Terrain Following을 위한 인접지역 탐색 알고리즘

(A Neighboring Area Search Algorithm for Terrain Following)

김 종 혁 † 최 윤 철 †† 고 견 †††
(Jong-Hyuk Kim) (Yoon-Chul Choy) (Kyun Koh)

요 약 Terrain Following이란 가상환경 내에서 지형의 표면을 이동할 때 지형의 모양에 따라 자연스럽게 지표 위를 이동할 수 있게 하는 것으로 가상환경 내에서 중력의 효과를 실현해 주는 기술이다. Terrain Following을 하기 위한 전통적인 접근 방법은 충돌감지(Collision Detection) 알고리즘을 이용하는 것이지만, 일반적인 충돌감지 알고리즘을 Terrain Following에 적용시키는 것은 계산에 드는 시간 비용이 너무 크다. 이러한 어려움 때문에, 많은 가상환경 및 시뮬레이션 분야에서는 높이(z)가 일정한 지형을 사용함으로써 이러한 문제들을 회피하고 있는 실정이다.

이에 본 논문에서는 네비게이션시 발생하는 근접성(Locality)에 기반한 인접지역 탐색(Neighboring Area Search) 기법을 사용하여 Terrain Following 알고리즘의 수행 시간을 단축시켜 Terrain Following을 적용시키는데 발생하는 부하를 줄일 수 있는 알고리즘의 제시를 목적으로 한다. 이는 Terrain Following 기법 자체의 속도 향상을 기대할 수 있고 동시에 Terrain Following을 구현하고 유지하는데 드는 부하를 줄임으로써, 가상환경을 구축하는데 사용되는 워크스테이션을 Terrain Following 이외에 다른 부분의 구현에 이용할 수 있는 부가적 효과를 가져다 줄 수 있다.

Abstract Terrain Following means that a mobile object, such as a user's avatar, must follow terrain, remaining in contact with the ground at all times in virtual environments. This makes a virtual environment have the effects of gravity. Terrain Following is often done using collision detection; however, this is inefficient, because general collision detection solves a problem that is inherently more complex than merely determining terrain contact points. Many virtual environments avoid the expense by utilizing a flat terrain with a constant altitude everywhere. This makes a terrain following trivial, but lacks realism.

This paper provides an algorithm and a data structure for a terrain following using a neighboring area search as a way to search neighboring polygons. Because this algorithm uses a pre-processing step that stores the terrain polygons for calculating, it results in reducing overheads to workstations that is used to construct and maintain a virtual environment. Consequently, workstations can be used to apply not only a terrain following but also other things.

1. 서 론

Terrain Following이란 가상환경 내에서 지형의 표면을 이동할 때 지형의 모양에 따라 자연스럽게 지표 위를 이동할 수 있게 하는 것으로 가상환경 내에서 중력의 효과를 구현하는 기술이다.

Terrain Following을 하기 위한 전통적인 접근 방법은 충돌감지(Collision Detection) 알고리즘을 이용하는 것이다. 충돌감지 알고리즘이란 물체들간의 간섭을 감지하기 위해 고안된 알고리즘으로 컴퓨터 그래픽스, 로

† 비 회 원 : 연세대학교 컴퓨터과학과
iwannado@korea.com
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
ycchoy@rainbow.yonsei.ac.kr
††† 종신회원 : 청주대학교 컴퓨터정보공학과 교수
kyunkoh@rainbow.yonsei.ac.kr
논문접수 : 2000년 4월 17일
심사완료 : 2001년 8월 16일

보틱스 분야에서 이용되고 있다[1]. 그러나 일반적인 충돌감지 알고리즘을 Terrain Following에 적용시키는 것은 계산에 드는 시간 비용이 너무 크다. 충돌감지 알고리즘은 오직 물체들간의 충돌을 감지하기 위해 고안된 알고리즘으로 비교적 단순한 Terrain Following에 이용할 경우 불필요한 연산을 많이 하게 되어 시간적 손실을 초래할 수 있다. 이러한 어려움 때문에, 많은 가상환경 및 시뮬레이션 분야에서는 높이(z)가 일정한 지형을 사용함으로써 이러한 문제들을 회피하고 있는 실정이다[2].

현재까지 알려진 Terrain Following을 위한 알고리즘은 Quadtree를 지형데이타의 표현을 위한 기본 데이타 구조로서 이용하는 방법이다. 이 방법은 트리 구조를 사용하기 때문에 데이타가 한쪽으로 치우치는 경우 저장공간과 탐색 속도에서 비효율적인 모습을 보이게 된다. 또한 알고리즘 측면에 있어서도 Terrain Following 방식에 특화된 알고리즘을 사용하는 것이 아닌 데이타 구조에 따른 탐색방식을 그대로 사용하기 때문에 폴리곤을 탐색하는 과정에서 항상 루트에서부터 탐색을 시작하게 된다[2].

