Journal of Korea Multimedia Society Vol. 11, No. 5, May 2008(pp. 692-702)

컴퓨터 게임에서의 경로 계획을 위한 캐릭터별 로드맵의 자동 생성

유 견 아†

요 약

게임이나 가상현실공간에서의 경로 계획은 자율적으로 움직이는 캐릭터들의 수가 많아짐에 따라 그 비중이 높아지고 있다. 로드맵 방식이란 이동 가능한 경로를 나타내는 지도를 사전에 제작하여 경로를 계획하는데 이용하는 방법을 말하며 높은 품질의 경로를 제공할 수 있다는 장점이 있다. 그렇지만 로드맵이 한번 정해지면 모든 캐릭터가 하나의 지도 위에서 움직이기 때문에 크기가 다른 캐릭터들의 특성이 반영되지 못하는 단점이 있다. 본 논문에서는 캐릭터별로 자신의 크기에 맞는 로드맵을 가지고 경로 계획에 각자 이용할 수 있는 효율적인 방법을 제안한다. 이 방법은 캐릭터의 수에 관계없이 전체 지도는 가시성 그래프를 응용하여 1회만 생성하고, 캐릭터의 크기에 따라 이동 가능한 경로를 점진적으로 추가하는 방식으로 제안되기 때문에 효율적이다. 시뮬레이션을 통해 개별 로드맵 방식으로 얻을 수 있는 효과를 보여주며 이때 수반되는 트레이드오프를 분석한다.

Automatic Generation of Character-Specific Roadmaps for Path Planning in Computer Games

Kyeonah Yu†

ABSTRACT

Path planning is gaining more weight in computer games and virtual reality as the number of self-moving characters increases. In the roadmap approach, the map of possible paths is built in advance to plan paths for a character, whose advantage is to provide high-quality paths. On the other hand, a disadvantage is that the road map doesn't reflect properties of characters such as their sizes because they move on the same map once the road map is constructed. In this paper we propose an efficient method to build a different road map for each character so that it can use its own map for path-planning. This method is efficient because the whole map is built once by applying the Visibility Graph regardless of the number of characters and walkable paths are incrementally inserted according to the sizes of characters. The effects of using separate roadmaps are demonstrated through simulations and the trade-offs accompanied with these effects are analyzed.

Key words: Computer game(컴퓨터 게임), Path planning(경로 계획), Roadmap(로드맵), Visibility graph(가시성 그래프), Voronoi link(보로노이 링크)

※ 교신저자(Corresponding Author) : 유견아, 주소 : 서울시 도봉구 쌍문동 419(132-714), 전화 : 02)901-8346, FAX : 02)901-8341, E-mail : kyeonah@duksung.ac.kr
접수일 : 2007년 11월 15일, 완료일 : 2008년 1월 31일

† 정회원, 덕성여자대학교 컴퓨터공학부 교수
※ 본 연구는 덕성여자대학교 2007년도 교내연구비 지원에 의해 수행되었음

1. 서 론

최근 주목을 받고 있는 Second Life 등의 가상현실 공간이나 RPG(Role-playing game)등과 같은 컴퓨터 게임 분야에서는 사용자가 조작하지 않고 자율적으로 움직이는 캐릭터의 사용이 늘고 있다. 이와 같은 캐릭터를 컴퓨터 게임이나 가상현실에서 자유롭게 움직일 수 있도록 하기 위해서 경로를 계획하는 일은 필수 요소가 되었다. 좋은 품질의 경로를 생성하기 위해 게임마다 적절한 로드맵(roadmap)을 사전에 제작하여 사용하는데 대표적인 로드맵 생성 방법으로는 경로점 그래프(waypoint graph), 가시성 그래프(Visibility Graph)나 보로노이 다이어그램(Voronoi diagram) 등이 있다. 경로점 그래프는 레벨 설계자가 캐릭터가 이동할 수 있는 몇 개의 경로점을 미리 정의하여 그 점만을 이용하여 이동하게 하는 방식이며[1] 후자의 두 가지 방식은 로보틱스 분야에서 로봇 경로 계획을 위해 사용되었던 방법이 게임에 응용된 것이다[2-4]. 일단 게임을 위한 로드맵이 만들어지면 게임에 등장하는 캐릭터들의 경로는 그 로드맵 위에서 생성되게 된다. 이는 게임에 등장하는 캐릭터들이 게임을 통해 같은 방식으로 움직인다는 전제하에 사용되는 방법이며 실제로 여러 캐릭터들의 네비게이션 특성이 다양하다는 점은 간과된 것이다. 간단한 예로 토끼가 지나간 좁은 길을 소나 말 등의 캐릭터가 지나갈 수 없는 경우를 들 수 있다. 본 논문에서는 이를 해결하기 위해 캐릭터별로 자신에 맞는 로드맵을 가지고 경로 계획에 이용하도록 제안하며 이를 위한 효율적인 방법을 소개한다.

컴퓨터 게임에서 가장 많이 사용되고 있는 경로점 그래프 방식은 레벨 설계자가 장애물이 없는 지역에 캐릭터가 지나갈 길은 미리 선정하고 캐릭터들은 그 위를 따라서만 움직일 수 있도록 제한한 방식이다. 이 방식이 가장 보편적으로 사용되는 이유는 따로 계산이 필요 없어 빠르고, 실시간 움직임에 가장 적합하다는 것이다. 최근에는 기술의 발달로 자원의 제약에 완화되고 있기 때문에 복잡한 계산 과정이 필요한 로봇 경로 계획 방식들도 컴퓨터 게임 분야에서 관심을 받기 시작하였다[3,5]. 그 중 하나인 가시성 그래프는 장애물의 꼭지점을 노드로 하고 노드들을 연결한 선분 중에서 서로 보이는 것들을 링크로 한 그래프이다. 이들 링크를 이용하여 경로를 생성하기 때문에 최단 경로를 찾는 것은 보장되어 있으나 캐릭터가 장애물에 붙어서 이동하는 단점이 있다. 반면 보로노이 다이어그램은 장애물 사이의 중간점들을 이어서 생성된 그래프로 장애물로부터 가장 많은 여유 공간을 확보하며 이동할 수 있지만 경로가 길고 비연속 점이 자주 발생하는 단점이 있다. 가시성 그래프와 보로노이 다이어그램의 장점을 취한 VV-복합체(VisibilityVoronoi-complex) 구조가 제안된 바 있다[6]. VV-복합체는 확대된 장애물과 보로노이 다이어그램의 교차부분을 합쳐서 일반화 가시성 그래프를 생성하는 방식으로 만들어진 새로운 형태의 다이어그램을 말한다.

