한국시뮬레이션학회 논문지
Vol. 16, No. 1, pp. 31-37 (2007. 3)

일반화 가시성그래프에 의해 계획된 경로이동 시뮬레이션

유견아[1†] · 전현주[1]

Movement Simulation on the Path Planned by a Generalized Visibility Graph

Kyeonah Yu · Hyunjoo Jeon

ABSTRACT

The importance of NPC's role in computer games is increasing. An NPC must perform its tasks by perceiving obstacles and other characters and by moving through them. It has been proposed to plan a natural-looking path against fixed obstacles by using a generalized visibility graph. In this paper we develop the execution module for an NPC to move efficiently along the path planned on the generalized visibility graph. The planned path consists of line segments and arc segments, so we define steering behaviors such as linear behaviors, circular behaviors, and an arriving behavior for NPC's movements to be realistic and utilize them during execution. The execution module also includes the collision detection capability to be able to detect dynamic obstacles and uses a decision tree to react differently according to the detected obstacles. The execution module is tested through the simulation based on the example scenario in which an NPC interferes the other moving NPC.

Key words : Computer game, Path planning, Decision tree, Visibility graph

요 약

최근 컴퓨터 게임에서 중대성이 부각되고 있는 NPC(NonPlayer Character)는 게임의 기본이 되는 이동에 있어서 스스로 장애물과 다른 캐릭터들을 인지하고 자신의 임무를 수행하여야 한다. NPC들의 자연스러운 이동을 위해 고정된 장애물 환경에서 일반화 가시성그래프를 이용하여 경로를 계획하는 방법이 제안된 바 있는데 본 논문에서는 이렇게 생성된 경로를 따라 효율적으로 이동할 수 있게 하기 위한 실행 모듈을 개발한다. 일반화 가시성그래프의 특성에 따라, 계획된 경로는 직선과 원의 호로 이루어져 있는데 본 실행 모듈에서는 이 특성에 적합하고 NPC의 움직임이 자연스럽도록 직선 이동 동작, 원 이동 동작, 도착하기 등의 기본 조타 동작을 정의하여 실제 경로 이동의 실행에 이용한다. 또한 이동 중에 나타나는 동적 장애물을 감지하기 위해 충돌감지 기능을 실행 모듈에 포함시키며 감지된 장애물의 종류에 따라 선택적으로 대처하기 위해 의사결정나무를 이용한다. 실행 모듈을 테스트하기 위해 NPC의 경로 이동을 방해하는 다른 NPC가 등장하도록 예제 시나리오를 작성하여 시뮬레이션한다.

주요어 : 컴퓨터 게임, 경로 계획, 의사결정나무, 가시성그래프

1. 서 론

게임에서 게이머가 조작할 수 없으며 컴퓨터에 의해 자동으로 움직이고 말하는 캐릭터를 NPC(Non-Player Character)라고 한다. 가상 세계 안에 항상 존재하면서 유저들과 상호작용을 해주는 NPC들을 통해 싱글 게임에서도 멀티플레이 게임을 하는 것과 같은 경험을 할 수 있게 된다. NPC들은 게임 환경 안에서 각기 맡은 임무에 따라 다른 위치에 존재하며, 다른 이동을 하게 된다. 게임의 기본이 되는 이동에 있어서도 스스로 장애물과 게이머의 캐릭터를 인지하고 자신의 임무를 다해야 하는 것이다 (Laird, 2000). 경로가 설계자에 의해 임의로 미리 지정되

* 본 연구는 2006년도 덕성여자대학교 자연과학연구소 연구비 지원에 의해 수행되었다.
2006년 11월 8일 접수, 2006년 11월 25일 채택
[1] 덕성여자대학교 컴퓨터공학부
주 저 자 : 유견아
교신저자 : 유견아
E-mail; kyeonah@duksung.ac.kr

