을 가진다. 반면에, 일반적인 물체를 둘러싸는 최소 지름의 구를 계산하기는 상당히 까다롭고, 계산기하학(computational geometry)의 연구 결과들을 필요로 한다[6,7]. 이 때문에, bounding sphere를 사용하는 경우의 상당수에서는 최적의 bounding sphere는 아니지만, 근사해로서, 해당 물체의 AABB 나 OBB를 둘러싸는 구를 사용하는 것이 일반적이다. 이 경우, bounding sphere 자체의 크기가 조금 커지는 단점이 있지만, 계산 시간의 면에서 상당한 이득을 볼 수 있다. bounding sphere 간의 교차 여부는 그림 4에서와 같이, 두 sphere 들의 중심점 사이의 거리를 두 sphere의 반지름들의 합과 비교하여 간단히 판별할 수 있다.

AABB와 bounding sphere의 장단점을 절충한 형태로는 k-dop(discrete orientation polytopes)가 있다. 이 방법에서는 x, y, z의 각 좌표축 방향에



(a) $d > R_1 + R_2$: 충돌 가능성 없음



(b) $d \le R_1 + R_2$: 충돌 가능성 있음

그림 4. Bounding sphere에서의 충돌 탐지

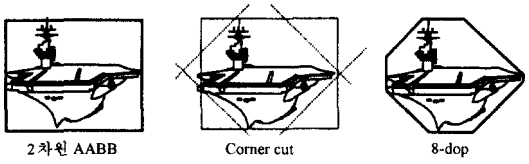


그림 5. 8-dop의 생성 예제

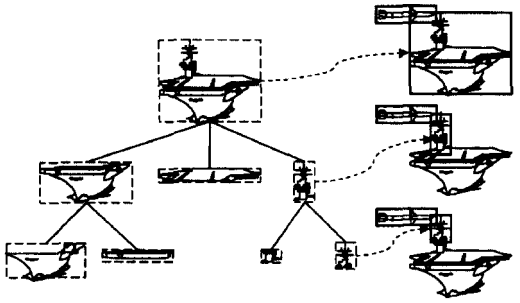


그림 6. 계층 구조에 의한 충돌 탐지

추가하여 대각선 방향으로도 물체가 차지하는 구간을 계산한다. 즉, AABB를 계산한 후, 각 코너(corner)에서 불필요한 부분들을 잘라내는 방식이다. 2차원 응용들에서는 x, y 좌표축 방향과 $y = x$, $y = -x$ 등의 2개 대각선 방향으로, 총 4개 방향의 최대, 최소가 되는 값들을 저장하는 8-dop가 흔히 사용되고, 3차원 응용들에서는 비슷한 방식으로 만

들어지는, 14-dop가 사용된다. *k*-dop 들 간의 교차 여부는 *k*-dop를 생성할 때 사용된 축 방향들에서 각각 구간의 겹침을 검사하면 된다. 즉, AABB에서 교차 여부를 검사하는 방식과 완전히 동일하지만, 검사하는 방향이 더 늘어나는 셈이다. 축 방향으로의 정렬(sorting)도 물론 그대로 적용할 수 있다[8].

3. 계층 구조에 기초한 개선 방법

현재까지 사용되고 있는 bounding volume 들은 이제까지 설명한 AABB, OBB, bounding sphere, *k*-dop 등이 대부분을 차지한다. 어떠한 bounding volume을 사용하든지, 충돌 탐지시의 효율을 높이기 위해서는 추가로 계층 구조(hierarchy)를 도입할 수 있다. 즉, 하나의 물체를 둘러싸는 bounding volume을 하나만 사용하기 보다는 물체의 내부 구조별로 별도의 bounding volume을 계산해 두고, 이를 트리 형태로 저장해 둔다. 다른 물체와의 교차 여부는 트리의 단말 노드로 내려가면서 점차 작은 bounding volume을 사용하여 효율성을 높이는 방식이 된다. OBB에 계층 구조를 도입한 예로는 RAPID library[5]가 있고, *k*-dop를 기반으로 계층 구조를 도입한 예로는 QuickCD library[9]가 있다.

계층 구조 이외에 교차 여부 판별 시의 효율을 높이는 방법으로 공간 분할(local space partitioning) 방법도 사용되고 있다. 이 방법에서는 해당 물체에 적당한 크기의 격자 형태를 겹친 후, 해당 물체가 각 격자의 내부를 차지하고 있는 지를 미리 검사해 둔다. 충돌 탐지 시에는 충돌 가능성이 있는 두 물체의 내부 격자들에 대해서, 물체가 들어가 있는 격자들끼리의 교차 여부를 판별함으로써, 좀더 정확하게 충돌 가능성을 판별하도록 한다[10].

