한국시뮬레이션학회 논문지
Vol. 15, No. 2, pp. 31-39 (2006. 6)

출발점과 목표점을 일반화 가시성그래프로 표현된 맵에 포함하기 위한 빠른 알고리즘

유견아[1†] · 전현주[2]

Fast algorithm for incorporating start and goal points into the map represented in a generalized visibility graph

Kyeonah Yu · Hyunjoo Jeon

ABSTRACT

The visibility graph is a well-known method for efficient path-finding with the minimum search space modelling the game world. The generalized visibility graph is constructed on the expanded obstacle boundaries to eliminate the "wall-hugging" problem which is a major disadvantage of using the visibility graph. The paths generated by the generalized visibility graph are guaranteed to be near optimal and natural-looking. In this paper we propose the method to apply the generalized visibility graph efficiently for game characters who moves among static obstacles between varying start and goal points. Even though the space is minimal once the generalized visibility graph is constructed, the construction itself is time-consuming in checking the intersection between every two links connecting nodes. The idea is that we build the map for static obstacles first and then incorporate start and goal nodes quickly. The incorporation of start and goal nodes is the part that must be executed repeatedly. Therefore we propose to use the rotational plane-sweep algorithm in the computational geometry for incorporating start and goal nodes efficiently. The simulation result shows that the execution time has been improved by 39%-68% according to running times in the game environment with multiple static obstacles.

Key words : Computer game, pathfinding, visibility graph, rotational sweep-line algorithm

요 약

가시성그래프는 최소 탐색 공간으로 게임환경을 모델링하여 효과적으로 길을 찾을 수 있도록 하는 방법으로 잘 알려져 있다. 일반화 가시성그래프는 가시성그래프의 가장 큰 단점으로 지적되는 "벽-껴안기" 문제를 해결하기 위해 확장된 장애물의 경계 위에 생성된 가시성그래프이다. 일반화 가시성그래프에 의해 구해진 경로는 근사 최적이며 자연스럽게 보이는 장점이 있다. 본 논문에서는 변화하는 출발점과 목표점과 정적인 장애물 사이를 움직이는 게임 캐릭터에 효과적으로 일반화 가시성그래프를 적용하는 방법을 제안한다. 일반화 가시성그래프는 일단 생성되면 최소 탐색공간을 보장하지만 그 생성 자체는 노드 사이의 링크의 교차 여부를 일일이 체크하여야 하므로 시간이 많이 소요된다. 아이디어는 먼저 정적인 장애물만으로 지도를 생성해 놓고 출발점과 목표점을 빠르게 포함시키는 것이다. 출발점과 목표점의 포함 부분이 여러 번 반복되어야 하는 과정이므로 출발점과 목표점을 빠르게 포함시키는데에 연산 기하학 분야의 회전 plane-sweep 알고리즘을 이용할 것을 제안한다. 시뮬레이션 결과는 전체 그래프를 매번 생성하는 것보다 제안한 방법의 실행시간이 39%-68% 정도 향상되었음을 보여준다.

주요어 : 컴퓨터 게임, 길찾기, 가시성그래프, 회전 plane-sweep 알고리즘

* 본 연구는 2005년도 덕성여자대학교 자연과학연구소 연구비 지원에 의해 수행되었음.
2006년 3월 20일 접수, 2006년 4월 11일 채택
[1] 덕성여자대학교 컴퓨터공학부 교수
[2] 덕성여자대학교 컴퓨터공학부 석사과정
주 저 자 : 유견아
교신저자 : 유견아
E-mail; kyeonah@duksung.ac.kr

1. 서 론

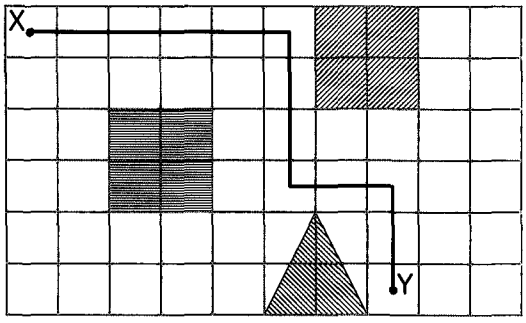
최근 컴퓨터 게임에는 스스로 목표를 향하여 장애물과 충돌하지 않고 움직일 수 있는 NPC(non-player character)들의 사용이 많아지고 있다. 다양한 길찾기 방법들이 제안되어 사용되고 있는데 그중에도 인공지능 분야의 A* 알고리즘이 가장 선호되는 방법이라고 할 수 있다[1,2]. 그러나 이와 같은 탐색 방법의 결정이 길찾기의 전부가 아니다. 실제 경로의 탐색에 앞서서 탐색해야 할 공간을 정의하는 일은 길찾기의 성능을 위해 탐색 알고리즘 못지않게 중요하다고 할 수 있다[3]. 균일 격자, 웨이포인트 그래프, 네비게이션 메쉬 등은 게임에서 많이 사용되는 대표적인 공간 표현 방식들이다. 균일 격자는 가장 공간을 일정한 격자로 나눈 간단한 표현 방식인 반면 큰 탐색공간과 어색한 경로가 단점이다. 웨이포인트 그래프는 축소된 탐색공간을 갖는 반면 최적 경로를 찾지 못한다는 단점이 있다. 네비게이션 메쉬는 최적경로를 보장하지만 엄청나게 많은 수의 다각형을 저장해야 하는 단점이 있다. 이와 같이 탐색공간 표현 방식들은 탐색공간 크기, 최적성, 경로의 질적인 측면에서 장단점을 가지고 있음을 알 수 있는데 탐색공간의 정의에 있어서 또 한 가지 중요한 요인은 자동 생성이 가능한가 하는 것이다. 개발자가 미리 지형을 분석하여 전체 공간을 작은 셀로 분할하거나 적절한 곳에 웨이포인트를 설정해 놓는 기존의 방식은 사용자가 직접 지형을 변화시키거나 동적인 장애물이 다양한 최근 게임에 적용하기에 적절하지 않다.