어 있는 방식의 게임과는 달리 사용자가 원하는 맵을 만들 수 있는 랜덤 지형 맵 방식의 게임에서 이렇듯 많은 캐릭터들이 등장하게 된다면 각각의 캐릭터들은 게임을 시작할 때 장애물들을 탐색하고 이동경로를 생성하는 작업을 각기 수행해야 할 것이다(Woodcock, 2000). 기존의 경로계획 방법 중, 그리드 기반 방법(Yap, 2002)과 네비게이션 메쉬(Snook, 2000) 방법은 경로를 탐색하기 위한 상태공간의 크기가 너무 커서 이용하기에 어려움이 있으며 웨이포인트 그래프 (Liden, 2002)는 설계자가 미리 지정하는 방식이므로 랜덤 지형 맵 방식에 이용하는 데에 문제가 있다. Tozour(2004)는 위의 방법들로 게임 공간을 표현하는데 대한 장단점을 분석해 놓았다. 반면에 가시성 그래프(Visibility Graph, Vgraph)는 장애물의 볼록 정점을 노드로 하고 서로 보이는, 즉 장애물과 교차하지 않는 두 정점 사이의 연결선을 링크로 하는 그래프를 이용하는 방법으로 랜덤 지형 맵 방식에 적합하다 (Woodcock 2000). Vgraph는 탐색공간의 크기가 작을 뿐 아니라 Vgraph 자체가 로드맵의 역할을 하여 A* 알고리즘을 이용하면 최단거리의 경로를 구할 수 있음이 증명되어 있다 (Latombe, 1991). 그러나 Woodcock(2000)은 Vgraph에 의해 생성된 경로는 장애물의 한 정점에서 다른 장애물의 정점으로 이동하거나 장애물의 경계선을 따라가는 것으로 게임에서 캐릭터가 움직이는 방식으로는 어색함을 지적하였으며 보다 자연스럽고 안전한 이동 경로를 확보하기 위해 일반화 가시성그래프(Generalized Vgraph, GVgraph)에 의한 경로 계획 방법이 제안된 바 있다. GVgraph는 장애물 위에 직접 Vgraph를 만드는 것이 아니라 확장된 장애물 위에서 생성되는 것이기 때문에 GVgraph 상에서 계획된 경로는 장애물에서 일정 거리 이상 떨어진 일련의 직선과 원의 호로 구성되게 된다. 본 논문에서는 GVgraph에 의해 계획된 경로를 따라 가는 NPC의 이동이 효과적이고 사실적으로 보이게 하기 위한 단위 동작들을 정의하여 구현하며 이동 중에 만난 동적 장애물에 선택적으로 대처하는 방법을 제안한다. 필요한 단위 동작으로는 GVgraph에서 계획된 경로가 직선과 원의 호로 구성되므로 '직선이동'과 호를 궤적으로 하는 '곡선이동'이 있고 자연스러운 정지를 위한 '도착하기', 이동 중 다른 캐릭터와의 충돌을 감지하는 '충돌감지' 등이 있다. 이동중에 만나는 장애물에 대해 선택적으로 대처하기 위해 의사결정나무(decision tree)를 이용하며 장애물이 이동중인가 아닌가, 일정 시간 경과 후 벗어나는가 아닌가 등의 경우에 따라 다른 단위 동작을 정의한다. 각 단위 동작을 효과적으로 구현하는 방법을 설명하고 실제 시뮬레이션 해봄으로써 경로 이동에 대한 실제적 효율성을 살펴볼 것이다.

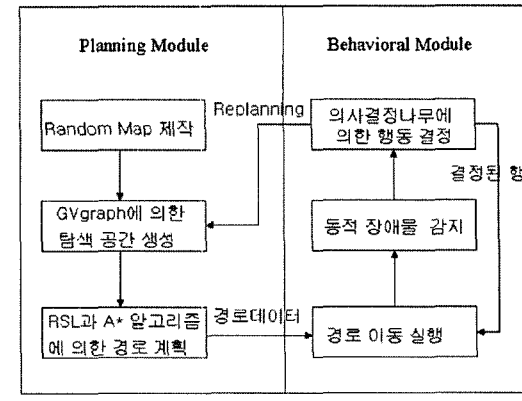

그림 1. 패스 파인더의 구조도

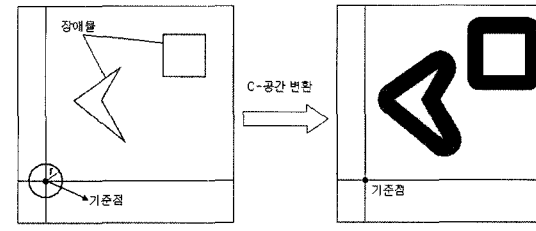

그림 2. C-공간 문제로의 변환

2. 선행 연구

게임 패스 파인더(path-finder)는 그림 1과 같이 계획 모듈(planning module)과 실행 모듈(execution module) 수 있다. 계획 모듈에서는 지도 환경으로부터 형상 공간(C-공간, configuration space)을 구성하고 이로부터 전역적으로 경로를 계획하며 실행 모듈에서는 동적인 주변 환경에 대해 반응적이면서도 신속한 경로 이동을 실행한다. 본 절에서는 (Yu, 2006)에서 제안된 방법인 일반화 가시성그래프을 이용한 경로 계획을 소개한다.