이에 본 논문에서는 네비게이션시 발생하는 근접성(Locality)에 기반한 인접지역 탐색(Neighboring Area Search) 기법을 사용하여 실시간 계산을 좀더 빠르게 처리할 수 있는 알고리즘과 데이타 구조를 제시한다. Terrain Following 연산에 이용될 수 있는 부분을 미리 데이타 구조로 구성하여 매시간 반복적으로 사용되는 Terrain Following 알고리즘의 수행 시간을 단축시켜 가상환경에서 Terrain Following을 적용시키는데 발생하는 부하를 줄일 수 있는 알고리즘과 데이타 구조 제시를 목적으로 한다. 이는 Terrain Following 기법 자체의 속도 향상을 기대할 수 있고, 동시에 Terrain Following을 구현하고 유지하는데 드는 부하를 줄임으로써 가상환경을 구축하는데 사용되는 워크스테이션을 Terrain Following 이외에 다른 부분의 구현에 이용할 수 있는 연쇄적 효과를 가져다 줄 수 있다.

본문의 구성은 다음과 같다. 1장에서는 전체 논문을 개괄적으로 설명하고, 2장에서는 지금까지 진행되어왔던 연구 중 본 논문의 내용과 관련 있는 연구들을 살펴볼 것이다. 3장에서는 본 논문에서 제시하는 새로운 기법인 인접지역 탐색 알고리즘과 구현 시 발생했던 문제점을 토대로 향상된 알고리즘을 제시하고, 알고리즘에 필요한 데이타 구조에 대해 알아본다. 4장에서는 본 알고리즘을 이용한 실험적 구현에 대해서 기술하고 그 성능에 대해 평가하며, 마지막으로 5장에서는 결론과 향후 연구 방향에 대하서 언급한다.

2. 관련연구

2.1 Terrain Following 구현 기법

2.1.1 충돌감지 기법을 이용한 Terrain Following

충돌감지란 3D 공간에서 서로 다른 물체가 동일한 시각에 같은 위치를 점유할 경우 이 물체들간의 충돌여부를 판가름 해 주는 것을 의미한다.

가상환경 참여자가 사용자 인터페이스 상에서 앞으로 한 걸음 다가간다는 것은, 그 즉시 시점(Viewpoint)이 한 걸음만큼 이동하고 그 다음에 다시 렌더링 되는 것이다. 이 과정을 반복하여 수행하다 보면 바로 가상 세계가 움직이는 것으로 느끼게 되는 것이다. 그러므로 단지 새로운 위치로 이동하는 곳에 어떤 하나의 객체가 존재하는지를 확인하는 것만으로는 충돌감지를 수행할 수 없다. 이 확인 작업을 한 장소에서 다른 장소로의 이동하는 중 거치게 되는 모든 영역에 대해 수행해야 한다[3]. 이 과정을 VRML에서의 충돌감지를 예로 들어 설명하면 (그림1)과 같다.

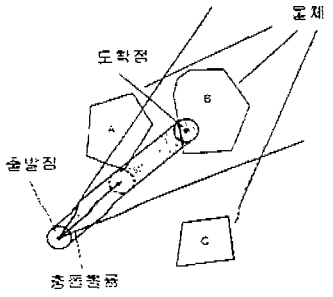


그림 1 VRML에서의 충돌감지

VRML에서 가상환경 참여자는 원통으로 간주한다. (그림 1)에서 볼 수 있듯이 참여자는 객체 B와 도착점에서 충돌하고, 객체 A와 출발점과 도착점 사이의 이동도중에 충돌한다. 이 그림에서는 객체 A가 출발점에 더 가깝기 때문에 참여자는 객체 A와 충돌한 후 점선으로 표시된 원에 위치하게 된다.

이러한 충돌감지는 가상환경 참여자가 움직일 때마다 매번 반복적으로 적용되어야 하기 때문에 가상환경 네비게이션시 시스템 부하의 커다란 원인 중 하나로 간주되고 있다. 이러한 어려움으로 많은 가상환경 및 시뮬레이션 분야에서는 높이(z)가 일정한 지형을 사용함으로써 이러한 문제들을 회피하거나 최소화하고 있다[2].

2.2.2 QOTA(Quick Oriented Terrain Algorithm)

현재까지 알려진 Terrain Following을 위한 알고리즘

에서는 Quadtree[4]를 지형데이타의 표현을 위한 기본 데이타 구조로서 이용하고 있다. 이 방법은 가상환경을 트리 구조로 구성하여 검색하기 때문에 기존의 방식에 비해 빠른 탐색시간을 보인다는 장점을 지닌다. (그림2)는 Quadtree를 이용하여 지형을 구조화시키는 예이다.