로보틱스 분야에서 대표적인 로봇 경로계획 알고리즘인 가시성 그래프 (Visibility graph)는 작은 탐색공간으로 인해 탐색 시간을 줄일 수 있고, 설계자의 개입 없이 자동으로 탐색공간을 생성할 수 있으며, 최단 경로를 구한다는 장점 때문에 컴퓨터 게임 분야에서도 많은 관심을 받고 있다[2,4,5]. 최단 경로가 중요한 로봇의 움직임과는 다르게 가상현실 혹은 게임 속 캐릭터의 움직임은 자연스러움이 중요하므로 이를 위해 가시성그래프의 수정된 형태인 VVc[6]나 일반화 가시성그래프의 사용도 제안되었다[7]. 가시성그래프의 노드는 출발점과 목표점, 그리고 장애물의 정점들로 이루어지며 링크는 노드 사이의 서로 보이는 선분들로 이루어진다. 생성된 후의 탐색공간 크기는 작지만 가시성그래프를 생성하는 것은 노드 사이의 연결된 선분이 장애물과 겹치는지 일일이 체크하여야 하기 때문에 시간이 많이 소요되는 작업이므로 이 점이 가시성그래프의 응용이 확대되는 것을 막는 요인 중 하나가 된다. 본 논문에서는 이 문제를 해결하기 위하여 우선 가시성그래프 생성과정을 정적인 장애물을 대상으로 하여 영구적 맵을 만드는 단계와 출발점과 목표점을 맵에 포함시키는 단계로 분리하여 가시성그래프를 이용한 길찾기의 성능 향상을 꾀한다. 즉, 첫 단계는 길찾기가 시작되기 전에 한번만 실행되면 되고 그 이후의 다른 길찾기에서는 반복해서 실행될 필요가 없으며 출발점과 목표점을 포함하는 두 번째 단계가 계속해서 반복 실행되어야 하는, 즉 실시간 실행 효율이 중요한 부분이다. 본 논문에서는 일반화 가시성그래프로 표현되는 맵에 임의의 출발점, 목표점을 신속하게 포함하는 효과적인 알고리즘을 제안한다. 새로 추가되는 노드(출발점 혹은 목표점)와 기존의 노드 사이의 가시 여부(visibility)를 판단하기 위하여 일일이 장애물과의 교차 여부를 계산하는 것이 아니라 연산 기하학 분야의 회전 스윕-선(*rotational sweep-line, RSL*) 알고리즘을 이용하여 선행되는 계산에서 얻어진 정보를 다음 순서의 계산에서 이용함으로서 필요한 계산양의 감소를 꾀한다. 또한 제안된 방식은 가시성그래프뿐만 아니라 웨이포인트 그래프 등의 임계노드 기반 표현 방식들에도 이용될 수 있음을 보여 준다.

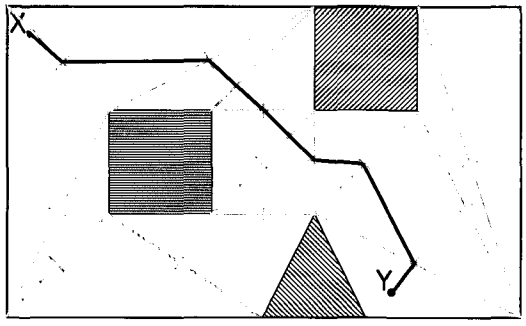
2. 관련연구

게임공간을 표현하는 방식은 크게 셀 분할 방식과 임계 노드 기반 방식, 두 가지로 나누어 볼 수 있다. 셀 분할 방식은 전체 공간을 균일하게 나누는 그리드 기반 방식, 자유공간을 볼록다각형으로 나누는 네비게이션 메쉬 방식등과 같이 공간을 작은 셀로 분할하여 나타내는 방식이며 각 셀이 하나의 노드를 구성하고 이웃한 셀로의 연결이 링크가 된다. 이와 같은 셀 분할 방식에서는 출발점을 포함한 셀을 시작노드로, 목표점을 포함한 셀을 목적노드로 하여 탐색을 시작하고 셀 간의 이동을 연결하여 경로를 표현하게 된다. 즉, 출발점과 목표점을 주어진 맵에 포함하는 일이 특별히 추가적인 작업을 요하지 않는다 (그림 2.1). 물론 이 방식들은 과도한 크기의 탐색공간과 생성된 경로가 자연스럽지 않은 등의 단점으로 계층적으로 분할하여 길찾기를 하며 생성된 경로에 대해 가시성 판별과 같은 후처리를 하는 등의 추가적인 작업이 요구된다[8,9].

반면 전체 공간을 셀로 나누는 것이 아니라 단지 몇몇 중요한 지점에 노드를 설정하는 임계 노드 기반 방식에서는 출발점과 목표점을 얼마나 효과적으로 맵에 포함시키