본 논문에서는 캐릭터마다 다른 로드맵을 생성하기 위해 가시성 그래프를 근간 로드맵으로 하고 보로노이 다이어그램을 보조적으로 이용하는 방식을 제안한다. 게임에서 가장 많이 이용되고 있는 경로점 그래프를 이용하지 않은 이유는 설계자가 수동으로 경로점을 선정해야 하므로 캐릭터별로 일일이 로드맵을 생성하기에는 적합하지 않기 때문이다. 그러나 가시성 그래프는 자동으로 생성되는 효과적인 알고리즘들이 많이 제안되어 있을 뿐 아니라 최단 경로를 찾을 수 있다는 큰 장점이 있다. 장애물의 경계를 따라 이동하는 단점은 장애물을 실제보다 확장하여 가시성 그래프를 생성하여 해결하고자 한다[7]. 장애물 확장을 통해 기존에 확보되어 있던 자유공간이 막히는 경우에는 보로노이 모서리를 삽입함으로서 자신의 크기에 맞는 경로를 확보할 수 있도록 한다. 제안된 방식은 VV-복합체의 기본 개념과 유사하지만 전체 다이어그램을 생성하지 않고 확대된 장애물이 겹치는 부분에 대해 보로노이 모서리를 첨가하는 방법으로 불필요한 계산을 생략하여 더욱 효과적이다. 또한 캐릭터의 수와 관계없이 가시성 그래프의 계산은 1회만 이루어지며 캐릭터마다 그 크기에 따라 가시성 그래프에 보로노이 모서리를 추가함으로서 캐릭터별 로드맵을 효과적으로 생성하고 관리할 수 있도록 한다. 이와 같이 생성된 로드맵에서 계획된 경로는 장애물로부터 일정한 간격을 보장하며 직선과 곡선이 혼합되어 있는 자연스런 경로이다. 2장에서는 여러 가지 관련된 연구들을 살펴보고 본 연구와의 유사점과 차이점을 살펴보며, 3장에서는 제안한 방법의 기초가 되는 가시성 그래프와 보로노이 다이어그램을 생성하는 방법을 설명하고, 이를 이용하여 캐

릭터별 로드맵을 생성하는 과정을 4장에서 소개한다. 5장에서는 게임 배경 에디터를 이용하여 제작된 게임에 제안된 방법을 적용한 시뮬레이션 결과를 소개하며 6장에서 향후 과제를 짚어 보며 논문을 맺는다.

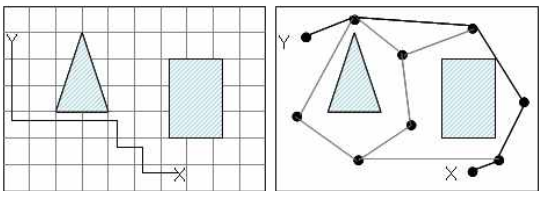
2. 관련연구

RPG 게임 배경이 셀-기반으로 이루어져 있기 때문에 있는 셀들을 그대로 이용하여 각 셀을 상태(state)로, 이웃한 셀로의 이동을 링크(arc)로 하는 상태공간을 생성하는 방법도 많이 사용된다. 격자-기반(grid-based) 방법이나 네비게이션 메쉬(navigation mesh) 방법 등이 이에 속한다[2,8,9]. 별도의 로드맵을 생성하지 않고 격자들의 집합을 상태 공간으로 하며 인공지능 탐색 기법인 A* 알고리즘으로 최단 경로를 찾아낸다. 이 방법은 게임 배경을 구성하는 격자들의 수가 많기 때문에 상태 공간이 매우 크며 셀들을 연결하여 경로가 구성되므로 그림 1(a)에서처럼 경로가 지그재그형으로 나타나는 것도 단점으로 지적된다[2,9].

컴퓨터 게임에서 초기에 가장 많이 사용되던 로드맵 방식은 레벨설계자가 사전에 캐릭터가 이동할 수 있는 몇 개의 경로점을 미리 지정해서 만든 경로점 그래프(waypoint graph) 방식이다[1]. 목적에 따라 경로점을 장애물의 모서리에 놓는 모서리그래프(corner graph), 중심에 놓는 원-기반(circle-based) 경로점 그래프도 있으며 그림 1(b)는 임의의 장소에 노드를 설정한 경로점 그래프이다. 공통적으로 설계 단계에서 캐릭터가 이동할 수 있는 지점을 사람이 설정하는 것이기 때문에 게임에 등장하는 모든 캐릭터에 대해 개별적인 로드맵을 할당하여 관리하는 것은 어렵다.

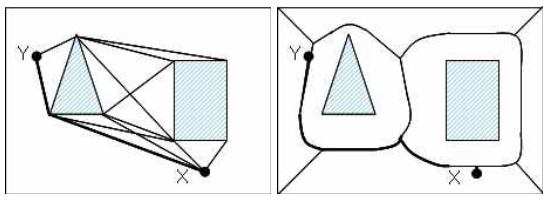
자동으로 로드맵을 생성하는 방법으로는 로보틱스 분야에서 개발된 가시성 그래프(visibility graph)와 보로노이 다이어그램(Voronoi diagram) 등이 있다. 가시성 그래프는 장애물의 꼭지점과 시작점, 목표점을 노드로 하고 각 쌍의 노드를 연결한 선분들 중에 다른 장애물과 교차 없는 선분들을 링크로 정의하는 무방향 그래프를 말한다[10]. 가시성 그래프는 Force21이라는 게임에서 실제로 사용되었을 뿐 아니라[5] 가시점(point of visibility) 아이디어는 다른 탐색 방법의 효율을 높이기 위해서도 응용되고 있다[11,12]. 가시성 그래프에 의해 탐색된 경로는 최단거리를 보장하는 것이 큰 장점이지만 장애물의 벽에 붙어 이동하게 되는 문제(wall-hugging problem)가 있어서 장애물을 임의의 상수만큼 부풀려 이를 해결하는 방법이 제안되기도 하였다[7]. 보로노이 다이어그램은 이와는 반대로 모든 장애물로부터 같은 거리만큼 떨어져 있는 지점에서 경로가 형성된다. 그러므로 보로노이 다이어그램에 의해 찾은 경로는 장애물을 최대한으로 멀리하여 움직임으로서 충돌위험을 최소화하는 장점이 있는 동시에 경로가 길고 비연속점이 자주 발생하는 단점이 있다[3]. 그림 2(a)는 가시성 그래프의 예를 (b)는 보로노이 다이어그램의 예를 보여준다. 이 두 가지 방식의 장점을 동시에 취하고자 VV-복합체가 제안되었다[6].