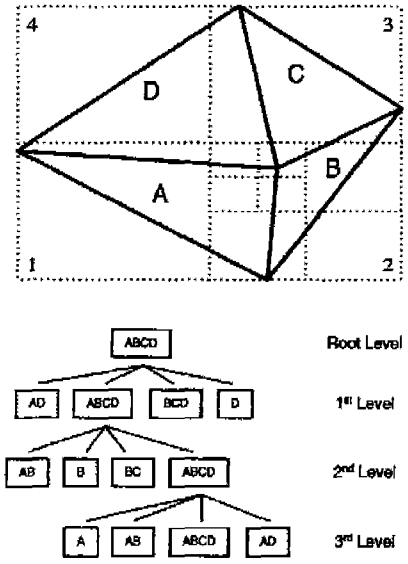


그림 2 Quadtree 데이타 구조[5]

QOTA에서 현재 위치 P가 지형의 어떠한 폴리곤 위에 위치하는지를 결정하는 과정을 살펴보면 다음과 같다. 먼저, 현재 위치 P가 존재할 수 있는 후보자 폴리곤을 탐색한다. 후보자 폴리곤을 찾기 위해서 정확하고 복잡한 계산식을 적용시키기보다는 간단하면서 빠른 계산식을 적용시킨다. 이 과정은 현재 위치 P가 속하는 정확한 폴리곤을 찾는 것이 아닌 속하지 않은 폴리곤을 제외시키는 작업이다. 다음으로 앞의 과정에서 추출된 후보자 폴리곤에서 실제로 위치하는 하나의 폴리곤을 결정한다. 후보자의 수가 전체 폴리곤의 개수에 비해 많지 않기 때문에 정확하고 복잡한 계산식을 각각의 후보자 폴리곤에 적용시켜 하나의 폴리곤을 결정한다. 마지막으로 실제로 현재 위치 P가 존재하는 하나의 폴리곤에서 평면의 방정식을 추출하여 현재 위치 $P(P_x, P_y)$에서의 높이(P_z)를 결정한다.

Quadtree를 이용하여 데이타를 구성하는 방식은 균형격자 방식으로 되어있는 데이타에 매우 적합한 방법이다. 이런 경우 완벽한 형태의 트리 구조(Complete tree)를 갖게되어 탐색시간을 최소화 할 수 있는 장점을 지닌다[6].

그러나 불균형 격자 방식으로 구성된 데이타에서 몇 가지 문제점을 가진다. 가령 Quadtree가 나눠지는 선상에 점이 존재하게 될 경우 이 점이 속하는 구역을 결정하는데 있어 문제가 될 수 있다. 또한, 지형 데이타의 편중(Skew) 현상이 클 때 Quadtree 방법은 저장 공간과 탐색 속도에서 비효율적인 모습을 보이게 된다[6].

3. 인접지역 탐색 알고리즘

3.1 탐색 알고리즘

3.1.1 인접지역 탐색 기법

지형 데이타는 방대한 자료를 가진다는 점과 갱신이 빈번하게 발생하지 않는다는 특성을 가지고 있고 Terrain Following은 매시간 반복적으로 발생한다는 특징이 있기 때문에 성능 향상을 위해서는 기존에 존재하는 알고리즘들 보다 특화된 알고리즘이 필요하다[7].

Terrain Following 시 특징을 살펴보면, 현재 위치 P에서 다음 위치 P'를 탐색할 경우 현재 위치 주변에 있을 확률이 크다는 것을 알 수 있다(그림 3). 즉, 현재 위치 P에서 다음의 위치 P'를 예측할 수 없을 때, P가 위치했던 폴리곤에서 찾을 경우 폴리곤 내부에 P'가 존재할 확률이 가장 크고, P가 위치했던 폴리곤과 선으로 접하는 세 개의 폴리곤에서 찾을 경우 내부에 P'가 존재할 확률이 그 다음으로 크다. 마지막으로 P가 위치했던 폴리곤과 점으로 접하는 나머지 여러 개의 폴리곤에서 찾을 경우 내부에 P'가 존재할 확률이 가장 적다.

이러한 지역성에 기반한 인접지역 탐색 기법을 이용하면 기존의 방식에 비해 좀더 빠른 탐색 시간을 예상할 수 있다. 인접지역 탐색 기법을 이용한 탐색 알고리즘을 요약하면 다음과 같다.