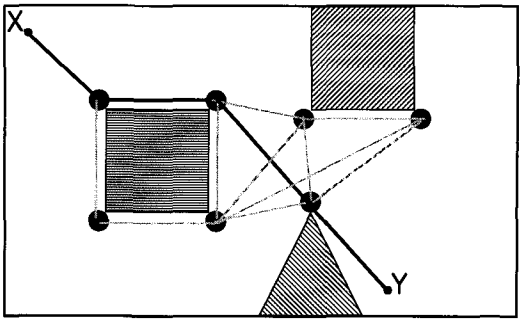


a) 그리드 기반 방식

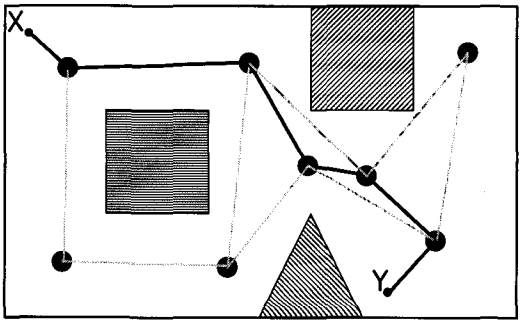


b) 네비게이션 메쉬 방식

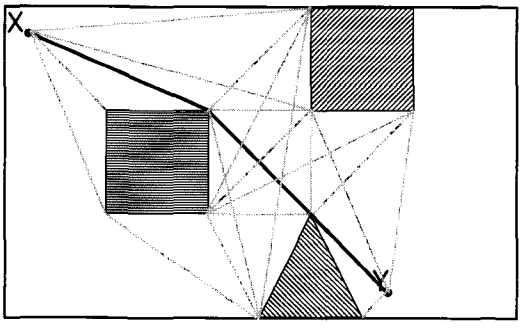
그림 2.1. 게임 공간 표현 - 셀 분할 방식



a) 모서리그래프



b) 웨이포인트 그래프



c) 가시성그래프

그림 2.2. 게임 공간 표현 - 임계 노드 기반 표현 방식

는가 하는 것이 길찾기의 성능을 좌우하게 된다. 대표적인 임계노드 기반 표현 방식으로는 모서리그래프, 웨이포인트 그래프, 가시성그래프 등이 있다. 모서리그래프는 장애물의 모서리에 웨이포인트를 놓고 그 사이를 연결한 링크들로 이루어진 그래프이다 (그림 2.2 a). 즉, 각 볼록 정점들을 노드로 하고 노드를 잇는 선분이 자유공간에 놓여 있으면 링크를 추가한다.

웨이포인트 그래프는 장애물의 정점을 노드로 하는 것이 아니라 장애물사이의 임의의 중간지점에두는 방식이다 (그림 2.2 b). 마찬가지로 노드를 잇는 선분이 자유공간에 있으면 링크가 된다[3]. 가시성그래프는 정점을 노드로 하는 점에서 모서리그래프와 유사하나 노드의 가시여부로 만들어진 가시성그래프는 움직이는 캐릭터의 부피를 고려하여 확장된 개념의 가시성그래프로 정의될 수 있다 (그림 2.2 c)[7].

임계 노드 기반 방식으로 맵이 만들어졌을 경우 캐릭터의 출발점과 목표점을 맵에 포함시키는 간단한 방법으로는 가장 가까운 노드로 연결하는 것이다[3]. 그림의 경로들은 이와 같은 방법으로 출발점 X와 목표점 Y를 포함한 후, 경로를 계획한 것이다. 출발점과 목표점을 포함하는 가장 간단한 방법이긴 하지만 최적경로를 구하는데 실패하는 요인이 됨을 볼 수 있다. 이를 해결하기 위하여 "string-pulling"에 의해 가장 가까운 노드가 아니라도 최적에 근사한 직선 경로를 찾아내는 방안을 제안하였다[3]. 그렇지만 이 방식은 일일이 선분이 장애물과 교차하는지를 테스트해야 하기 때문에 계산상의 부담이 가중되는 단점이 있다.

또한 최적성을 보장하기 위한 방안으로는 가시성그래프의 본래 정의에 입각하여 출발점과 목표점을 포함시켜

전체 맵을 생성하는 것이다. 그렇지만 이 방법은 매번 전체 맵을 계산하여야 하므로 실행의 효율을 떨어뜨린다. 그러므로 본 연구에서는 정적인 장애물로 이루어진 맵을 일단 생성하고 출발점과 목표점이 주어지는 데에 따라 회전 sweep-line 알고리즘을 이용하여 효과적으로 이들을 포함하여 경로를 계획하는 방법을 제안한다. 가시성그래프를 기본 탐색 방식으로 하여 제안하지만 제안된 방식은 다른 임계 노드 기반 방식에도 적용될 수 있다.

3. 회전 Sweep-line (RSL) 알고리즘을 이용한 노드의 삽입

정적인 장애물로 이루어진 맵이 있을 때, 캐릭터의 출발점과 목표점이 주어지면 빠르게 맵에 포함되어 최적 경로를 찾을 수 있어야 한다. 이를 위해 RSL 알고리즘을 이용한다. 임의의 점 p와 장애물의 임의의 정점 w가 가시성을 갖는지를 알기 위해 선분 pw와 장애물의 변이 교차하는지 파악해야 한다. 장애물의 정점의 수가 O(n)이라면 선분 pw에 대하여 장애물의 모든 변과의 교차여부를 확인해야하므로 $O(n^2)$의 계산이 필요하다. 그러나 이러한 계산을 순환적(cyclic) 순서로 수행하면, 수행할 때 얻어지는 정보는 다음 순서에서의 가시성을 결정할 때 도움이 되어 O(nlogn)에 가능하다는 것이 *RSL* 알고리즘을 적용하고자 하는 이유이다. 이 장에서는 우선[10]에 소개된 *RSL* 알고리즘을 살펴보고 이를 응용하여 컴퓨터 게임 환경에 적용하는 방법을 소개한다.

3.1 회전 Sweep-line(RSL) 알고리즘

RSL 알고리즘은 plane-sweep 패러다임과 비슷한 접근으로 차이점은 sweep-line을 수직/수평선으로 이동하면서 스위핑하는 것이 아니라 임의의 점을 중심으로 반직선(half-line) ρ를 (0, 2π]로 회전시키며 스위핑한다는 것이다. 처리가 일어나는 중요 이벤트는 장애물의 정점들이다. 정점들을 시계방향 순서로 정렬을 하고 시작점을 p로 하고 w를 지나는 반직선 ρ를 x의 양의 축 방향, 즉 0°에서부터 스위핑한다. 반직선 ρ가 임의의 장애물의 변과 교차하면 교차하는 변을 이진탐색트리(binary search tree)에 저장한다. 즉 이진탐색트리의 리프노드에는 ρ와 교차하는 순서대로 변이 저장되고 내부노드에는 자신의 왼쪽 서브트리의 가장 오른쪽 노드의 변이 저장된다. 즉, 노드의 오른쪽 서브트리에 있는 변들은 ρ를 따라 모두 큰 쪽에 있는 변들이며 왼쪽 서브트리에 있는 변들은 작은 쪽에 있는 변들이다. 그림 3.1은 초기 상태를 나타내며 (0, 2π]의 회전 방향으로 정렬된 정점 리스트 W는 그림 3.1에 있다.