움직이는 캐릭터의 크기를 고려한 경로 계획을 위해 그림 2와 같이 캐릭터를 반지름 r의 원형으로 모델링하고 C-공간 문제로 변환한다. C-공간은 움직이는 물체를 점으로 치환할 때 환경을 구성하는 장애물들을 그에 맞게 확장한 공간이다. 이 확장 과정은 Minkowski 합과 차 연산의 특별한 경우로 정의되는 양의 솔리드 오프세팅 (positive solid offsetting) 연산을 적용하면 된다 (Rossignac, 1986). NPC를 원으로 모델링했을 때의 확장 결과는 직선분과 원의 호로 이루어진 일반화 다각형이다

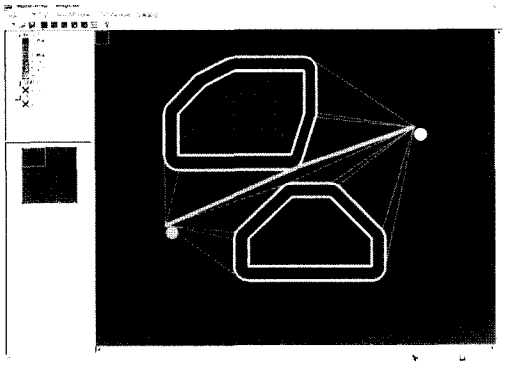

그림 3. 확장된 장애물 위에 생성된 GVgraph와 최단 경로

(Laumond, 1987). 그림 2에서처럼 볼록 정점의 경우에는 호를 형성하고 오목한 경우에는 또 다른 정점으로 변환되어 일반화 다각형을 형성하게 되는 것이다. Young (2001)에서와 같이 움직이는 캐릭터를 원이 아닌 다각형을 모델링하여 C-공간으로 치환할 수도 있다. 그러나 이와 같이 하면 움직이는 캐릭터가 방향을 바꾸지 않는다고 가정하여야 2차원 C-공간으로 치환되며 자유로운 방향을 다루기 위해서는 C-공간이 3차원이 되어 C-공간에서 경로를 계획하는 일이 복잡해지는 단점이 있다.

장애물이 일반화 다각형으로 표현된 경우에 호를 포함하기 때문에 다각형 위에 생성되는 가시성그래프(Vgraph)의 구성요소 이외에 호에 관한 부분이 포함되는데 이와 같이 일반화 다각형 위에 세워진 Vgraph를 일반화 가시성그래프(GVgraph)라고 한다. 먼저 Vgarph와 마찬가지로 노드에는 시작점, 목적점, 장애물의 정점을 포함한다. 호의 경우에는 호의 가시성을 따져주기 위해 가상점(fictitious point)을 생성해야 한다. 정점과 호 사이에서는 정점에서 호로 그은 접선의 접하는 점이 가상점이다. 정점에서 하나의 호에 최대 2개의 가상점이 생성된다. 호와 호 사이에서는 하나의 호에 대해 외접점과 내접점이 최대 2개씩 그러므로 총 4개의 가상점이 생긴다. 즉, GVgraph의 노드에는 가상점이 추가되고 링크에는 가상점을 생성한 선분과 호 위에 추가된 두 개의 이웃한 가상점을 연결한 부분이 추가된다. 이렇게 생성된 GVgraph는 그림 3과 같으며 그래프에 유클리디언 거리 휴리스틱과 함께 A* 알고리즘을 적용하면 그래프의 최단 경로를 찾을 수 있다. 그림 3에서 굵은 선이 최단거리를 나타낸다.

3. 경로 이동의 실행

계획 모듈에 의해 계획된 경로와 게임 환경에 대한 정보는 실제 경로 이동을 실행하게 되는 실행 모듈에 전달된다. 각각의 NPC는 자신의 시나리오에 따라 알맞은 기본 동작들을 선택하고 계산된 속도에 따라 비연속적으로 이동을 실행하게 된다(Reynolds, 1999). 직선과 원의 호로 이루어진 경로를 따른 평면 이동의 실행을 위해 필요한 기본 단위 동작들과 동적인 장애물을 만났을 때 장애물의 특성에 따른 대처 동작들에 대해 알아본다.