VV-복합체는 장애물에 대한 보로노이 다이어그램을 계산하고 장애물을 확대하여 민코스키 합한 결과에 대해 가시성 그래프를 계산하여 이 두 가지를 병합하여 나온 새로운 형태의 다이어그램이다. VV-복합체는 짧고(short), 부드럽고(smooth), 여유 공간(clearance)을 확보한 경로를 제공하기 위해 제안되었는데 비연속적인 보로노이 링크가 삽입됨으로서 이에 반하는 결과가 나올 수도 있다. 보로노이 다이어그램에 의한 비연속점은 그림 2(b)의 예제에 잘 나타나 있다. 본 논문에서는 장애물을 확대함에 따라 사라지는 자유공간에 장애물의 기하학적 요소에 따



(a) (b)

그림 1. 격자-기반(a)과 경로점 그래프(b)에서의 경로 탐색



(a) (b)

그림 2. 가시성 그래프(a)와 보로노이 다이어그램(b)에 의한 경로 탐색

라 이등분 궤적을 삽입하는 방법으로 전체 보로노이 다이어그램의 구성을 피하고도 VV-복합체와 유사한 효과를 거두며 이때 발생하는 비연속점들을 최소화함으로서 전체 경로가 어느 정도의 여유공간은 확보하는 동시에 부드러운 턴을 할 수 있는 방법을 제안한다.

3. 일반화 가시성 그래프와 개산 보로노이 링크의 생성

본 장에서는 제안하는 로드맵의 기초가 되는 가시성 그래프와 보로노이 다이어그램에 대해 설명한다. 가시성 그래프의 경우, 다각형 장애물을 움직이는 캐릭터의 반경만큼 확대하여 부풀려진 장애물에 대해 생성되어야 하므로 원의 호와 선분을 포함하는 일반화 다각형 환경에서 생성되는 과정을 설명한다. 원래 보로노이 다이어그램은 점들 사이에서 정의되어 있는데 다각형 환경으로 확장해야 하므로 이를 위해 다각형을 점의 집합으로 이산화한 후에 기존의 알고리즘을 적용하는 방법을 설명한다.

3.1 일반화 가시성 그래프의 생성

다각형 장애물 사이를 부피가 있는 캐릭터가 돌아다니게 하기 위해 다각형 장애물을 캐릭터를 최소로 둘러싸는 원의 반경(r)만큼 확장하여 가시성 그래프를 생성하는데 이 과정은 다음과 같다.

단계 1: 장애물의 경계를 r만큼 확장하여 장애물을 부풀린다.

단계 2: 부풀려진 장애물들을 합집합한다.

단계 3: 단계 2의 합집합 결과에 대해 가시성 그래프를 생성한다.

단계 1에서는 다각형 장애물(S)의 경계를 r만큼 확장하기 위해 다음 식과 같이 정의된 양의 솔리드 오프팅 연산자 $\uparrow^*$를 이용한다: $S\uparrow^* r = \{p : \exists\, q \in S, \|p-q\| \le r\}$ [13]. 이와 같은 정의에 따라 솔리드-오프세팅을 구현하기 위하여는[14]에서와 같이 장애물을 n개의 꼭지점으로 시계 반대 방향의 리스트 [v0,v1,v2, …, vn-1, vn](v0=vn)로 표현하고 다음과 같이 볼록(convex)과 오목(concave) 꼭지점의 경우로 나누어 처리한다. ni- 와 ni+ 를 선분 [vi-1, vi] 와 [vi, vi+1] 각각의 외법선(outward normal)이라고 할 때, q= angle(ni-, ni+)가 양이면, vi는 볼록이고 q 가 음이면 vi는 오목이다. vi가 볼록 꼭지점인 경우에 r만큼 양의 오프세팅을 하면 vi를 원의 중심으로 하고 vi로부터 각각 ni- 와 ni+ 방향으로 r만큼 뻗어 나간 두 점을 종점(endpoints)으로 하는 호로 변환된다. vi가 오목 꼭지점인 경우에는 vi는 ni- 와 ni+ 의 중선 방향으로 r/sin(q /2)만큼 뻗어 나간 오목점으로 변환된다. 이 연산의 결과는 반지름 r인 원이 다각형의 경계를 따라 쓸고 간 형태로 다각형 장애물 각각은 r만큼씩 부풀려져 더 이상 다각형이 아닌 지역적 비볼록 일반화 다각형(locally non-convex generalized polygon)이 된다[10].

단계 2의 일반화 다각형들을 합집합하는 과정은 다각형의 합집합 경우와 유사하다. 그림 3의 예에서 일반화 다각형 A와 B를 각각 {V_{a1}, V_{a2}, V_{a3}, V_{a4}, V_{a1}}와 {V_{b1}, V_{b2}, V_{b3}, V_{b4}, V_{b1}}의 순서리스트라고 하자. 여기서 V_i는 양 끝점과 중점으로 이루어진 가상 꼭지점(pseudo vertices)이다. A와 B가 P_1과 P_2에서 교차한다고 하면 도형 A의 V_{a1}에서 시계반대방향으로 출발하여 V_{a2}를 거쳐 P_1을 만나면 도형 B의 P_1을 찾아 그로부터 도형 B를 추적하고 P_2에 이르면 다시 도형 A에서 P_2를 찾아 V_{a1}을 만나면 이 과정을 종료하게 된다. 이 결과는 다음과 같은 순서리스트가 된다. {V_{a1}, V_{a2}, P_1, V_{b2}, V_{b3}, V_{b4}, P_2, V_{a4}, V_{a1}} 호와 호, 호와 일반 선분의 교차점도 마찬가지로 처리한다.