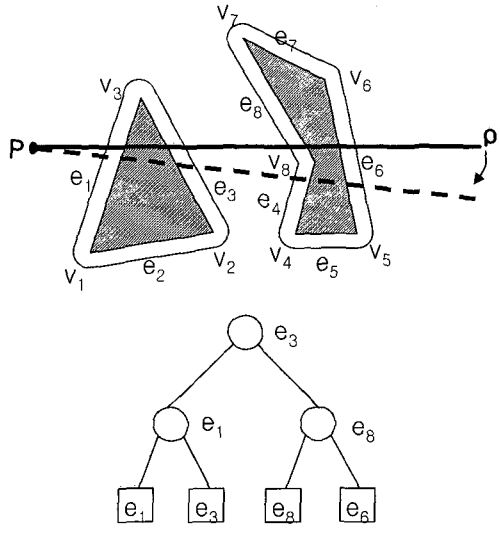


그림 3.1. RSL 알고리즘의 초기상태와 이진탐색트리에 저장된 교차선분

p와 W 리스트의 정점 v_i간의 가시성 여부는 이진트리를 이용하여 O(1) 시간복잡도로 결정된다. 즉, 선분 pv_i를 이진트리의 가장 왼쪽 노드의 변 e1과 교차여부를 체크하여 교차하면 가시성이 없는 것이고 교차하지 않으면 가시성 있는 것이다. 중요한 것은 각 이벤트에 대해 이진트리를 다음과 같이 적절히 수정하여 유지하는 것이다: 정점을 기준으로 시계 방향에 있는 변은 이진트리에 삽입하고, 시계반대방향에 있는 변은 삭제한다. 즉 W의 첫 번째 이벤트인 v_8의 경우에는 pv_8이 e_1과 교차하므로 가시성이 없는 노드로 판명되며 v_8 이벤트에 의해 e_8은 이진트리로부터 삭제되고 e_4가 삽입된다. 그 다음 이벤트인 v_5를 만나면 e_6가 삭제되고 e_5가 삽입되며 이와 같은 방식으로 W에 있는 정점 모두에 대해 차례로 실행하여 p로부터 가시성을 갖는 노드의 집합을 반환한다.

장애물의 정점의 수를 O(n)이라 할 때, 초기 이진트리를 생성하기 위해 O(nlogn), 각 정점에 대해 트리의 맨 왼쪽 선분과의 교차 여부를 확인하기 위해 O(logn), 이진트리로부터 노드의 삭제 연산이나 이진트리로의 삽입 연산을 통하여 이진트리의 상태를 유지하기 위해 각각 O(logn) 등의 시간복잡도가 소요되어 전체적인 시간복잡도

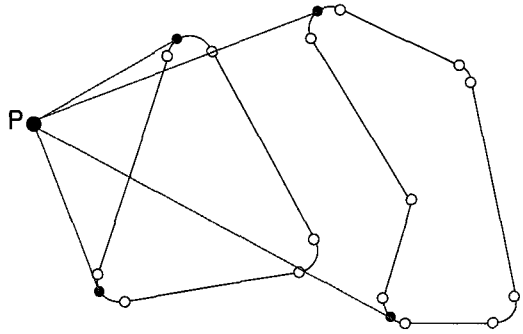


그림 3.2. 지역적 비볼록 일반화다각형에 대한 RSL 알고리즘의 적용

는 O(nlogn)으로 기존의 $O(n^2)$의 알고리즘과 비교해 효율적임을 알 수 있다. 필요한 공간은 O(n)이다.

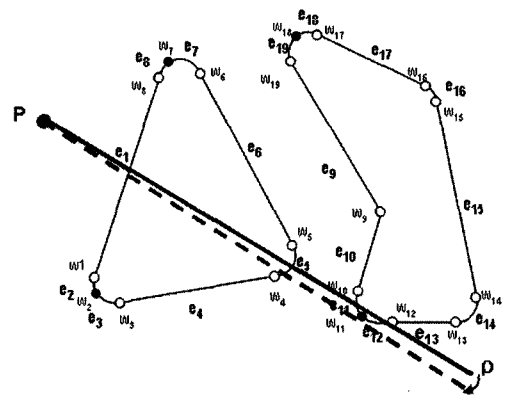
3.2 컴퓨터 게임 환경에서 RSL 알고리즘의 응용

컴퓨터 게임에서 임계 노드 기반 방식으로 게임 공간을 표현하는 방법 중에 가시성 그래프를 이용하는 경우와 설계자가 임의로 임계 노드를 설정하는 웨이포인트 방식, 두 가지의 경우에 대해 RSL 알고리즘을 응용해 본다.