3.1 단위 동작의 실행

GVgraph에 의해 구해진 경로는 직선과 원의 호로만 구성되어 있기 때문에 계획 모듈에 의해 전달된 경로 데이터를 이용하여 이동을 실행하기 위한 단위 동작은 '직선 경로 이동'과 '원 경로 이동'이 필요하다. 여기에 캐릭터의 이동 방향을 바꾸게 되는 연결 부분에서의 이동이 자연스럽게 보이게 하기 위해 '도착하기'를 정의한다.

① 직선 경로 이동

직선운동의 경우 도착점에 도달하지 않는다면 등속도 운동을 가정한다. 직선형태의 경로가 시작되는 노드와 다음 직선이나 원 형태의 경로가 시작되는 노드(앞의 직선이 끝나는 지점)의 좌표를 통해 해당 직선의 방향 벡터를 계산하고, 일정한 속도를 유지하면서 현재 위치에 앞에서 계산한 방향벡터를 적용하여 시간에 따라 다음 위치로 이동한다. 방향벡터는 초기에 한번만 계산된다.

② 원 경로 이동

직교좌표(Cartesian coordinate)에서의 원의 방정식을 프로그래밍에 사용할 경우 픽셀로 그려지는 컴퓨터 모니터화면에서는 정확히 표현할 수가 없으므로 평면 극좌표(polar coordinate)에서의 원의 중점에 대한 식($x = r\cos\theta, y = r\sin\theta$)을 이용하여 세타($\theta$)의 값을 변화시킴으로 원 형태 경로의 이동을 표현한다. 특별히 원 운동의 경우 사람의 이동과 흡사한 모습을 위해 직선운동보다 속도를 감속하여 이동하게 한다.

③ 도착하기

보통 목표 위치에서는 NPC가 사람과 마찬가지로 부드럽게 정지하기를 원하는데, 도착하기는 NPC를 목표 위치로 감속해가는 방식으로 조타하는 행동이다. NPC가 목표에 도달하는데 얼마나 많은 시간을 쓰길 원하는가를 계산

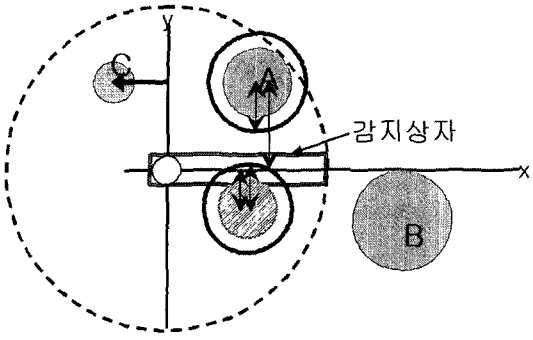

그림 4. 감지된 장애물의 여러 가지 경우

하기 위해 이 값을 사용한다. 이 값을 통해 NPC가 목표까지 원하는 시간 내에 도달하기 위해 얼마만한 가속도로 가야 하는지를 계산할 수 있다. 이후, 계산 된 가속도를 현재 NPC의 속도 벡터와 목표 위치에 도달하기 위한 속도 사이의 계산에 적용하여 감속하는 행동을 보이게 된다.

3.2 장애물 충돌 감지(Obstacle collision detection)

앞 절에서 소개된 단위 동작들의 실행은 시간-이산적이므로(time-discrete) 정해진 주기로 동적 장애물의 등장을 체크하여 감지 할 수 있다. 고정 장애물은 이미 계획 단계에서 고려되어 생성된 경로이므로 동적 장애물의 등장만 감지해내면 된다. 여기서 동적 장애물은 움직이는 NPC의 경우와 마찬가지로 장애물을 둘러싸는 최소의 원으로 모델링 되며 NPC의 진행 방향으로 확장시켜 얻은 감지상자인 정방형 지역과의 충돌을 감지한다. 감지 상자의 폭은 NPC의 경계반경과 동일하고, 길이는 NPC의 현재 속도에 비례하는데, 속도가 빨라지면 감지 상자는 더욱 길어지게 된다. 장애물 피하기의 장애물 교차부분을 확인하는 과정은 아래 3단계로 그림 4에서 확인할 수 있다. 여기서 x의 양의 축이 NPC의 진행방향이다.

① 장애물의 경계 반경을 감지 상자 폭의 절반으로 확대하고 해당 장애물의 지역 y값이 확대된 반경 값보다 작지 않다면 감지상자와 겹치지 않으므로, 고려 대상에서 제외한다. 그림 4의 A가 바로 이 경우이며 라벨 표시가 없는 빗금친 원은 장애물로 감지된다.