가시성 그래프는 시작 노드, 목표 노드, 장애물의 꼭지점들을 그래프의 노드로 하고 노드와 노드를 연결하는 선분 중에 장애물과 교차하지 않는 선분을 그래프의 링크로 하여 형성된다. 최단 경로를 찾기 위한 로드맵으로 사용되는 가시성 그래프는 일반적으로 축소형(reduced) 가시성 그래프이다. 축소형 가시성 그래프란 가시성 그래프의 노드 중에서 오목 노드는 모두 제거하고 링크 중에서는 양쪽 다각형 모두에 접하는 링크만 취한 그래프를 말하는데 오목

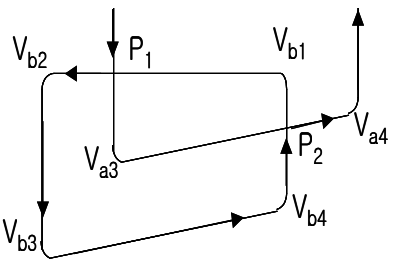


그림 3. 두 개의 일반화 다각형을 합집합하는 알고리즘

꼭지점으로 구성하는 링크는 절대 최단 경로를 구성하지 않기 때문에 최단경로를 찾기 위해서는 축소형 가시성 그래프면 충분하다[10]. 이와 같이 하면 가시성 그래프를 생성하는데 드는 시간도 절약할 수 있다. 일반화 다각형에 대해서 가시성 그래프를 생성하기 위하여 가상 꼭지점에 대한 링크를 정의해야 한다. 링크는 장애물과 교차하지 않도록 정의되어야 하므로 원의 호에 대해서는 접하는 선(tangent segment)이 링크가 되고 이때의 접점이 노드가 되면 된다. 볼록 꼭지점 사이에는 링크가 하나만 생성되지만 꼭지점과 원의 호 사이에는 최대 2개의 접선이, 호와 호 사이에서는 최대 4개의 접선이 가능하므로 접선의 수만큼의 링크가 생성될 수 있다. 그림 4는 일반화 다각형 위에 형성된 가시성 그래프의 예를 보여 준다. 점선들은 오목 꼭지점으로부터 구성된 링크를 나타내며 양쪽 노드 모두에 접선인 링크들은 실선으로 표기되어 있다. 최단 거리를 구하기 위한 축소형 가시성 그래프는 점선을 제외한 실선과 그의 접점들로 구성된다.

3.2 보로노이 링크의 생성

보로노이 다이어그램은 점들의 집합에 대해 정의되어 있다. 같은 직선상에 있지 않은 점들의 집합 $P=\{p_1,p_2, \cdots, p_n\}$가 있을 때, 점 pi의 보로노이 영역(Voronoi region) $R(p_i)$는 다음과 같이 정의된다[4]: $R(p_i) = \{p: |p_i-p| \leq |p_j-p|, \forall j \neq i\}$. 즉, 보로노이 사이트(site) pi에 대해 $R(p_i)$는 p_i에 가장 가까운 점들의 집합이다. $R(P_i)$와 $R(P_j)$의 경계는 사이트 p_i와 p_j에서 동일한 거리에 있는 점들의 집합이며 보로노이 모서리(edge)라고 부른다. 다시 말해, 보로노이 모서리는 p_i와 p_j에서 동시에 가장 멀리 떨어진 점들의 집합이라고도 말할 수 있다. 사이트가 점이 아닌 선 이상의 도형으로 확대되었을 때 같은 정의에 의해 만들어지는 보로노이 모서리의 집합을 일반화 보로노이 다이어그램(generalized Voronoi diagram)이라고 하는데 정의에 의해 일반화된 보로노이 다이어그램에 의해 찾은 경로는 장애물을 최대한으로 멀리 있게 되며 이는 장애물로부터 최대의 여유 공간을 확보하는 경로를 의미하여 보로노이 다이어그램을 이용한 경로 계획의 가장 큰 장점이 된다.

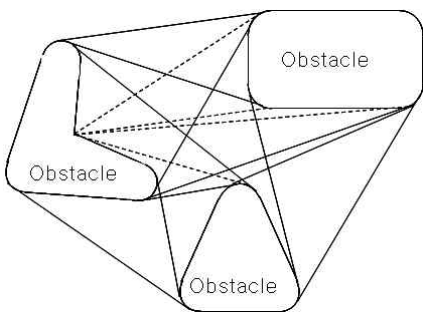


그림 4. 일반화 다각형에 대해 생성된 가시성 그래프

게임에서 장애물의 모양을 다각형으로 한정할 때, 보로노이 모서리는 두 장애물 사이의 기하학적 요소에 따라 다음 3가지 경우로 분류된다: ① 꼭지점과 꼭지점의 경우- 두 꼭지점의 중점, ② 모서리와 모서리 - 선분과 선분의 이등분선, ③ 꼭지점과 모서리 - 선분과 점에 이르는 거리가 같은 궤적인 포물선.

위와 같이 정해(exact solution)를 구하는 방법은 포물선을 포함하는데다가 구현하기가 어렵고 복잡하여 '개산'(approximate) 방법을 많이 사용한다[15]. 개산 방법에서는 다각형의 경계를 일정 수 이상의 점으로 이산화하여 점 사이트에 대해 기존의 방법으로 보로노이 다이어그램을 계산하고 불필요하게 생성된 모서리들을 제거하여 근사 일반화 보로노이 다이어그램을 구한다. 많은 수의 점을 사용할수록 정해에 가깝게 구해지지만 보로노이 모서리를 구하는 시간이 많이 소요되는 것은 자명하다. 효과적인 계산을 위해 Fortune의 알고리즘을 이용하여 보로노이 모서리를 구한다. Fortune의 알고리즘은 점 사이 중선의 교점을 수평선 혹은 수직선 방향으로 이동하면서 구하는 sweep-line 알고리즘을 응용하는 방식으로 기존의 방식에 비해 시간복잡도를 개선한 방식이다. 이렇게 장애물의 경계를 점으로 이산화하여 보로노이 모서리를 구하면 같은 장애물에 속해 있는 점들 사이에서 생성되는 불필요한 보로노이 모서리가 생기므로 이를 제거해야 한다. 불필요한 보로노이 모서리의 판단은 모서리의 끝점이 장애물의 경계 혹은 내부에 있는지 확인하여 하고 그렇지 않은 모서리들만 남긴다[16]. 그림 5(a)는 장애물의 경계를 점의 집합으로 가정했을 때, Fortune 알고리즘으로 보로노이 모서리를 생성한 결과이고 이로부터 불필요한 모서리를 제거한 결과는 그림 5(b)와 같다. 아래쪽 장애물의 꼭지점 v를 기준으로 살펴보면, v와 우측 장애물 모서리 e1과의 이등분선은 그 사이의 이등분점을