3.2.1 일반화 가시성 그래프에서의 RSL 알고리즘의 응용

컴퓨터 게임에서 길찾기를 위해 가시성그래프를 이용할 때, 캐릭터의 부피를 고려하지 않아 캐릭터가 장애물과 부딪히거나 캐릭터가 장애물의 모서리에 밀착하여 움직이는 것처럼 보이는 단점이 있다. 이를 해결하기 위하여 장애물의 모서리를 원래보다 r(움직이는 캐릭터를 포함하는 원의 반지름)만큼 오프세팅하여 적용하는 방법이 이용된다[7]. 이렇게 하면 다각형으로 모델링되었던 장애물들의 볼록 정점들은 원의 호로 변환되고 오목 정점들은 다른 오목 정점으로 변환되어 지역적인 비볼록 (locally non-convex) 일반화다각형이 된다[11]. 즉, 그림 3.2와 같이 볼록 호와 오목 정점으로 이루어지는데 이들 중, 점 p와 호 사이의 가시성을 확인해야 한다. 오목 정점들은 가시성그래프의 정의에 따라 최단 경로를 구하는 데에는 사용되지 않으므로 가시성 여부는 따져주지 않아도 된다.

선분은 그의 끝점에 대해 sweep 방향으로 시계 방향, 혹은 시계 반대방향에 있는 것을 판별할 수 있으나 호는 sweep 방향에 대해 단순 증가, 감소하지 않아 판별이 불가능하므로 이를 위해 호 위에 가상점(fictitious point)을 생성해야 한다. 이 가상점은 호를 sweep 방향으로 단순



$W=\{w_9, w_{14}, w_5, w_{13}, w_{10}, w_{12}, w_{11}, w_4, w_3, w_1, w_2, w_7, w_8, w_6, w_{18}, w_{17}, w_{19}, w_{16}, w_{15}\}$

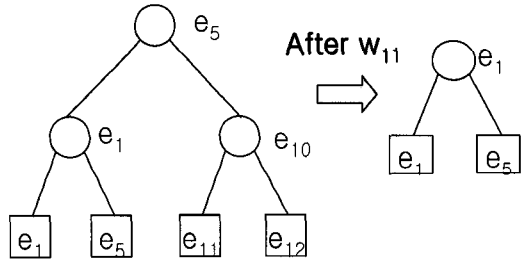


그림 3.3. 지역적 비볼록 일반화다각형에 대한 RSL 알고리즘의 적용

감소 혹은 증가하는 성질을 가지도록 하는 것이므로 접선에 의해 생성된다. 그림 3.2의 예에서는 4개의 가상점이 생성되는 것을 보여준다. 흰 점들이 기존의 정점들이며 까맣게 표시된 점들이 접선에 의해 새로 생긴 가상점들이다. 이와 같이 생성된 가상점들은 그림 3.3과 같이 기존의 정점들과 함께 이벤트 리스트를 구성하게 되며 하나였던 호는 가상점을 중심으로 두개의 세그먼트로 분할된다. 즉, 그림 3.3에서 w_2, w_7, w_{11}, w_{18}은 가상점들이며 나머지는 기존의 정점들이다. 이 점들을 x축의 양의 방향을 시작으로 시계 방향으로 정렬한 이벤트 리스트 W가 생성되면 다각형에 대해 설명한 RSL 알고리즘과 유사하게 진행되는데 이를 정리하면 다음과 같다.

일반화 다각형 환경에서의 RSL 알고리즘

1 단계 : 점 p로부터의 반직선에 접하는 호 위의 가상점들을 모두 생성하고 기존의 끝점들과 가상점을 시계 방향으로 정렬한 리스트 W를 생성한다. 이 때 동일 선상에 여러 점이 있는 경우-

에는 점 p로부터 떨어진 순서대로 정렬한다.

2 단계 : 점 p로부터 양의 x축 방향의 반직선 ρ와 교차하는 선분 혹은 호 세그먼트들로 이진탐색트리를 생성한다.

3 단계 : W의 첫 번째 이벤트 w가 가상점이면 <4 단계>로, 아니면 즉, 원래의 끝점들이면 <5 단계>로 진행한다.

4 단계 : 선분 pw와 이진트리의 맨 왼쪽 노드와의 교차를 체크하여 교차하지 않으면 w는 점 p로부터 가시성 있는 것이므로 L에 넣는다.

5 단계 : w의 시계 방향에 있는 세그먼트들은 이진트리에 삽입하고 시계 반대방향에 있는 세그먼트들은 이진트리로부터 삭제한다.

6 단계 : w를 W로부터 제거하고 W가 공집합이 아니면 <3 단계>로 되돌아가며 공집합이면 L을 리턴하고 종료한다.

<1, 2단계>는 초기화의 단계이다. 앞에서 설명한 가상점을 생성하는 단계가 알고리즘의 도입부분에 추가되며 일단 가상점이 생성되면 기존의 끝점들과 함께 정렬된다. <3 단계>에서 주목할 것은 W의 이벤트들 가운데 가상점에 대해서만 가시성 여부를 확인한다는 것이다. 이는 앞에서 언급한 바와 같이 가시성그래프를 이용한 경로계획에서는 bitangent한 링크만이 최단 거리를 구성하므로 w_9와 같은 오목 정점이나 e_5, e_{14}, e_{16}등과 같이 p로부터의 접점을 생성하지 않는 호들은 가시성 여부와 관계없이 그래프의 링크를 생성하지 않는 것이다.

<4 단계>는 w가 점 p에 대하여 가시성이 있는가를 결정하는 단계로서 pw가 이진트리의 가장 왼쪽 노드의 선분과 교차하는지만 판별하면 충분하다. 교차하는 경우는 가시성이 없는 경우이고, 교차하지 않으면 가시성이 있는 경우이다. 여러 점들이 동일 직선상에 존재하는 경우와 같이 특수한 경우(degenerate case)를 처리하기 위해 현재 노드 앞에 이미 처리된 동일 직선상의 노드가 있는지 확인하고 그 정보를 이용한다[10].

<5 단계>는 RSL 알고리즘의 상태 유지를 위한 부분이다. 이진트리로부터의 삭제는 해당하는 리프노드를 삭제한 후, 내부 노드를 그에 따라 수정해 주면 된다. 이진트리로의 삽입의 경우에는 삽입되어야 하는 장애물 변과 내부 노드의 장애물 변을 비교하여 교차하면 왼쪽 서브트리로, 교차하지 않으면 오른쪽 서브트리로 진행한다. 이와 같이 하면 삭제/삽입 연산은 각각 O(logn)의 시간복잡도로 실행된다.