② NPC 자신의 감지 상자의 범위 내에서 감지되었던 장애물들은 감시상자를 벗어난 경우에도 계속 감시 대상으로 남으며 이런 장애물에는 표시를 해 둔다.

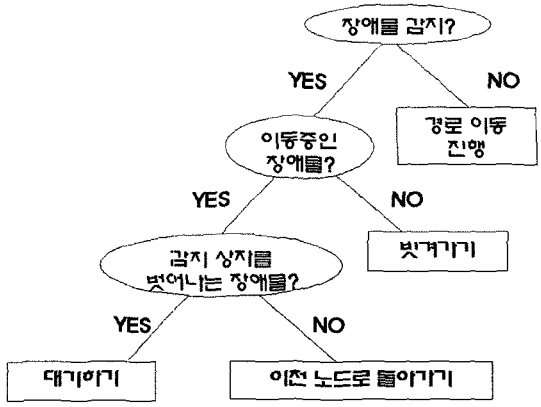

그림 5. 장애물에 대한 의사결정 나무의 예

(그림 4의 B)

③ ②에서 표시된 장애물이 NPC의 중심을 원점으로 한 지역 공간으로 변환 후, 음수 x좌표를 갖게 되는 장애물은 버린다.(그림 4의 C)

3.3 의사결정나무에 의한 선택적 동작

계획 모듈에서 NPC의 목표에 따라 생성된 경로는 실행 모듈에 의해 단순 행동으로 분리되어 실제 이동으로 구현되며 이때, 동적인 장애물이 감지되면 계획된 경로 실행에 수정을 결정하여야 한다. 본 논문에서는 일반화 가시성그래프에 의해 계획된 경로 이동을 시뮬레이션 하기 위해 NPC의 몇 가지 시나리오를 정의하고 의사결정나무(Decision tree)를 이용해 선택되는 시나리오의 종류에 따라 '장애물 피하기'의 조타행동을 각기 다르게 제안해 본다.

의사결정나무는 인공지능의 기계학습 분야에서 분류(classification)에 의한 학습 방법으로 제안된 기법으로, 과거에 수집된 데이터의 레코드들을 분석하여 이들 사이에 존재하는 패턴, 즉 부류별 특성을 속성의 조합으로 나타내는 분류모형을 나무의 형태로 만드는 것이다. 의사결정나무는 순환적 분할(recursive partitioning) 방식을 이용하여 나무를 구축하며 나무의 가장 상단에 위치하는 뿌리노드(root node), 속성의 분리기준을 포함하는 내부노드(internal nodes), 마디와 마디를 이어주는 가지(link), 그리고 최종 분류를 의미하는 잎(leaf)들로 구성된다 (Luger, 2005).

그림 5는 NPC의 시나리오에 따른 의사결정나무의 예를 보여준다. 각각의 NPC들은 특성에 따라 다른 시나리오를 갖게 되며 각각의 시나리오를 따라 장애물에 대한

다른 반응을 보이게 된다.

위 시나리오에 나오는 반응들에 대한 각각의 설명은 다음과 같다.

• 대기하기

진행 중인 NPC의 감지상자에 장애물이 인지되는 경우 장애물이 감지상자에서 사라지기까지 현재 위치에서 대기한다. 이후, 장애물이 더 이상 감지되지 않으면 계획된 경로를 계속 진행한다.

• 빗겨가기

진행 중인 NPC의 감지상자에 인지된 장애물이 움직이지 않는 경우 해당 장애물과 접촉하게 되기까지 진행하여 장애물의 모서리를 따라 이동한다. 모든 장애물들은 원으로 모델링 되므로 동적 장애물의 크기에 맞는 '원 경로 이동' 조타행동에 따라 실행된다. 이때 장애물의 모서리와 계획된 경로가 만나는 시점까지 빗겨가기가 이루어진다.

• 이전 노드로 돌아가기

감지된 장애물이 움직이고 있으면서도 감지상자를 벗어나지 않는 경우. 현재 이동하고 있는 링크의 이전 노드로 돌아가 그 노드를 시작점으로 다시 경로를 계획한다. 이 경우, 장애물이 감지되었던 링크는 이동 가능한 경로에서 제외한다. 또한 경로를 다시 계획함에 있어서 정적인 장애물에 대한 일반화 가시성그래프는 다시 계산하지 않으며 빠른 계산을 위해 선행 연구에서 제안한 대로 RSL(Rotational sweep line)알고리즘 (M.de Berg, 2000)을 일반화 다각형에 확장한 방법을 적용한다.