4. Narrow-phase 알고리즘들

물체의 충돌 가능성이 확인되었더라도 대부분의 broad-phase 알고리즘들이 bounding volume 들 간의 충돌만을 확인하기 때문에, 해당 물체들이 실제로 충돌했는지는 narrow-phase 알고리즘들에서 다시 검사한다. 대부분의 응용 프로그램들에서는 물체들을 다면체로 표현하는 경우가 많은데, 다면체들 간의 교차 여부는 다양한 방법들이 제시되어 있다. 가장 직관적인 방법은 두 다면체들 간의 기하학적 관계를 직접 이용하여, 다면체를 구성하는 꼭지점, 변, 면이 각각 다른 다면체의 내부에 포함되는 지를 검사하는 것이다[11].

계산기하학 분야의 보로노이 다이아그램(Voronoi diagram)은 주어진 기하학 원소들에 대해서, 전체 공간 상의 모든 점들에 가장 가까운 기하학 원소 하나를 대응시킴으로써 얻어진다[6]. 최종 결과는 전체 공간이 여러 개의 공간으로 분할되고, 각 공간에 속한 점들은 모두 주어진 기하학 원소들 중에서 가장 가까운 원소가 특정한 하나의 기하학 원소로 정해지는 특징이 있다. Closest feature 알고리즘은 보로노이 다이아그램의 성질을 이용하여, 운동하는 두 다면체 간의 충돌 여부를 계속 탐지하는 방법이다[12]. 각 다면체마다 보로노이 다이아그램을 미리 계산해 두고, 다른 다면체와의 거리를 계산할 때에는, 보로노이 다이아그램 상에서 대응되는 기하학 원소로부터의 거리를 구함으로써 효율적인 계산이 가능하다. 특히, 다른 다면체가 운동하는 경우에는 보로노이 다이아그램의 경계를 넘어가는 순간에 대응되는 기하학 원소를 교체함으로써, 두 물체 간의 거리 관계를 지속적으로 유지할 수 있다. 두 물체 간의 거리가 0 이하가 되는 순간에 두 물체는 충돌하게 되고, 충돌 지점은 바로 거리 계산에 사용된 기하학 원소들이 된다.

Separating plane 방법은 이와 비슷한 아이디어로, 3차원 공간에서, 보로노이 다이아그램 대신에 두 물체를 분리하는 평면을 계속 유지하는 방법이다. 이 방법에서는 주어진 두 물체를 분리하는 평면의 방정식을 계산해 둔 후, 어느 한 물체가 이동하여 이 평면을 넘어가는 순간, 새로운 separating plane을 계산하는 방식으로 작동한다. 두 물체를 분리하는 separating plane을 계산하는 것이 불가능한 경우에 두 물체는 충돌한 것으로 판정하고, 충돌 위치는 separating plane의 정보를 이용하여 구할 수 있다. 특히 이 방법은 곡면들 간의 충돌 시에도 이용 가능하다[13].

5. Single-phase 알고리즘들

일반적인 충돌 처리 방법들에서는 broad-phase 알고리즘과 narrow-phase 알고리즘들이 별도로 분

리되지만, 두 단계를 한꺼번에 처리하는 single-phase 알고리즘들도 연구되어 있다. 이들은 특별한 자료구조나 하드웨어를 이용하여, 충돌 탐지 단계를 하나의 알고리즘으로 처리하여도 전체적인 처리시간에 부담을 주지 않는다.

대표적인 single-phase 알고리즘인 BSP-tree(binary space partitioning tree) 방법은 원래 화면 출력을 위한 렌더링(rendering) 목적으로 설계되었지만, 자료 구조 자체가 충돌 탐지에도 효과적이다. 주어진 공간을 2개의 작은 공간으로 분할하는 과정을 반복적으로 적용해 나가는 이 방법은 자료 구조를 생성하는 전처리 단계에서는 시간이 걸릴 수 있지만, 이후에는 렌더링이나 충돌 탐지가 실시간에 이루어질 수 있다는 장점 때문에, 최근에는 특히 1인칭 슈팅 게임(first person shooting game)을 비롯한 실시간 응용 프로그램들에서 활발히 사용되는 추세이다. BSP-tree에 대한 자세한 정보는 다른 문헌들[1,2]을 참고하기 바란다.

충돌 탐지를 이산 형태로 구현한 예로는 깊이 버퍼(depth buffer) 방법이 있다. 이 방법에서는 일반적인 Z-버퍼가 최소의 z 값을 저장하는 반면에, 각 물체가 차지하는 공간 상의 구간 $[z_{min}, z_{max}]$를 동시에 버퍼에 저장해 두었다가, 다른 물체와의 충돌 여부를 판별하는 방식을 취한다. 이 방법은 하드웨어적으로 구간을 저장할 수 있는 경우에는 효율적으로 충돌 여부를 탐지할 수 있는 장점이 있다.