이진트리의 상태 변화는 그림 3.3에 나타나 있다. 반직선 ρ가 진행되면서 가상점 w_{11}을 이벤트로 하는 전후의 상황을 이진트리의 변화로 보여준다. 선분 pw_{11}이 이진트리의 맨 좌측 노드의 변인 e_1과 교차하므로 이 가상점 w_{11}은 가시성이 없다고 판별된다. 시간복잡도에 대한 분석은 지역적 비볼록 일반화다각형에 대해서 RSL 알고리즘을 적용할 때에는 호에 대해 가상점을 생성하여 이벤트 리스트가 커지고 이에 따라 세그먼트의 수도 증가하지만 장애물의 정점의 수를 O(n)이라 할 때, 그 수가 모두 상수배만큼 증가하여 전체 시간복잡도에는 영향을 미치지 않게 된다. 한 점에서 반원으로의 접선을 최대 2개까지 가능하므로 이벤트 리스트의 크기는 기존의 리스트의 최대 2배가 되며 이로 인해 증가하는 세그먼트의 수도 3배까지 가능하기 때문이다. 이에 따라 초기 이진트리 생성에 O(nlogn), 각 이벤트마다 이진트리 상태 유지를 위해 O(nlogn), 선분 pw의 교차 여부 확인을 위해 이진트리를 탐색하는데에 O(nlogn)이 소요되어 전체 시간복잡도는 O(nlogn)으로 변함이 없다.

3.2.2 웨이포인트 그래프에서의 RSL 알고리즘의 응용

컴퓨터 게임에서 게임 공간을 표현하기 위해 많이 이용하고 있는 모서리그래프, 웨이포인트 그래프, 원형 기반 웨이포인트 그래프는 모두 레벨 설계자가 임의로 자유공간에 임계 노드를 설정해 놓는 방식들이다. 게임 세계를 구성하는 장애물은 그대로 두고 설계자가 설정한 웨이포인트가 그래프의 노드가 되는 것이다[12]. 이와 같이 구성된 그래프에 출발점과 목표점을 추가하려면 각 점에서 장애물과 교차하지 않는 가장 가까운 노드에 연결하야 한다. 그림 2.2에서 보여준 예제 그림들이 이와 같은 방식으로 찾아진 경로들이다. 간단해 보이는 방법이지만 시간복잡도 측면에서는 모든 웨이포인트가 추가되는 점에 연결될 후보들이며 각각에 대해 장애물과 교차하는지를 판별해야 하므로 웨이포인트 노드를 m, 장애물의 정점의 수를 n이라고 하면 시간복잡도는 O(nm)이다. 모서리그래프의 경우, 웨이포인트를 장애물의 정점에 위치시키므로 웨이포인트의 수는 장애물의 정점 수와 일치한다. 그러므로 시간복잡도는 $O(n^2)$이 된다. 시간복잡도의 측면에서 뿐 아니라 모든 웨이포인트 기반 표현 방식들은 구해진 경로가 최적의 답과는 거리가 있다는 공통적인 문제점을 가지고 있다. 웨이포인트 방식에 의해 맵을 표현하는 게임에서도 출발점과 목표점을 맵에 연결하는 방식을 RSL 알고리즘으로 한다면 기존의 경로보다 계산적인 측면에서도 향상되고 질적인 면에서도 나아질 수 있다.

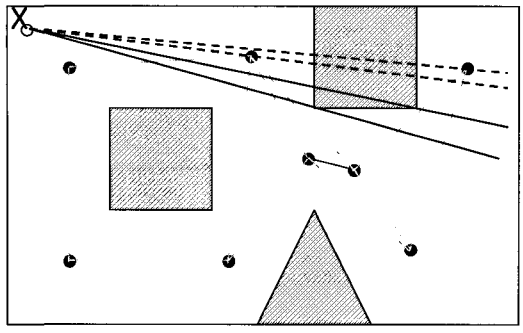


그림 3.4. 웨이포인트 그래프에서의 RSL 알고리즘- 웨이포인트 노드(점선)와 장애물 정점(실선)의 두 종류의 이벤트가 있다.

웨이포인트 그래프에 출발점을 연결하기 위해 RSL 알고리즘을 적용하면 우선 이벤트 리스트의 구성이 장애물의 정점과 웨이포인트 노드로 이원화된다. 즉 그림 3.4에서와 같이 출발점 X를 중심으로 실선으로 표현된 반직선들은 장애물의 정점 이벤트들이며 점선으로 표현된 반직선들은 웨이포인트 노드 이벤트들이다. RSL 알고리즘은 이들 이벤트를 정렬한 리스트 W와 장애물의 변들로 구성된 초기 이진탐색트리로 시작하여 3.1절에서 소개된 기존의 방식과 같이 진행된다. 단, w가 어떤 이벤트인가에 따라 처리를 달리한다. w가 장애물의 정점이면 스위핑의 상태를 이진탐색트리를 통해 갱신하고 w가 웨이포인트 노드이면 w가 출발점에서 visible한가를 이진탐색트리를 이용해 판별한다. 이벤트 정렬 리스트의 크기는 커지지만 초기 이진트리 생성은 장애물의 변에 대해서만 하므로 O(nlogn), 각 장애물 정점 이벤트마다 이진트리 상태 유지를 위해 O(nlogn), 웨이포인트 노드 w에 대해 선분 pw의 교차 여부 확인을 위해 이진트리를 탐색하는데에 O(mlogn)이 소요되어 전체 시간복잡도는 O((n+m)logn)이 된다.

그림 3.4. 웨이포인트 그래프에서의 RSL 알고리즘- 웨이포인트 노드(점선)와 장애물 정점(실선)의 두 종류의 이벤트가 있다.