4. 시뮬레이션 결과

본 절에서는 계획 모듈로부터 전달된 경로 데이터를 받아 정의된 기본 동작들로 경로 이동을 실행하고 여러 종류의 캐릭터를 이동중인 경로 상에 투입 시켰을 때 주어진 시나리오에 따라 다르게 행동하는 NPC의 동작을 시뮬레이션 한다.

4.1 경로 계획 구현

본 논문에서 시뮬레이션 하는 예제 맵은 각각의 장애물이 그림 6과 같이 구성되었다. NPC의 시작점을 원으로 표시하고 목표위치, 즉 도착점을 X로 표시한다. 그리고 NPC의 이동이 실행될 때 등장하는 방해 NPC는 이중선을 가진 원으로 표시한다. 계획 모듈은 사용자에 의해 제작된 맵에 대한 정보를 담고 있는 파일을 불러옴으로써 시작된다. 우선 경로의 잦은 방향전환을 피하기 위해 가로, 세로 5셀 이하의 연속되는 모서리를 대각선화 하고 장애물을 움직이는 NPC와 장애물 사이에 필요한 여유공간 만큼 확장한다. 이와 같이 확장된 장애물에 시작점과 목표점을 추가하여 일반화 가시성그래프를 구축한다. 최종적으로 계획된 경로에 A* 알고리즘을 적용하여 최적을 경로를 찾는다. 이때에는 이동하는 방해 NPC의 위치는 무시한다. 경로계획의 최종적인 결과로 찾아지는 시작점으로부터 목표점까지의 최적의 경로는 그림 6에서 중앙 검은 선으로 나타나 있다.

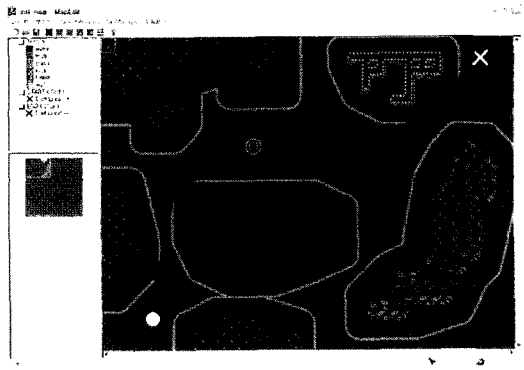

그림 6. 경로 계획 결과 최적 경로

4.2 경로 이동의 실행 구현

실행 모듈은 계획 모듈에 의해 계획된 경로에 따라 캐릭터의 실제적인 이동을 구현하는데 본 논문에서 시뮬레이션하기 위해 설정한 NPC들의 시나리오는 다음과 같다.

• NPC 이동에 대한 예제 시나리오

1. NPC는 현재 위치(원)으로부터 설정된 목표점(X)까지 장애물을 피해 이동한다.
2. NPC의 이동 시, 최적의 경로를 따라 이동한다.
3. NPC의 이동 경로에서 방해 NPC(이중선의 원)를 만나는 경우, 다음의 3가지 방해 NPC에 대해 각각 시뮬레이션한다.
   - 3-1. 일정 시간 후에 이동 NPC의 감지상자 범위를 벗어나는 방해 NPC ⇒ 현재 위치에서 일정시간 동안 기다린다.
   - 3-2. 움직이지 않는 방해 NPC ⇒ 이동 NPC는 방해 NPC를 빗겨간다.
   - 3-3. 일정시간 이후에도 감지상자를 벗어나지 않는 방해 NPC ⇒ 이동 NPC가 이전의 노드로 돌

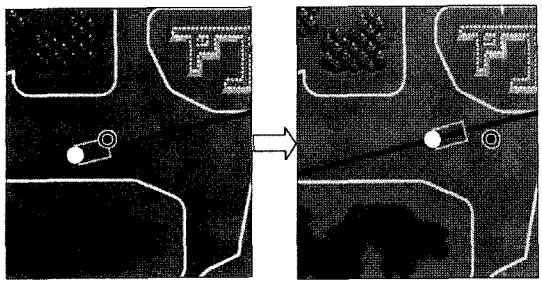

그림 7. 대기후, 계속 진행하는 실행 결과

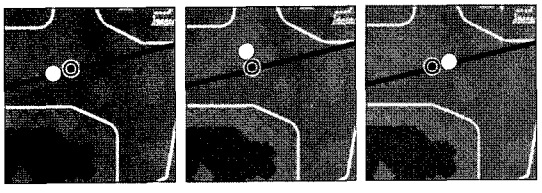

그림 8. 방해 NPC 빗겨가기

아가 목표점까지의 최적 경로를 다시 계산하여 다른 경로를 찾는다.