° **인접지역 탐색 알고리즘**

1. 시작점에서의 높이(Z) 추출
2. 우선 순위에 따라 현재 속해있는 폴리곤 탐색
   2-1. 우선적으로 현재 속한 폴리곤에서 탐색(1번 구역 탐색)
   2-2. 만약 존재하지 않으면, 다음 순위에 있는 3개의 폴리곤에서 탐색(2번 구역 탐색)
   2-3. 만약 존재하지 않으면, 나머지 연결된 폴리곤에서 탐색(3번 구역 탐색)
3. 실제 높이(Z) 정보 추출
4. 2와 3의 과정을 반복하며 네비게이션

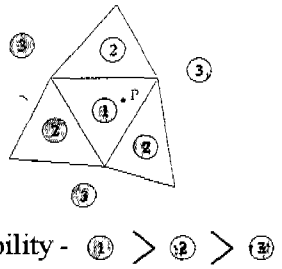
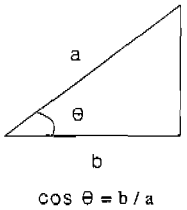


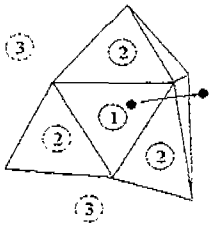
그림 3 Terrain Following 시 현재 위치에서 다음 위치를 찾을 수 있는 확률

기존의 인접지역 탐색 알고리즘에서의 가장 큰 문제점으로는 추출 빈도(Sampling Rate)에 따른 폴리곤 건너뜀(Skipping) 현상을 들 수 있다(그림4 (a)). 이 문제의 해결책으로는 추출 빈도를 증가시켜 세밀하게 네비게이션 하는 방법과 찾을 수 없는 경우 기존의 트리 탐색을 이용하여 다시 찾는 방법 등을 생각할 수 있다.
본 알고리즘에서는 폴리곤의 기울기에 따라 추출 빈도를 다르게 적용시키는 방법을 사용한다. 즉, 평지에서의 추출 빈도를 기준으로 지표면과 현재 폴리곤의 기울기를 계산하여 기울기에 맞게 조정해 주는 방법을 사용한다. 이와 같은 방법을 사용하면 경사가 가파른 지형 위를 움직일 때 보폭이 작아지는 실제 현상과 비슷한 효과를 얻을 수 있고, 앞에서 언급했던 폴리곤 건너뜀 현상이 발생할 수 있는 확률이 줄어드는 결과를 가져온다.

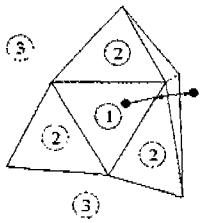
지표면과 현재 위치하는 폴리곤의 기울기 θ를 계산하여, 평지에서의 추출 빈도(a)를 기준으로 현재 지표면과 폴리곤의 기울기에 맞게 추출 빈도(b)를 조정(b = a · cos θ)



폴리곤 건너뜀 현상을 해결하기 위해 추출 빈도를 증가시켜 세밀하게 네비게이션 하는 방법과 찾을 수 없는 경우 기존의 트리 탐색을 이용하여 다시 찾는 방법 등을 생각할 수 있으나 완전한 해결책이라고 볼 수 없다. 따라서 중간 경로를 이용한 새로운 인접지역 탐색 알고리즘을 사용해야 한다. 이 방법은 기존의 방식으로 찾을 수 없을 경우, 즉 탐색 범위를 벗어나는 경우 가상의 경로를 만들어 연쇄적으로 인접지역 탐색을 하도록 유도하는 방식이다. 이 방법을 이용하면 비록 기존의 방법에 비해 탐색 시간은 증가하지만 폴리곤 건너뜀 현상은 방지할 수 있다(그림4 (b)).



(a) 추출 빈도에 따른 폴리곤 건너뜀 현상



(b) 중간 경로를 이용한 새로운 인접지역 탐색기법

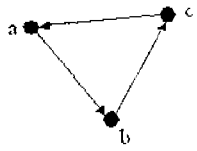
그림 4 지역 탐색 기법의 문제점과 해결 방안

3.1.2 CCW(Counter-Clock-Wise)

CCW란 Counter-Clock-Wise의 약어로 알고리즘 분야에서 두 선의 교차(Intersect)를 알아내는데 사용되는 기법이다. 본 논문에서는 현재의 위치 P점이 폴리곤 내부에 속하는지를 알기 위해 CCW를 사용한다. CCW의 정의는 다음과 같다.

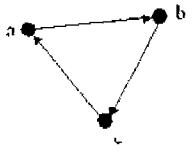
° CCW의 정의
- 세 개의 점 a, b, c가 주어진 경우,
  CCW(a, b, c) = + 1
  · 만약 세 개의 점이 시계 반대방향의 순서일 경우 (그림5 (a))
  CCW(a, b, c) = - 1
  · 만약 세 개의 점이 시계 방향의 순서일 경우 (그림5 (b))

cf. CCW(a, b, c) = CCW(b, c, a) = CCW(c, a, b)



CCW(a,b,c)= +1

(a) 시계 방향



CCW(a,b,c)= -1

(b) 시계 반대 방향

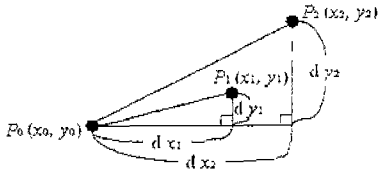
그림 5 CCW의 정의

세 점 $P_0(x_0, y_0)$, $P_1(x_1, y_1)$, $P_2(x_2, y_2)$가 주어졌을 경우 CCW(P_0, P_1, P_2)의 실제적인 연산은 다음과 같다.