4. 구현 및 결과

일반화 가시성그래프로 만들어진 맵에 출발점 혹은 목표점을 추가하는 RSL 알고리즘의 구현은 그림 4.1의 흐름도와 같다. 추가되는 노드를 P라고 할 때, 흐름도 상의 VISIBLE(w_i)라는 함수는 기존의 노드 w_i가 점 P로부터

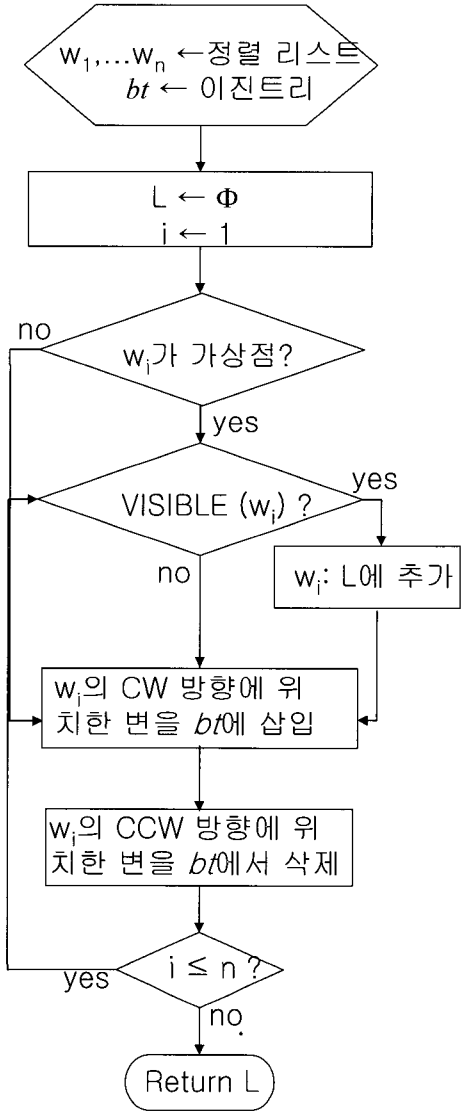


그림 4.1. 일반화 다각형을 위한 RSL 알고리즘의 흐름도

가시성이 있는가를 판별하는 부프로그램이다. VISIBLE (w_i)는 장애물 배치의 여러 경우에 따라 선분 pw_i가 장애물의 선분 또는 호와 교차하는지를 이진탐색트리 검색을 통하여 계산한다. VISIBLE(w_i)함수의 결과에 따라 p와 링크 생성이 가능한 노드의 집합을 반환하고 이진탐색트리의 정보를 갱신한다.

제안과 같이 길찾기 할 때마다 매번 전체 맵을 계산하지 않고 주어진 출발점과 목표점을 효율적인 알고리즘에 의해 삽입하는 방법의 이론적인 우월성은 3장에서 시간복

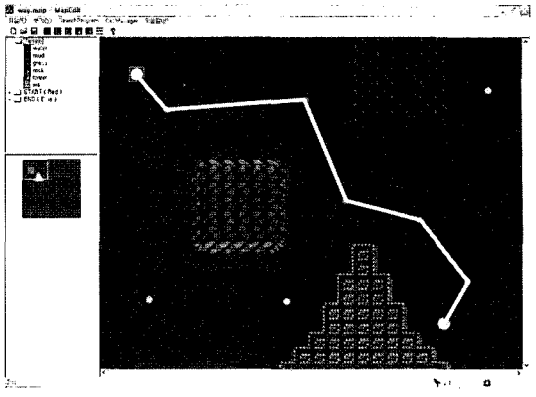
표 4.1. 기존의 일반화 가시성그래프 대비 제안된 방식의 실행시간 개선도 백분율

꼭지점수	실행회수			
	1	3	5	7
8	100.00	76.92	62.50	55.56
14	103.03	66.13	47.78	35.38
20	103.51	64.29	47.10	34.36
32	105.56	63.04	46.04	32.65
평균	103.03	67.60	50.85	39.49

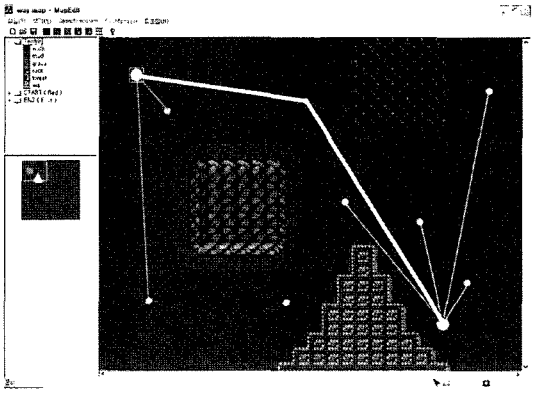
잡도에 대한 분석으로 확인한 바 있다. 이에 대해 게임을 진행하면서 실제로 얻을 수 있는 정량적인 효과를 시뮬레이션을 통해 확인하였다. 시뮬레이션 방법은 한 게임 환경에 대해 NPC가 이동할 필요가 있을 때마다 기존의 방식대로 일반화 가시성 그래프를 생성하여 길찾기 하는 경우와 정적인 장애물에 대해 일단 맵을 만들어 놓고 NPC가 이동할 때마다 출발점과 목표점을 삽입하며 길찾기 하는 경우의 실행 시간을 비교하였다. 이와 동일한 실험을 게임 배경의 복잡도를 n=8,14,20,32로 증가시키면서 반복하였다. 표 4.1의 결과를 보면 1회 실행시에는 제안한 방법보다 출발점, 목표점을 포함하여 일반화 가시성그래프를 생성하는 것이 약간 더 효율적인 것을 볼 수 있는데 이는 제안된 방식이 전체 과정을 둘로 나누어 실행하기 때문에 생긴 오버헤드에 기인한 것이다.