위와 같은 시나리오를 전제하고 NPC의 이동은 직선 이동과 원 이동, 도착하기의 동작을 조합하여 이동한다. 현재 노드에서 최적 경로에 대한 데이터를 받아 다음 노드까지 이동을 실행한다. 이때, 경로에 대한 데이터 정보에는 시작하려는 링크의 노드가 곡선 링크의 시작 노드인지 직선 링크의 시작 노드인지를 구별하는 정보를 함께 가지고 있다. 따라서 NPC가 이동할 때 다음 노드에 담겨 있는 링크에 대한 정보에 따라 이동법칙을 적용하게 된다. 따라서 다음 노드를 만날 때까지 NPC의 이동 방향에 대한 계산 과정이 생략될 수 있다. 그리고 목표점에 도달했을 경우나 방해 NPC의 부근에 접근하여 감지상자에 정보가 인지되는 경우, '도착하기' 동작을 적용하여 현재 NPC의 속도를 조금씩 줄여감으로써 보다 현실감 있는 이동 모습을 보인다. 이러한 기본 이동 방법으로 각각의 시나리오에 대해 실험된 결과가 아래 그림에 나타나 있다. 그림 7는 방해 NPC가 화살표 방향으로 움직여서 감지상자를 벗어나는 경우로 이때에는 이동 NPC가 잠시 대기하다가 원래 진향 방향으로 이동하는 것을 볼 수 있다. 그림 8은 방해 NPC가 멈춰있어 이동 NPC를 이를 빗겨가는 모습이다.

마지막으로 그림 9는 감지상자의 범위에 방해 NPC가 지속적으로 나타나는 경우 이전에 지나온 노드로 돌아가 해당 노드를 시작점으로 다시 GVgraph를 계산하여 최적

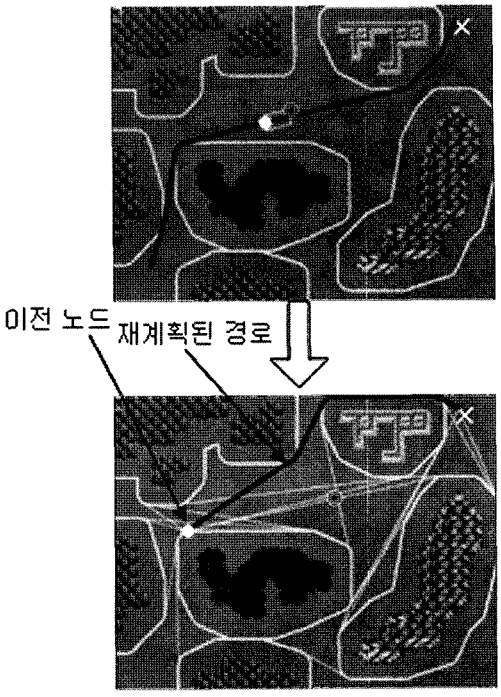

그림 9. 이전 노드로부터 재계획

의 경로를 계산한 결과이다. 이때, 장애물의 경우 정적인 형태를 띠고 있으므로 RSL 알고리즘을 적용하여 새로운 시작점과 목표점의 데이터만 추가하고 방해 NPC에 의해 실패한 링크는 이동 가능한 링크에서 제외하고 계산하여 경로를 빠르게 다시 계획한다. 재계획된 경로는 아래 그림에 검은 선으로 나타나 있다.

