일반화 가시성그래프에 의한 생성 경로를 효과적으로 구현한 조타행동

전현주° 유견아
덕성여자대학교 컴퓨터공학부
{hzoo°, kyeonah}@duksung.ac.kr

Steering behaviors that efficiently implement paths generated by Generalized Visibility Graph

Hyunjoo Jeon°, Kyeonah Yu
Dept. of computer engineering, Duksung Women's University

요 약

게임의 장르가 다양해지고 대규모의 온라인 게임이 가능해 짐에 따라 게임 환경과 더불어 게임 안에 등장하는 캐릭터의 수도 많아지고 있다. 게이머에 의해 움직이는 캐릭터 외에도 여러 종류의 다양한 NPC(Non-Player Character)들이 각각 맡은 임무를 띠고 각기 움직이게 된다. 본 논문에서는 NPC들의 자연스러운 이동을 위해 제안된 일반화 가시성그래프를 이용하여 계획된 경로의 효과적인 구현을 위한 조타행동(steering behaviors)과 경로 이동 중에 만나는 다른 캐릭터에 대한 NPC들의 조타행동을 정의하고 구현하여 이동에 따른 효율을 실험해 본다.

1. 서 론

게임에서 게이머가 조작할 수 없으며 컴퓨터에 의해 자동으로 움직이고 말하는 캐릭터를 NPC(Non-Player Character)라고 한다. 보통 게이머 캐릭터의 적대역인 적대자 캐릭터(몬스터, 적군 등)와 반대로 우호적인 보조캐릭터(상점주인, 안내자, 애완동물 등)로 나뉜다[1]. 가상 세계 안에 항상 존재하면서 유저들과 상호작용을 해주는 NPC들을 통해 싱글게임에서도 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. 경로가 설계자에 의해 임의로 미리 지정되어 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 수행해야할 것이다[2].

가시성 그래프(Visibility Graph, Vgraph)는 장애물의 볼록 정점을 노드로 하고 서로 보이는, 즉 장애물과 교차하지 않는 두 정점 사이의 연결선을 링크로 하는 그래프로 탐색공간의 크기를 줄일 수 있는 방안으로 로봇의 경로 계획을 위해 널리 이용되며 최적의 답을 찾아낸다는 점 뿐 아니라 Vgraph 자체가 로드맵의 역할을 하여 A* 알고리즘을 이용하면 최단거리의 경로를 구할 수 있음이 증명되어 있다[3]. 그러나 Vgraph의에 의해 생성된 경로는 장애물의 한 정점에서 다른 장애물의 정점으로 이동하거나 장애물의 경계선을 따라가는 것으로 게임에서 캐릭터가 움직이는 방식으로는 적합하지 않기 때문에 보다 자연스럽고 안전한 이동 경로를 확보하기 위해 일반화 가시성 그래프(Generalized Vgraph, GVgraph)에 의한 경로 계획 방법이 제안된 바 있다4][5]. GVgraph는 장애물 위에 직접 Vgraph를 만드는 것이 아니라 장애물에서 일정 오프세팅을 두고 생성되는 것이기 때문에 GVgraph 상에서 계획된 경로는 장애물에서 일정 거리 이상 떨어지며 직선과 원의 호로 구성되게 된다.

본 논문에서는 GVgraph에 의해 생성된 경로위에서 NPC의 이동이 효과적이고 사실적으로 보이게 하기 위한 조타 행동을 정의하고 구현한다. GVgraph에서 계획된 경로는 직선과 원의 호로 구성되기 때문에 직선운동과 호를 궤적으로 하는 곡선운동이 기본이 되며 이동중 만나는 다른 캐릭터을 피하는 조타행동이 필요하다. 각 조타행동을 효과적으로 구현하는 방법을 설명하고 실제 애니메이션 해봄으로써 운동에 대한 실제적 효율성을 살펴볼 것이다.

2. 일반화 가시성 그래프에 의한 경로 계획

동적인 캐릭터의 크기를 고려한 경로 계획을 위해 캐릭터를 반지름 r의 원형으로 모델링한 후, 로봇 운동계획에서 로봇을 점으로 치환할 때 환경을 구성하는 장애물들을 그에 맞게 매핑한 공간인 형상공간(C-공간) 문제로 변환한다.

그림 1과 같이 원래의 문제를 C-공간 문제로 변환하기 위해서는 다각형을 원형 장애물의 반지름만큼 확장하는 것이 필요하다. 이와 같은 확장 과정은 *Minkowski* 합과 차 연산의 특별한 경우로 정의되는 양의 솔리드 오프세팅(positive solid offsetting) 연산을 적용하면 된다[6]. 볼록 정점의 경우에는 호를 형성하고 오목한 경우에는 정점으로 변환된다.