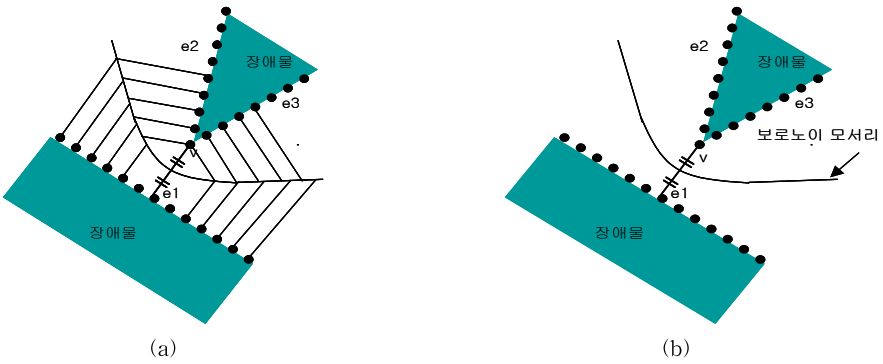


그림 5. 점으로 이산화된 장애물에 대한 보로노이 모서리 생성 과정

정점으로 하는 포물선으로, v에서 e2의 법선 방향으로 포물선과 만나는 점부터 e1과 e2 사이의 이등분선으로 근사치가 일치하는 것을 볼 수 있다.

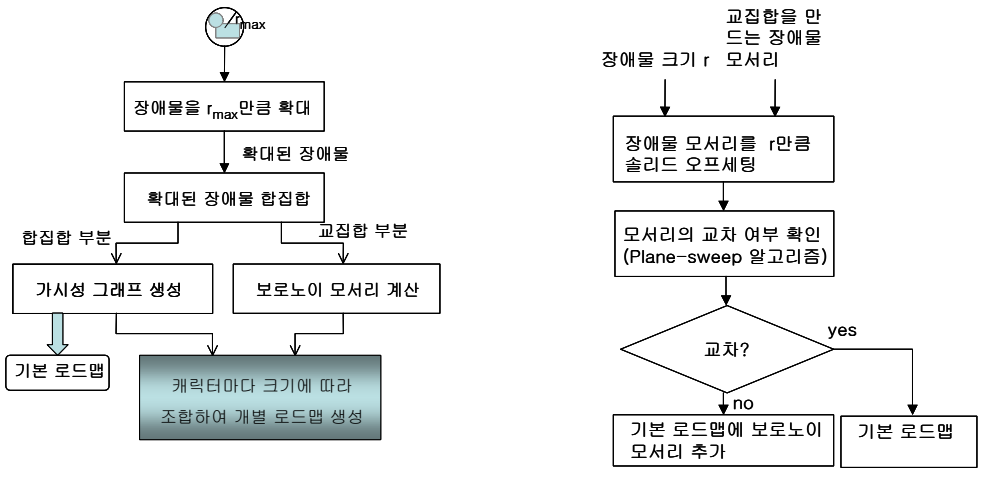
4. 캐릭터별 로드맵의 생성과 경로 계획

본 장에서는 크기가 다른 캐릭터별로 효과적으로 로드맵을 생성하는 과정과 생성된 로드맵을 이용하여 최적 경로를 탐색하는 방법에 대해서 소개한다.

4.1 캐릭터별 로드맵의 생성

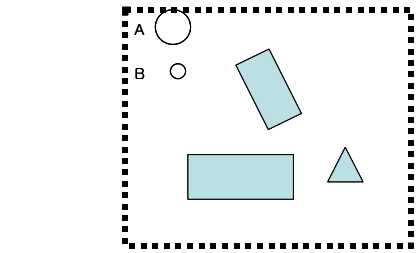
단계 1-3까지의 가시성 그래프의 생성과 개산 보로노이 모서리의 생성은 캐릭터의 수에 관계없이 1회만 계산되며 각각의 자세한 구현 방법은 Ⅲ장에서 설명되었다. 본 절에서는 이와 같이 생성된 가시성 그래프와 보로노이 모서리를 이용하여 캐릭터별로 로드맵을 구성하는 과정을 설명한다. 전체 흐름을 나타내는 흐름도는 그림 6(a)이고 음영이 있는 상자가 본 절의 내용에 해당하며 그림 6(b)와 같이 정리될 수 있다.

가시성 그래프는 가장 큰 캐릭터의 로드맵이자 모든 캐릭터의 기본 로드맵이 된다. 그림 7(a)에서와 같이 크기가 다른 캐릭터 A,B를 예로 들면 큰 캐릭터를 둘러싸는 원의 반경만큼 장애물을 확대하여 가시성 그래프를 생성한 그림이 (b)와 같다. 확대한 장애물 사이에 교집합이 생긴 부분은 캐릭터 A가 지나갈 수 없는 길이므로 캐릭터 A의 로드맵에서는 막힌 길이 되어 링크가 없지만 그보다 작은 캐릭터인 B에

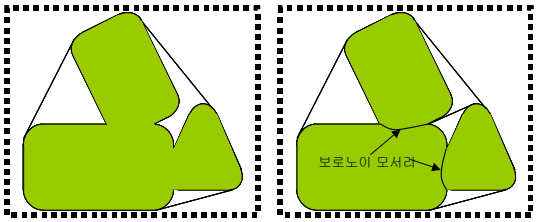


(a) 전체 흐름도 (b) 개별 로드맵 생성 흐름도

그림 6. 개별 로드맵 생성 흐름을 나타내는 흐름도



(a) 장애물과 크기가 다른 두 캐릭터



(b) 캐릭터 A에 대한 로드맵 (c) 캐릭터 B에 대한 로드맵

그림 7. 크기가 다른 캐릭터에 대한 로드맵 생성과정

대해서는 지나갈 수 있는지 확인하여 링크를 추가해 주어야 한다. 그림 7(c)에 보이는 2개의 보로노이 모서리가 캐릭터 B를 위해 추가된 링크이다. 즉 최종 로드맵은 가시성 그래프와 보로노이 다이어그램의 복합체인 VV-복합체 구조가 되는 것이다.