하지만 실행 횟수가 증가함에 따라 실행시간의 개선도가 점점 좋아지는 것을 알 수 있다. 보통 컴퓨터 게임에서 NPC가 한번 움직이고 마는 경우는 매우 드물며 대부분의 경우에 동일한 환경에서 다양한 위치 이동을 하게 되므로 제안된 방식에 따른 성능 향상은 자명하다고 볼 수 있다. 같은 실험을 게임 환경의 복잡도를 증가시켜 가면서(장애물 정점의 수를 증가시키면서) 반복하였을 때에는 복잡해질수록 실행시간 개선도가 조금 나아지는 경향을 보였다.

기존의 웨이포인트 그래프에 의한 경로와 제안한 방법으로 출발점과 목표점을 연결한 경우의 경로가 그림 4.2에 비교되어 있다. 두 점을 그래프에 추가할 때 가시성 여부를 체크하여 링크를 생성하고 이를 이용해 길찾기를 하면 기존의 웨이포인트를 그대로 이용함에도 불구하고 질적으로 향상된 경로를 찾아낼수 있었다.



(a)



(b)

그림 4.2. 웨이포인트 그래프에서 기존의 방식대로 찾은 경로(a)와 가시성 있는 링크를 이용한 경로(b)의 비교

5. 결 론

본 논문에서는 일반화 가시성그래프를 이용한 길찾기 보다 효과적으로 하기 위하여 정적인 장애물에 대한 맵의 생성과 변하는 출발점과 목표점의 포함 과정을 분리하였으며, 반복해서 시행하게 되는 후자의 과정을 RSL 알고리즘을 이용하여 빠르게 실행하는 방법을 제안하였다. 연산기하학 분야의 RSL 알고리즘은 원래 다각형 환경에서 고안된 알고리즘으로 본 논문에서는 일반화다각형 환경으로 확장하기 위하여 기존의 알고리즘을 수정하였다. 이와 같은 방법으로 길찾기 한 시뮬레이션 결과는 다양한 환경에서 실행 횟수가 거듭될수록 실행시간 개선도가 평균 103%> 68%> 51%>39%로 향상되었음을 보여주었다. 뿐만 아니라 일반화 가시성그래프 이외의 임계노드기반 게임공간 표현 방식의 하나인 웨이포인트 그래프에도 적용될 수 있었으며 이와 같이 하면 기존 방식의 문제점이던 지그재그형이면서 최적과는 거리가 멀었던 경로의 질적인 측면을 상당 부분 향상시킬 수 있음을 보

여 주었다.

정적 장애물로 구성된 맵에 새로이 노드를 추가하는 이 방식을 확장하면 임의의 동적 장애물에 대해서도 효과적으로 처리 할 수 있는 가능성이 있다. 동적 장애물이 다각형으로 표현되어 있다면 다각형은 정점의 집합으로 표현할 수 있으며 결국 정점 여러 개에 대해 논문에서 제안한 방법을 적용하면 되는 것이다. 그러나 본 논문에서 채택한 일반화 가시성그래프에서는 장애물을 정점의 집합 그대로 사용하는 것이 아니라 확장하여 원의 호를 포함하게 되므로 일반화 다각형 환경에 다시 일반화 다각형을 포함하는 문제를 해결하기 위해서는 더 많은 연구가 필요하다.

참 고 문 헌

1. B. Stout, "Smart moves: Intelligent path-finding", Game developer magazine, October:28-35, 1996.
2. S. Woodcock, "Game AI: The state of the industry", Game Developer Magazine, August, 2000.
3. P. Tozour, "Search Space Representations", AI Game Programming Wisdom 2, pp85-102, Edited by Steve Rabin, Charles Rive Media, 2004.
4. S. Rabin, "A* Aesthtic Optimizations", Game Programming Gems, pp264-271, Charles River Media, 2000.
5. T. Young, "Optimizing Points-of-Visibility Pathfinding", Game Programming Gems 2, pp. 324-329, Charles Rive Media, 2001.
6. R. Wein, J.P.van der Berg, and D. Halperin, "The Visibility-Voronoi Complex and Its Applications" Proc. European Workshop on Computational Geometry pp. 151-154, 2005.
7. 유견아, 전현주, "컴퓨터 게임환경에서 일반화 가시성그래프를 이용한 경로찾기", 한국시뮬레이션학회 논문지 14(3), 2005.
8. A. Botea, M. Muller, J. Schaeffer, "Near optimal hierarchical path-finding", Journal of game development, vol 1(1), pp 1-22, 2004.
9. D. Jung, H. Kim, J. Kim, K. Um, H. Cho, "Efficient Path Finding in 3D Games by using Visibility Tests with the A* Algorithm", Proceedings of the International Conference Artificial Intelligence and Soft Computing, Spain, September 2004.
10. M. de Berg et al., Computational Geometry :Algorithms and Applications 2nd edition, Springer Verlag, 2000.
11. J.C. Latombe, "Robot Motion Planning", Kluwer Academic Publishers, 1991.
12. L. Liden, "Strategic and Tactical Reasoning with Waypoints", AI Game Programming Wisdom, pp211-220, Edited by Steve Rabin, Charles Rive Media, 2002.

유 견 아 (kyeonah@duksung.ac.kr)

1986 서울대학교 공과대학 제어계측공학과 학사
1988 서울대학교 공과대학 제어계측공학과 석사
1995 미국 USC Computer Science 박사
현재 덕성여자대학교 컴퓨터공학부 교수

관심분야 : 인공지능, 로봇 알고리즘, 연산 기하학

전 현 주 (hzoo@duksung.ac.kr)

2004 덕성여자대학교 지연과학대학 컴퓨터과학부 학사
현재 덕성여자대학교 전산 및 정보통신대학원 석사과정

관심분야 : 인공지능, 게임 프로그래밍