° **CCW(P_0, P_1, P_2) 연산**

- $P_0(x_0, y_0)$, $P_1(x_1, y_1)$, $P_2(x_2, y_2)$가 주어졌을 경우
  $dx_1 = x_1 - x_0$, $dy_1 = y_1 - y_0$
  $dx_2 = x_2 - x_0$, $dy_2 = y_2 - y_0$ (그림6)

모든 d가 음이 아닐 경우
  CCW(P_0, P_1, P_2) = + 1
  · 만약 $dy_2/dx_2 > dy_1/dx_1$ 이거나 혹은 기울기(P_0, P_1) > 기울기(P_1, P_2)
  dx가 0인 경우 $dy_2 \cdot dx_1 > dy_1 \cdot dx_2$로 결정



$dx_1 = x_1 - x_0$ $dy_1 = y_1 - y_0$
$dx_2 = x_2 - x_0$ $dy_2 = y_2 - y_0$

그림 6 CCW(P_0, P_1, P_2) 연산

현재 위치 P가 폴리곤 내부에 존재하는지를 알기 위해서는 반복적인 CCW 연산 과정이 필요하다. 우선 폴리곤을 구성하는 점 P_0, P_1, P_2가 주어지면 CCW(P_0, P_1, P_2)를 계산한다. CCW(P_0, P_1, P_2)의 값이 +1인 경우, 각각의 P_0, P_1, P_2를 P로 치환했을 때의 CCW 값도 항상 +1을 가져야 한다. 즉, CCW(P, P_1, P_2) = +1, CCW(P_0, P, P_2) = +1, CCW(P_0, P_1, P) = +1인 경우에 점 P는 P_0, P_1, P_2로 이루어진 삼각형 내부에 존재하게 된다. 마찬가지로 CCW(P_0, P_1, P_2)의 값이 -1인 경우, 각각의 P_0, P_1, P_2를 P로 치환했을 때의 CCW 값도 항상 -1을 가져야 한다. 위의 사항은 다음과 같이 정리된다.

° **점 P가 폴리곤 내부에 존재하는지의 검사**

- CCW(P_0, P_1, P_2) = + 1 일 경우
  CCW(P, P_1, P_2) = + 1
  CCW(P_0, P, P_2) = + 1
  CCW(P_0, P_1, P) = + 1
- CCW(P_0, P_1, P_2) = - 1 일 경우
  CCW(P, P_1, P_2) = - 1
  CCW(P_0, P, P_2) = - 1
  CCW(P_0, P_1, P) = - 1

3.2 데이타 구조

인접지역 탐색을 위한 데이타 구조는 크게 Polygons와 Points로 나뉜다.

Polygons의 구성은 다음과 같다. ID는 폴리곤의 구분을 위한 폴리곤 이름을 나타내고 Points는 폴리곤을 구성하고 있는 세 점의 이름을 의미한다. Bounding Box는 현재의 위치가 폴리곤 내부에 포함되는지에 대한 계산을 빠르게 하기 위해 사용되는 최소 경계 사각형으로 폴리곤을 둘러싸는 최소점과 최대점을 저장한다[8]. Neighbors는 현재 폴리곤과 인접하고 있는 폴리곤들의 이름을 나타내고, Etc.에서는 폴리곤의 색 정보나 폴리곤의 진입을 막기 접근 불가 표시등의 기타 정보를 저장하고 있다(표1).

표 1 Polygons 데이타 구조

ID	Points	Bounding Box	Neighbors	Etc.
A	1, 2, 3	(Ax_1, Ay_1) (Ax_2, Ay_2)	B, C, D, E, F, I	
B	1, 3, 7	(Bx_1, By_1) (Bx_2, By_2)	A, C, D, F	
C	2, 4, 8	(Cx_1, Cy_1) (Cx_2, Cy_2)	A, B, D, F, G, I, K	
...				

Points는 두 부분으로 구성된다. ID는 점을 구분하는 이름으로 Points에서는 폴리곤을 구성하는 세 점을 표현할 때 실제 점의 좌표가 아닌 각 점의 ID를 저장하고 있고 (x, y, z)는 각 점에 대한 실제 (x, y, z) 좌표를 나타낸다(표2).

표 2 Points 데이타 구조

ID	(x, y, z)
1	(1.0, 2.2, 3.5)
2	(2.1, 4.4, 7.2)
3	(2.8, 4.8, 8.1)
...	