이제 남은 문제는 작은 캐릭터가 교집합 부분을 통과할 수 있는 크기인지 어떻게 효율적으로 판단하는 가인데 정리하면 다음과 같다. 교집합을 생기게 하는 장애물의 경계선들은 이미 알려져 있으므로 이 선들에 대해 솔리드 오프세팅을 하여 그 결과들이 서로 교차하는지 확인하면 된다. 3.1절에서 설명한 바와 같이 경계선의 종점(endpoint)에 대해 볼록점인 경우에 r만큼 양의 오프세팅을 하면 양변의 외법선 방향으로 r만큼 뻗어 나간 두 점을 종점으로, 오목점인 경우에는 양변의 외법선들의 중선 방향으로 $r/\sin(q/2)$만큼 뻗어 나간 오목점으로 변환된다. 외법선은 가장 큰 캐릭터의 로드맵을 구하기 위해 이미 계산되어 있으므로 추가적인 계산 시간은 거의 소요되지 않는다. 이제 이렇게 확대된 경계선들이 서로 교차하는가를 판별해야 하는데 이를 위해서는 선분들의 교차를 빠르게 확인하기 위한 Plane-sweep 알고리즘을 호와의 교차로 확장하여 사용한다.

Plane-sweep 알고리즘은 n개의 선분의 교차를 구하기 위하여 각각의 선분과 나머지 (n-1)개의 선분에 대해 일일이 교차 여부를 확인하는 것이 아니라 선분의 정렬을 통해 이웃하는 선분끼리만 교차 여부를 확인하면 되도록 고안된 알고리즘이다[17]. 원리를 간단히 살펴보면 수평의 선(sweep-line), L이 선분을 구성하는 y값 중에 가장 큰 값으로부터 시작하여 y축 음의 방향으로 스위핑되면서 선분의 ① 시작점, ② 끝점, ③ 두 선분 사이의 교점과 같은 이벤트가 발생할 때 , L의 상태를 변화시킨다. 선분의 시작점과 끝점은 y축 정렬된 방향을 기준으로 정한다. L의 상태는 동적 큐(dynamic queue), Ł,로 표현되는데 이벤트 ①이 발생하면 해당하는 선분을 큐에 넣고 이벤트 ②가 발생하면 해당하는 선분을 큐에서 제거하며 이벤트 ③이 발생하면 Ł에서 해당하는 두 선분의 위치를 바꾼다. 큐 Ł은 x의 값으로 정렬된 순서를 유지하도록 한다. 이와 같이 하면 L의 상태가 변할 때마다 새로이 이웃하게 되는 선분들 사이의 교점만 구하면 된다.

본 논문에서와 같이 선분뿐 아니라 원의 호가 포함되어 있는 구조에서는 원의 호가 선분처럼 단순성(monotonicity)가 없기 때문에 정렬이 의미가 없으며 스위핑이 시작되기 전에 L의 방향과 L과 수직 방향으로 호를 분할(decompose)하면 정렬이 필요한 모든 축에 대해 단순 증가 혹은 감소하므로 Plane-sweep 알고리즘이 적용 가능해 진다. 그림 8은 그림 7 예제의 교차 부분을 확대한 것이며 원의 호는 L의 스위핑 방향의 접선에 대해 단순 감소하는 a_1과 단순 증가하는 a_2로 분할되어 있음을 볼 수 있다. Plane-sweep 알고리즘에 의해 선분 s_1, s_2, s_3이 차례로 삽입되고 교차 이벤트인 X_{13}이 발생할 때, L의 상태가 어떤 순서로 변하는지 보여준다. 장애물의 꼭지점의 수가 O(n)이라 할 때, 일일이 교차여부를 확인할 때의 $O(n^2)$의 시간복잡도를 O(nlogn)으로 향상시켜 두 확대된 장애물 사이에 교차가 발생하는지 효율적으로 알 수 있다.

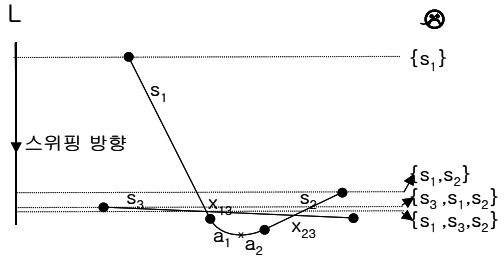


그림 8. plane-sweep 알고리즘 적용 과정

4.2 경로찾기 탐색 알고리즘

로드맵들이 완성되면 캐릭터 c가 이동할 시작점 s와 목표점 g까지의 최단 경로를 찾아야 한다. 즉, 경로 찾기는 로드맵의 집합 M에 대한 3개의 입력 변수 <s, g, c>의 질의로 정의할 수 있다. 여기서 s와 g는 캐릭터 c를 둘러싸는 원의 중심으로 가정하며 c는 캐릭터의 고유번호를 정수로 나타낸 것이다. 주어진 질의에 대해 고유번호 c와 매칭하는 로드맵을 찾고 그 로드맵에서 s와 g 각각에 가장 가까운 노드에 연결한다. 이 때, Plane-sweep 알고리즘에서 y축으로 스위핑하던 것과 유사하게 s나 g를 중심으로 시계방향으로 회전 스위핑하면서 가까운 노드를 효과적으로 찾을 수 있다. 이와 같이 하면 그래프를 탐색하여 경로를 찾을 준비가 끝나는 것이며 A* 탐색 알고리즘을 이용하여 최단 경로를 찾을 수 있다. A* 알고리즘은 현재 위치에서 목표 위치까지 소요될 비용을 추정한 값을 실제 시작지점에서 현재 노드까지 소요된 비용에 합하여 노드의 평가 값으로 이용하여 최소 비용의 노드를 우선적으로 탐색한다.

추정 값을 휴리스틱이라고 하며 실제 소요될 값보다 작게 추정하여야 최단 경로를 찾을 수 있음이 증명되어 있다. 최단 경로를 찾기 위한 휴리스틱으로는 목표점까지의 직선거리를 나타내는 유클리디언 거리를 사용하는데 직선이 아닌 원의 호와 포물선을 포함하는 VV-복합체 구조에서도 유클리디언 거리가 과소 추정된 휴리스틱임을 다음과 같이 간단하게 증명할 수 있다. h(u)와 h(v)를 u와 v에서 목표 노드까지 각각의 직선거리라 하고, w(u,v)를 u에서 v로의 실제 이동에 드는 비용이라고 할 때, $h(u) \leq h(v)+w(u,v)$이다. VV-복합체에서 링크는 선분, 원의 호, 포물선 중의 하나이므로 실제 이동하는 경로도 선분, 호, 포물선의 궤적위에 만들어진다. 그러므로 실제 이동 거리는 직선거리로 정의된 휴리스틱 값보다 항상 크게 된다. 이 식은 탐색 알고리즘의 단조성을 나타내며 허용성의 충분조건이 된다.