6. 시간을 고려한 bounding volume

움직이는 3차원 물체는 주어진 시간 간격 동안에 3차원 공간에서의 궤적이 스웹트 볼륨(swept volume)을 형성한다. 두 물체의 스웹트 볼륨들에 대해서 교차 여부를 검사하면, 충돌 가능성을 정확하게 판별할 수 있다. 특히, 일반적인 bounding volume 들이 시간축 상에서는 특정 시점만을 샘플링(sampling)하기 때문에, 경우에 따라서는 충돌 탐지에 실패하는 경우도 생길 수 있는 반면에, 스웹트 볼륨에서는 반드시 충돌을 탐지하게 된다. 반면에, 일반적인 물체의 스웹트 볼륨을 계산하는 문제는 상당히 까다로와서, 특히 물체의 운동 궤적이 실시간으로 생성되는 응용 프로그램들에서는 스웹트 볼륨을 직접 사용하기가 어렵다.

Parabolic horn 방법은 주어진 물체의 bounding sphere를 계산한 후에, bounding sphere의 swept volume을 계산한다. 주어진 시간 간격 $t \in [0, T]$에서, 물체의 가속도 $\mathbf{a}(t)$가 상수 f에 대하여 $|\mathbf{a}(t)| \le f$의 관계를 만족한다면, 뉴튼의 운동 방정식으로부터 물체의 위치 $\mathbf{x}(t)$는 $|\mathbf{x}(t) - (\mathbf{x}(0) + \mathbf{v}(0))t| \le (f/t^2)$의 관계식으로 추정할 수 있다. 이 관계식을 bounding sphere에 적용하면, parabolic horn 형태가 만들어 지고, 이를 이용하여 정확한 충돌 탐지 여부를 판별할 수 있다[14]. 이 방법은 비교적 최근에 제안되어, 격자 형태를 이용한 가속 기법들과 결합하는 등의 개선이 이루어지고 있다.

7. 맺는 말

두 물체의 충돌을 처리하기 위해서는 우선적으로 충돌을 탐지하여야 한다. 효율적으로 충돌을 탐지하기 위한 bounding volume 들이 다양하게 개발되었고, 이들을 이용한 알고리즘들이 연구되어 왔다. 이 글에서는 이들 알고리즘들 중에서 자주 사용되는 것들을 중심으로 살펴보았다. 충돌이 탐지된 이후에는 충돌한 두 물체의 운동 궤적을 수정하는 충돌 반응 단계를 수행하여야 하는데, 이 과정에서는 물리 기반 모델링에 기초한 반응이 필수적이다. 물리 기반 모델링은 현재에 와서는 컴퓨터 그래픽스 분야 내에서 차지하는 비중이 상당한 관계로, 짧은 지면으로 설명하기는 곤란하다. 이를 위해서는 별도의 문헌을 참고하기 바란다.

참고문헌

1. J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics, Principles and Practice, 2nd Edition*, Addison-Wesley, 1990.
2. A. Watt and F. Policarpo, *3D Games*, Vol. 1, Addison-Wesley, 2001.
3. J. Cohen, M. Lin, D. Manocha, M. Ponamgi, "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments", *Symp. on Interactive 3D graphics*, pp. 189-196, 1995.
4. S. Gottschalk, M. Lin, and D. Manocha, "OBB-Tree: A Hierarchical Structure for Rapid In-

terference Detection", *SIGGRAPH'96*, pp. 171-180, 1996.
5. http://www.cs.unc.edu/~geom/OBB/OBBT.html
6. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 1997.
7. http://www-vision.uscd.edu/~dwhite/ball.html
8. J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan, "Efficient Collision Detection Using Bounding Volume Hierarchies of *k*-DOPs", *IEEE Trans. on Visualization and Comp. Graphics*, Vol. 4, No. 1, pp. 21-36, 1998.
9. http://www.ams.sunysb.edu/~jklosow/quickcd/
10. Garcia-Alonso *et al.*, "Solving the collision detection problem", *IEEE CG&A*, Vol. 14, No. 3, pp. 36-43, 1995.
11. M. Moore and J. Wilhelm, "Collision Detection and Response for Computer Animation", *SIGGRAPH'88*, pp. 289-298, 1988.
12. M. Lin, *Efficient Collision Detection for Animation and Robotics*, Ph.D. Thesis, Univ. of California, Berkeley, 1993.
13. D. Baraff, "Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation", *SIGGAPH'90*, pp. 19-28, 1990.
14. P. Hubbard, "Interactive Collision Detection", *Proc. IEEE Symp. on Res. Frontiers in Virtual Reality*, pp. 24-31, 1993.