5. 결 론

본 논문에서는 게임에서 NPC들의 자연스럽고 안전한 경로 이동을 효율적으로 구현함으로써 현실감을 더 할 수 있는 패스 파인더의 실행 모듈을 구축하였다. 실행 모듈의 입력은 일반화 가시성그래프 위에서 A* 알고리즘을 적용하여 얻은 경로 데이터인데 이 경로는 직선과 곡선의 조합으로 이루어져 사람의 움직임과 유사한 자연스러운 모습이다. 뿐만 아니라 곡선과 직선형태의 단순한 경로를 반복 생성하므로 적은 수의 조타행동으로도 캐릭터의 이동 표현이 가능하며 방해물을 만났을 경우에도 이전 성능의 저하 없이 목표점까지 이동을 계속할 수 있었다. 움직이는 방향이 바뀌어야 하는 연결점에 가까워졌을 경우에는 현재의 캐릭터에 가해지는 힘을 감소하여 속도를 줄임으로서 현실감 있게 보이는 효과를 주었

다. 이동을 하면서 감지 상자를 이용하여 진행 방향에 나타나는 방해 NPC를 감지할 수 있도록 충돌 감지 기능을 추가했으며 충돌이 감지되는 경우에는 방해 NPC의 여러 가지 경우에 따라 의사결정나무를 이용하여 다르게 대응하도록 하였다.

본 논문에서 실험한 예의 경우, 하나의 NPC와 지정된 좁은 경로를 이동하는 방해 NPC, 정적인 장애물을 고려하여 이루어졌다. 하지만 실제 게임에서는 보다 복잡한 환경 하에 더 많은 수의 NPC들이 구성되어 있다. 그러므로 향후에는 미리 계획된 NPC들에 대한 반응 외에 미리 계획되지 않은 사용자에 의해 움직이는 캐릭터의 동적인 반응에 대처하는 실험이 필요하다. 이를 위하여 의사결정나무를 이용하여 미리 계획되어 있지 않던 시나리오에 대해 학습하고 스스로 의사결정나무를 구축할 수 있는 기능을 갖도록 연구를 수행할 계획이다.

참 고 문 헌

1. M. de Berg et al. (2000), Computational Geometry :Algorithms and Applications 2nd edition, Springer Verlag.
2. Laird, J.E. and van Lent, M. (2000), "Human-level AI's Killer Application", *Interactive Computer Games*. AAAI Press.
3. J.C. Latombe (1991), Robot Motion Planning, Kluwer Academic Publishers.
4. Laumond, J.P. (1987), "Obstacle growing in a nonpolygonal world" *Inform. Proc. Letter*, vol. 25(1), pp 41-50.
5. L. Liden (2002), "Strategic and Tactical Reasoning with Waypoints", *AI Game Programming Wisdom, Edited by Steve Rabin*, Charles Rive Media, pp. 211-220.
6. Luger, G. and Stubblefield, W (2005), Artificial Intelligence:Structures and Strategies for Complex Problem Solving. fifth edn., Addison Wesley Longman Inc.
7. C. W. Reynolds (1999), "Steering Behaviors for Autonomous characters", Game Developer Conference.
8. J. Rossignac and A.G. Requicha (1986), "Offsetting operations in solid modelling", *Computer Aided Geometric Design*, vol. 3, pp. 129-148.
9. Snook, G (2000). "Simplified 3D Movement and Pathfinding Using Navigation Meshes," *In: Deloura, M. (eds.): Game Programming Gems*, Charles Rive Media, pp. 288-304.
10. Tozour, P. (2004), "Search Space Representations" *In: Rabin*, S. (eds.): AI Game Programming Wisdom 2, Charles Rive Media, pp. 85-102.
11. S. Woodcock (2000) "Game AI: The state of the industry", *Game Developer Magazine*, August.
12. P. Yap (2002), "Grid-based path-finding", *Lecture notes in Artificial Intelligence*, vol. 2338, pp. 44-55.
13. Young, T. (2001), "Expanded Geometry for Points-of-Visibility Pathfinding", *In: Deloura, M. (eds.): Game Programming Gems 2*, Charles Rive Media, pp. 317-323.
14. Yu, K. (2006), "Finding a Natural-Looking Path by Using Generalized Visibility Graphs", *Trends in Artificial Intelligence*, LNAI 4099, pp. 170-179.

유 견 아 (kyeonah@duksung.ac.kr)

1986년 서울대학교 공과대학 제어계측공학과 학사
1988년 서울대학교 공과대학 제어계측공학과 석사
1995년 미국 USC Computer Science 박사
1996년~현재 덕성여자대학교 컴퓨터공학부 교수

관심분야 : 인공지능, 로봇 알고리즘, 연산 기하학

전 현 주 (hzoo@duksung.ac.kr)

2004년 덕성여자대학교 컴퓨터과학부 학사
2006년 덕성여자대학교 전산 및 정보통신대학원 석사
2006년~현재 덕성여자대학교 교양 강사

관심분야 : 인공지능, 게임 프로그래밍