5. 구현 및 시뮬레이션 결과

캐릭터별 로드맵 생성을 위해 제안된 방식의 결과를 셀-기반 편집기에서 샘플로 제작된 게임에 적용시켜 분석하였다. 이 셀-기반 편집기는 RPG에서 게이머가 자신이 원하는 맵을 직접 구성할 수 있도록 많이 사용되는 배경 화면 편집기이다[18]. 우선 모든 캐릭터에 대해 가시성 그래프로 동일한 로드맵을 적용한 경우와 캐릭터의 크기에 따라 다른 로드맵을 적용한 경우의 생성된 경로의 질을 비교하였다. 4장의 그림 7의 예제를 이용하여 시작점과 목표점을 지정하고 두 가지 방식에 대해 다른 크기의 캐릭터의 이동경로를 표시한 결과가 그림 9에 나타나 있다. 그림 9(a)에서는 모든 캐릭터에 대해 공통 로드맵을 사용하고 그 로드맵은 가장 큰 캐릭터의 기준으로 만들어지게 되므로 작은 캐릭터들이 가까운 경로를 이용하지 못하는 결과가 나오는 반면 그림 9(b)에서는 각기 적합한 경로를 찾아 이동하는 모습을 보여준다. 그림 10은 같은 실험을 제작된 게임 배경에 대해 실행한 결과이다. 셀 기반 편집기에 좌측에 있는 6가지의 장애물을 이용하여 배경을 생성한 후, 'START'와 'END'를 선택하여 시작점과 목표점을 화면에 클릭하고 프로그램을 실행하면 캐릭터들이 각기 다른 경로로 이동하는 것을 확인할 수 있다. 그림 10은 이와 같이 실행된 경로를 따라 캐릭터 크기의 원으로 자취를 남긴 것이다.

위의 결과에서 볼 수 있듯이 크기가 다른 캐릭터들이 개별 로드맵을 가지고 경로를 계획하여 이동하면 단일 로드맵 위에서 이동하는 것에 비해 자연스럽게 보이는데 이와 같은 효과를 위해서는 개별 로드맵을 생성하기 위한 오버헤드가 발생한다. 이를 최소화하기 위해 본 연구에서 제안한 방법은 전체 보로노이 링크를 생성하지 않고 필요에 의해 최소 부분에 대해서만 가시성 그래프에 개산 보로노이 링크를 추가하는 것이었는데 이 방법에 의해 발생하는 오버헤드가 공통으로 사용하는 로드맵을 생성하는 시간에 비하여 어느 정도인지 캐릭터의 크기에 따라 비교해 보았

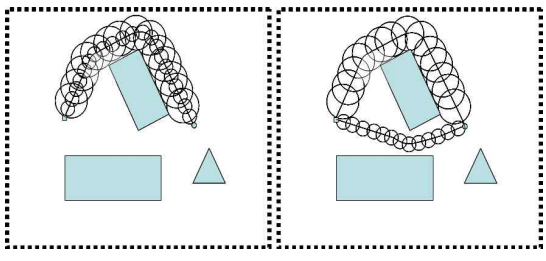


그림 9. 크기가 다른 캐릭터 A와 B에 대해 공통 로드맵을 이용하여 이동하는 경우(a)와 개별 로드맵을 사용하여 다른 경로로 이동하는 경우(b)의 비교

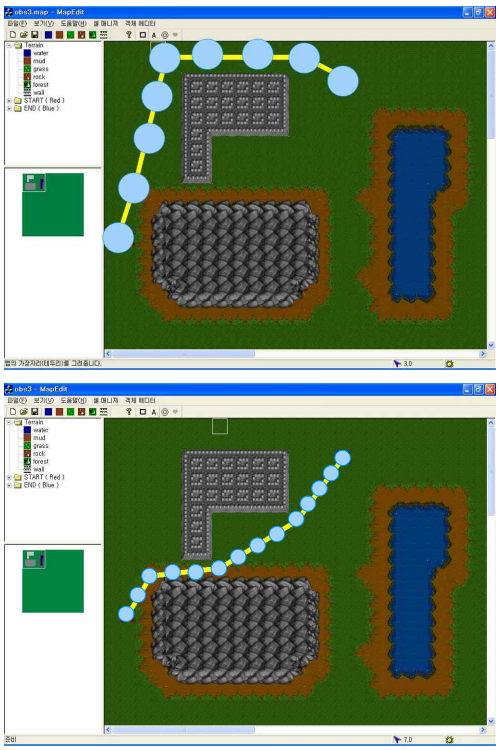


그림 10. 실제 게임에서 크기가 다른 캐릭터들이 다른 경로로 이동하는 예

다. 그림 11의 -◆-로 나타난 꺽은선 그래프인데 가장 큰 캐릭터에 대해 생성하는 공통 로드맵을 생성하는 데 드는 시간을 기준값 1로 했을 때의 비율로 나타내었다. 크기가 다른 여러 캐릭터가 존재하는 경우, 막힌 길 중에 가장 큰 캐릭터가 지나갈 수 있는 부분부터 보로노이 링크가 추가되고 차례로 그 다음으로 큰 캐릭터가 지나갈 수 있는 부분에 보로노이 링크를 추가하게 되므로 그림 11의 결과와 같이 캐릭터의 반지름이 작아질수록 오버헤드가 커지는 것을 볼 수 있다. 하지만 보로노이 링크의 추가는 더욱 짧은 경로를 보장하므로 작은 캐릭터들은 경로 이동 시간을 단축하게 되는데 가장 큰 캐릭터를 기준으로 하여 작은 캐릭터들의 경로 이동 시간 이득을 비교한 결과가 그림 11의 -■- 꺽은선 그래프로 표시되어 있다. 경로 이동으로 얻는 이득은 로드맵 생성의 오버헤드에 비해 비율적으로 작지만 로드맵은 한번 생성되면 반복해서 사용되고 이에 대해 많은 수의 경로 이동이 일어나게 되므로 이에 대한 보상의 정도는 실행회수가 거듭됨에 따라 증가할 것이다.