4. 시스템의 구현 및 성능 평가

본 논문에서는 인접지역 탐색 알고리즘 구현을 위해 OpenGL기술을 사용한다[9]. 다양한 폴리곤 수를 갖는 지형 데이타를 사용하였고, 펜티엄 II 400, 램 256M, 그래픽 카드 리바 TNT 환경 하에서, 윈도우즈 2000, Visual C++ 6.0을 사용하여 구현하였다(그림7).

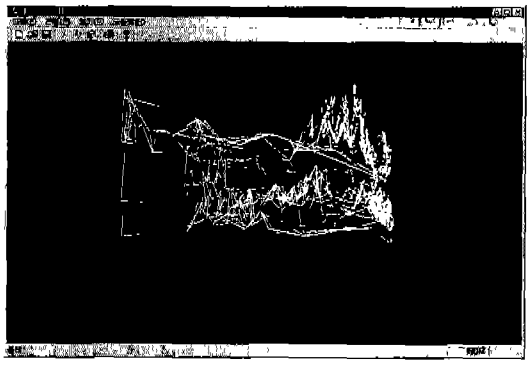


그림 7 Terrain Following 구현

4.1 평균 비교 횟수

본 논문에서는 알고리즘의 성능 평가를 위해 실리콘 그래픽스사에서 제공하는 IRIS Performer Toolkit을 사용한 충돌감지 알고리즘과 QOTA와 비교하였다. 실리콘 그래픽스사의 워크스테이션에서 가상환경을 구축할 때 많은 사람들이 IRIS Performer Toolkit을 사용한다. 이것은 라이브러리 형태로 제공되어 개발자의 편의를 제공해 주는데, 그 중 빠른 속도로 충돌감지를 가능케 해주는 서브루틴이 존재한다. 이 루틴을 사용하면 기하에서 선의 교차를 빠르게 계산할 수 있어 Terrain Following 구현에 이용할 수 있다[2].

(표3)은 본 논문에서의 인접지역 탐색 알고리즘을 기존의 Performer 라이브러리 및 QOTA[2]와 비교한 것이다.

표 3 Performer와 QOTA, 인접지역 탐색 알고리즘의 비교

	Performer	QOTA	개선된 QOTA	인접지역 탐색 알고리즘
폴리곤 수	13,346	13,346	13,346	13,346
평균 비교 횟수	1881	19.2	4.87	1.66

같은 수의 폴리곤으로 비교했을 경우 평균 비교 횟수는 Performer에서 1881, QOTA에서 19.2, 인접지역 탐색 알고리즘에서 1.66의 결과를 보여 주었다. Performer와 QOTA의 성능 차이는 약 100배, QOTA와 인접지역 탐색 알고리즘의 성능 차이는 약 10배 정도로 나타난다. Performer와 QOTA의 성능 차이는 충돌감지 알고리즘과 Terrain Following 알고리즘의 성능 차이에 기인한다. 앞에서 언급했듯이 충돌감지 알고리즘은 오직 물체들간의 충돌을 감지하기 위해 고안된 알고리즘으로 비교적 단순한 Terrain Following에 이용할 경우 불필요한 연산을 많이 하게 되어 시간적 손실을 초래할 수 있다. 폴리곤 13,346개의 지형에서 충돌감지 알고리즘과 Terrain Following 알고리즘과의 성능은 약 100배 정도의 차이를 보여 준다.

QOTA와 인접지역 탐색 알고리즘의 성능 차이는 Terrain Following의 지역성에 기인한다. Terrain Following 시 대부분 경우에 이전에 있었던 폴리곤에 다시 위치할 확률이 가장 크다. 그러나 QOTA에서는 이러한 가능성을 무시하고 항상 새롭게 탐색을 반복한다. 가장 빈번하게 발생하는 상황에서 한 번의 비교로 검색을 완료할 수 있는 것이 QOTA와 지역 탐색 알고리즘의 결정적인 성능 차이 요인으로 분석된다. 개선된 QOTA는 기존 QOTA에 이러한 네비게이션의 지역성을 반영시킨 것으로, 이 개선만으로도 커다란 성능 향상이 있음을 알 수 있다. 지역 탐색 알고리즘은 QOTA에서 사용하는 데이타 구조가 아닌 네비게이션의 지역성을 반영할 수 있는 데이타 구조를 새롭게 구성한 것으로 개선된 QOTA와 비교하면 이에 따른 성능 향상을 확인 할 수 있다. 즉 QOTA와 인접지역 탐색 알고리즘의 성능 차이는 네비게이션의 지역성 반영과 그에 맞는 데이타 구조 구성의 두 가지 요인에 의한 것으로 분석된다.