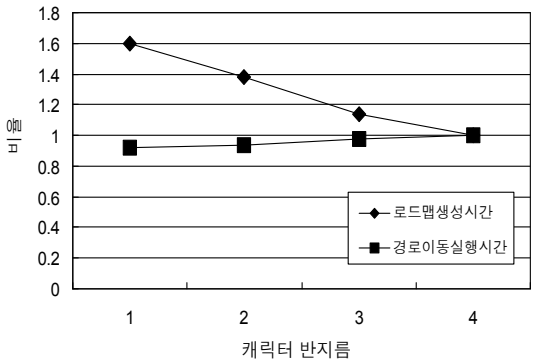


그림 11. 개별 로드맵 생성에 따른 오버헤드 및 실행 시간 이득의 비교

6. 결 론

본 논문에서는 게임에 등장하는 여러 캐릭터들의 크기에 맞추어 최적의 경로를 제공할 수 있도록 개별 로드맵을 생성하는 효과적인 방법을 제안하였다. 각각의 캐릭터에 대해 일일이 로드맵을 따로 생성하는 것이 아니라 가장 큰 캐릭터에 대해 생성된 가시성 그래프를 기본으로 그보다 작은 캐릭터들에 대해 점진적으로 보로노이 링크를 추가함으로서 효과적으로 개별 로드맵을 가질 수 있도록 하였다. 시뮬레이션 결과에서 보듯이 캐릭터의 크기에 맞는 다른 경로를 택하여 이동하는 것이 경로의 최적성(optimality) 측면에서나 질적인 측면에서 장점이 된다. 즉, 크기가 다른 캐릭터들이 각각의 로드맵을 가지고 경로를 계획하기 때문에 그 캐릭터에 최적인 경로를 찾아내어 우회하는 상황이 감소할 뿐 아니라, 이는 전체 게임의 흐름 속에서 캐릭터들의 움직임이 자연스럽게 보이도록 하는데도 도움이 된다.

캐릭터의 크기가 2-4종류일 때 모든 캐릭터에 대해 하나의 로드맵을 생성하여 공유하는 것보다 캐릭터 별로 로드맵을 생성하는 것이 약 1.14-1.60배까지의 시간이 소요되었으나 최단 경로로 이동하므로 이동 시간은 0.92-0.98 정도로 줄어들었다. 실행 이득이 로드맵 생성 시간의 손해보다 작지만 로드맵이 일단 한번 생성되고 나면 실행은 반복해서 하게 되므로 궁극적으로는 제안한 방법에 의해 얻게 되는 이득이 월등해 진다고 할 수 있다.

본 논문에서는 캐릭터의 특성을 크기에만 국한하였는데 향후에는 다양한 네비게이션 특징을 고려하

여 개별 로드맵을 만들 수 있도록 확장하는 것이 필요하다. 예를 들어 캐릭터의 종류에 따라 장애물로 분류되어야 할 지형이 다를 수 있으며 지형에 따라 이동 속도도 차이가 많이 날 수 있다. 이를 위해서는 가중치 있는 링크(weighted link)를 포함하는 가시성 그래프로 확장이 필요하며 보로노이 링크도 가중치를 가질 수 있어야 하고 이들을 탐색하는 A* 알고리즘의 평가 함수 또한 가중치를 포함하여 다시 정의되어야 한다.

참 고 문 헌

[1] Tozour, P., "Search Space Representations," AI Game Programming Wisdom 2, Charles River Media, pp. 85-102, 2004.

[2] Rabin, S., "A* Speed Optimizations and A* Aesthetic Optimizations," In: Deloura, M. (eds.): Game Programming Gems. Charles River Media, pp. 264-287, 2000.

[3] D. Nieuwenhuisen, A. Kamphuis, M. Mooijekind, M.H. Overmas, "Automatic Construction of Roadmaps for Path Planning in Games," http://citeseer.ist.psu.edu/nieuwenhuisen04automatic.html, 2004.

[4] K. D. Forbus, J.V. Mahoney, K. Dill, "How Qualitative Spatial Reasoning Can Improve Strategy Game AIs," AAAI spring symposium on AI and Interactive Entertainment, pp. 35-40, 2001.

[5] Woodcock, S., "Game AI The State of the Industry," Game Developer Magazine Aug. 2000.

[6] Wein, R.,Van der Berg, J.P., Halperin, D., "The Visibility-Voronoi Complex and Its Applications," In: Proc. European Workshop on Computational Geometry, pp. 151-154, 2005.

[7] Young, T., "Expanded Geometry for Points-of-Visibility Pathfinding," Game Programming Gems 2, Charles River Media, pp. 317-323, 2001.

[8] Yap, P., "Grid-Based Path-Finding," Lecture Notes in Artificial Intelligence, Vol. 2338, pp. 44-55, 2002.

[9] Snook, G., "Simplified 3D Movement and Path-finding Using Navigation Meshes," Game Programming Gems, Charles Rive Media, pp. 288-304, 2000.

[10] Latombe, J.C., Robot Motion Planning, Kluwer Academic Publishers, 1991.

[11] Young, T., "Optimizing Points-of-Visibility Pathfinding," Game Programming Gems 2, Charles River Media, pp. 324-329, 2001.

[12] Jung, D., Kim, H., Kim, J., Um, K., Cho, H., "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, pp. 50-53, Sep. 2004.

[13] Rossignac, J., and Requicha, A.G., "Offsetting Operations in Solid Modelling," Computer Aided Geometric Design, Vol. 3, pp. 129-148, 1986.

[14] 유견아, "NPC의 자연스러운 이동경로를 보장하는 효율적인 상태공간의 생성," 정보과학회 논문지:소프트웨어 및 응용, 제 34권 4호, pp. 368-376, 2007.

[15] Roque, W. L., and Doering, D., "Constructing Approximate Voronoi Diagrams from Digital Images for Generalized Polygons and Circular Objects," Proceedings of the 11th International Conference in Computer Graphics, pp. 119-125, Feb. 2003.

[16] Allen, P., Stamos,P., Gueorguiev, A., Gold, E., Blaer, P., "AVENUE: Automated Site Modeling in Urban Environments", Proceedings of Third International Conference on 3-D Digital Imaging and Modeling, pp 357-364, 2001.

[17] O'Rourke, J., Computational Geometry in C, Cambridge Press, 1998.

[18] http://blog.naver.com/harkon/.

유 견 아

1982년～1986년 서울대학교 공과대학 제어계측공학과 학사
1986년～1988년 서울대학교 공과대학 제어계측공학과 석사
1988년～1989년 한국통신공사 사업지원단
1989년～1995년 미국 USC Computer Science 박사
1996년～덕성여자대학교 컴퓨터공학부 교수
관심분야 : 인공지능, 로봇 알고리즘, 연산 기하학