표 4 폴리곤 수에 따른 평균 비교 횟수

	TIN			DEM		
폴리곤 수	13,346	3,996	972	13,346	3,996	972
평균 비교 횟수	1.66	1.58	1.70	1.36	1.42	1.33

(표4)는 TIN과 DEM 지형에서 폴리곤 수에 따른 인접지역 탐색 알고리즘의 평균 비교 횟수를 나타낸다. 표를 살펴보면 동일한 방식의 지형에서 비교 횟수는 폴리곤 수와는 무관하다는 것을 알 수 있다. 즉 인접지역 탐색 알고리즘은 폴리곤 수와 상관없이 항상 현재 위치하는 폴리곤과 그 주변의 폴리곤만을 대상으로 탐색을 하기 때문에 폴리곤의 수와는 무관한 성능을 보이게 되는 것으로 분석된다. 폴리곤의 수보다는 지형의 특성과 네비게이션 방법에 영향을 받는 것으로 추측된다. Performer와 QOTA는 알고리즘의 특성상 폴리곤 수가 증가할수록 탐색 시간이 증가하는 반면 인접지역 탐색 알고리즘은 폴리곤 수에 커다란 영향을 받지 않는다. 따라서 폴리곤 수가 많아질 수록 Performer와 QOTA에 비해 상대적으로 더 큰 성능향상을 기대할 수 있다.

표 5 폴리곤 수에 따른 프레임 수

폴리곤 수	13,346	3,996	972
평균 프레임 수	11.7	22	33
최소 프레임 수	11	20	25
최대 프레임 수	12.5	25	50

(표5)는 인접지역 탐색 알고리즘에서 폴리곤 수에 따른 초당 프레임 수를 비교한 것이다. 가상환경 참여자가 이동할 때 렌더링 시스템에서는 움직임에 따라 보여지는 화면을 계속해서 렌더링 해 주어야 한다. 이때 자연스러운 화면을 보여주기 위해서는 초당 25~30 이상의 프레임이 필요하다[10]. 인접지역 탐색 알고리즘에서 각각의 폴리곤의 경우 평균 프레임 수가 11.7, 22, 33으로 폴리곤 수 3,996과 972의 경우에는 어느 정도 부드러운 화면을 보여줄 수 있지만, 폴리곤 수 13,346의 경우 부드러운 화면을 위한 요건을 만족시키지는 못한다. 그러나 13,346정도의 지형을 개인 PC에서 작동한다는 점을 고려한다면 자연스러운 화면을 보여주기 위한 최소 프레임수인 10~12 정도의 수준으로 분석될 수 있다[7].

4.2 저장 공간

인접지역 탐색 알고리즘에서는 각 폴리곤마다 인접하고 있는 폴리곤들의 정보를 저장하고 있다. 따라서 기존 QOTA에 비해 저장 공간의 낭비가 발생하게 된다. 인접지역 탐색 알고리즘 각각의 폴리곤 주변에는 평균 15.3개의 폴리곤이 인접해 있는 반면, QOTA의 Quadtree에서 연산에 사용되는 폴리곤 수는 평균 7.2개이다. 즉 8.1개의 폴리곤만큼 저장 공간의 낭비가 발생하게 된다. 이를 수치적으로 계산해 보면 (표6)과 같다[11].

표 6 폴리곤의 저장 공간 크기 계산

	ID	Points	Bounding Box	Neighbors	Etc.
형태	int	float	float	int	int
크기	2bytes	4bytes	4bytes	2bytes	2bytes
전체 크기	1×2=2	9×4=36	4×4=16	2×15.3=30.6	1×2=2

각 폴리곤마다 사용되는 저장 공간의 크기를 계산하면 다음과 같다. 인접지역 탐색 알고리즘에서는 하나의 폴리곤에 평균 86.6bytes(= 2 + 36 + 16 + 30.6 + 2)의 저장 공간이 요구된다. 반면 QOTA에서는 인접지역 탐색 알고리즘에서 사용되는 데이타 구조를 사용한다고 가정했을 경우 하나의 폴리곤에 평균 70.4bytes(= 2 + 36 + 16 + 14.4 + 2)의 저장 공간을 사용한다. 전체 저장 공간 측면에서 살펴보면 인접지역 탐색 알고리즘이 기존 QOTA에 비해 약 23%정도 저장 공간을 낭비하는 결과를 보여준다.

5. 결론 및 향후 연구방향

인접지역 탐색 알고리즘은 저장 공간의 측면에서 볼 때 기존의 방식에 비해 약 23% 정도 낭비하게 발생한다. 반면, 탐색 과정에서 발생하는 비교 횟수를 기존의 방식의 1/10 정도로 줄일 수 있어 실시간 네비게이션시의 탐색 성능을 향상시킬 수 있다. 또한 기존의 방식에서는 가상환경 데이타의 크기가 커짐에 따라 탐색 성능의 저하가 발생하지만, 인접지역 탐색 알고리즘에서는 지형 데이타의 크기에 상관없이 항상 일정한 탐색 시간을 보여주고 있어서 가상환경의 지형 데이타가 커짐에 따라 기존에 방식에 비해 상대적으로 더욱 뛰어난 성능을 나타낼 수 있다.

그러나 기존의 인접지역 탐색 알고리즘에서는 추출 빈도에 따른 폴리곤 건너뜀 현상이 발생할 수 있다. 다시 말해 자전거와 같은 연속적인 움직임의 물체에서는 인접 폴리곤 검색만으로 충분히 탐색이 가능하지만 사람의 걸음걸이와 같은 불연속적인 움직임의 물체에서는 인접 폴리곤을 검색해 나가는 방식에 문제가 발생한다. 물론 이러한 문제에 대해 이미 앞에서 몇 가지 해결책을 제시하고 있지만 완벽한 해결책이라고 할 수는 없다. 따라서 이러한 문제를 해결할 수 있는 개선된 알고리즘이 요구된다.

또한, 앞의 실험결과에서 살펴보았듯이 인접지역 탐색 알고리즘에서는 기존의 방식에 비해 약 23% 정도의 저장 공간을 더 사용하고 있다. 저장 공간의 문제는 탐색 속도에 직접적인 영향을 미치지는 않지만 Terrain Following 준비를 위한 전처리 과정(Pre-processing) 시간에 영향을 미칠 수 있고, 가상환경이 네트웍을 이용한 다중 참여자 환경으로 확대 될 경우 네트웍 부하의 결정적인 원인이 될 수 있다. 이러한 문제를 방지하기 위해서 탐색 시간에는 크게 영향을 미치지 않으면서 저장 공간을 절약할 수 있는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] 이선우, "A Survey on Collision Detection (in Computer Graphics)," http://dangun.kaist.ac.kr/publish/vr.techmemo.html, 1997

[2] John W. Barrus and Richard C. Waters, "QOTA:A Fast, Multi-Purpose Algorithm For Terrain Following in Virtual Environments," VRML '97. Proceedings

of the Second Symposium on Virtual Reality Modeling Language, pp. 59-, 1997
[3] Chris Marrin and Bruce Campbell, *VRML2*, pp. 311-319, sams net, 1997
[4] Mark de Berg, Marc van Kreveld, Mark Overmars and Otfried Schwarzkopf, *Computational Geometry: Algorithms and Applications*, pp. 291-298, Springer, 1998
[5] John W. Barrus and Richard C. Waters, "QOTA:A Fast, Multi-Purpose Algorithm For Terrain Following in Virtual Environments," http://www.stl.nps.navy.mil/vrml97cd/papers/barrus/html/qotafull.htm, 1999
[6] Ellis Horowitz, Sartaj Sahni and Susan Anderson-Freed, *Fundamentals of Data Structures in C*, pp. 191-200, W. H. Freeman and Company, 1996
[7] Kwang-sung Shin and In-mook Lee, "A terrain following simulation considering the dynamic of 2-wheel vehicle," Daewoo Electronics, 1998
[8] Kang Shin-Bong, Joo In-Hak and Choy Yoon-Chul, "Efficient Spatial Query Processing using MMP Multi-filter," Journal of Korean Information Science Society, Vol.24, No.5, pp.476-486, 1997
[9] Richard S. Wright and Michael Sweet, *OpenGL SuperBible*, pp. 31-45, 136-454, Wiith Group, 1996
[10] 최윤철, 한탁돈, 조성배, *인터넷 배움터*, pp. 505-513, 생능출판사, 1998
[11] Jesse Liberty, *Teach Yourself C++ Programming in 21 Days*, pp. 35, SAMS, 1996

김 종 혁
1999년 연세대학교 컴퓨터과학과(학사). 2001년 연세대학교 컴퓨터/산업시스템(석사). 2001년 ~ 현재 해군사관학교 전산과학과 교수요원. 관심분야는 가상환경

최 윤 철
1973년 서울대학교(학사). 1975년 Univ. of Pittsburgh(석사). 1976년 Univ. of California, Berkeley(석사). 1979년 Univ. of California, Berkeley(박사). 1979년 ~ 1982년 Lockheed 사 및 Rockwell International 사 연구원. 1984년 ~ 현재 연세대학교 컴퓨터과학과 교수. 1990년 ~ 1991년 University of Massachusetts 교환교수. 관심분야는 멀티미디어 문서처리(SGML/XML), 가상환경, GIS, Web Based in-struction

고 견
1987년 2월 연세대학교 전산과학과 졸업(이학사). 1989년 2월 연세대학교 수학과(전산전공) 졸업(이학석사). 1993년 9월 일본 동경대학교 공학부 졸업(공학박사). 1995년 3월 ~ 현재 청주대학교 컴퓨터정보공학과 교수. 관심분야는 컴퓨터비전, 그래픽스, 가상현실 등