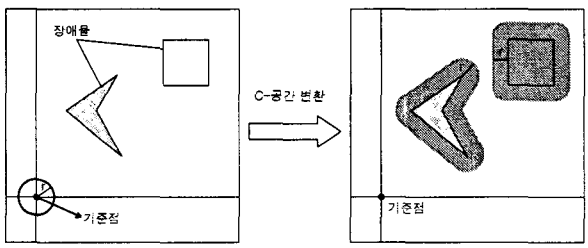


그림 1. 솔리드 오프세팅

일반화 다각형은 원의 호와 다각형으로 이루어진 도형군이다. GVgraph는 C-공간상에 시작점, 끝점과 함께 일반화 다각형으로 이루어진 장애물들이 배치되어 있는 경우에 생성된다. 일반화 다각형에서는 호를 포함하기 때문에 Vgraph에 비해 특수한 경우를 포함한다. 먼저 Vgarph와 마찬가지로 노드에는 시작점, 목적점, 장애물의 정점을 포함한다. 호의 경우에는 호의 가시성을 따져주기 위해 가상점(fictitious point)을 생성한다. 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 2와 같으며 그래프에 A* 알고리즘이 사용되며 과소추정된 휴리스틱으로 그래프의 최단 경로를 찾는 것을 보장한다.

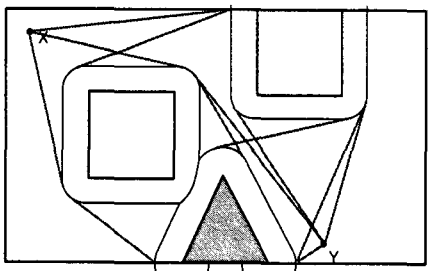


그림 2. 일반화 가시성 그래프

3. 조타행동(Steering Behavior)

조타행동은 NPC의 이동을 지능적이고 매끄럽고 자연스럽게 보이게 하는 데 중요한 역할을 한다[7]. NPC의 환경에 대한 정보를 받고 가속도를 계산함으로 NPC가 어떠한 단순한 목표(나아간다거나 장애물을 피한다거나 대형을 유지하는 등)를 실행하게 만든다. NPC는 목표(어딘가로 이동하는 이유)를 가져야 하며 거기에 도달하는 방법도 있어야 한다. 따라서 NPC의 목표를 택하고 그것을 이용해서 조타행동을 결정하고 또한 NPC가 나아갈 또는 피할 위치를 결정하는 어떠한 고수준의 AI 시스템이 필요하다. 따라서 조타행동은 게임 AI에 대한 강력한 추상을 제공한다고 할 수 있다. 고수준 AI 시스템은 NPC 이동의 저수준 세부사항들과 NPC의 환경을 알 필요가 없이 그냥 목표 위치를 지정하고 거기에 도달하기 위한 조타행동을 선택하기만 하면 된다. 그 나머지는 조타행동들이 처리한다. 단순한 조타행동들은 합쳐져 좀 더 복잡한 행동을 만들 수 있다.

본 논문에서 일반화 가시성 그래프에 의해 생성된 경로를 구현하기 위해 정의한 조타행동의 종류는 아래와 같다.

3.1 직선 및 원의 호 따라가기

직선 운동은 직선이 시작되는 노드와 다음 직선이나 원이 시작되는 노드(앞의 직선이 끝나는)의 좌표를 통해서 해당 직선의 방향 벡터를 계산하고, 일정한 속도(가속도=0)를 유지하면서 현재 위치에 앞에서 계산한 방향벡터를 적용하여 시간에 따라 다음 위치로 이동한다.

원 운동의 경우 일반적으로 알고 있는 $x^2 + y^2 = r^2$이라는 원의 방정식을 프로그래밍에 사용할 경우 픽셀로 그려지는 컴퓨터 모니터화면에서는 정확히 표현할 수가 없다. x값에 따라서 y값이 정의되기 때문에 y값은 많이 변하지만 x값이 거의 변하지 않는 경우인 원의 맨 좌측과 맨 우측에는 정확히 그려질 수가 없다. 이것을 표현하기 위해 원의 방정식보다는 원의 중점에 대한 식, 즉 x = r cos(theta), y = r sin(theta)를 이용하여 세타의 값을 변화시킴으로 원운동의 이동을 표현한다. 특별히 사람의 이동과 흡사한 모습을 위해 직선운동의 경우보다 느린 속도로 이동하게 한다.

3.2 도착하기(Arrive)

보통 목표 위치에서는 NPC가 사람과 마찬가지로 부드럽게 정지하기를 원하는데, 도착하기는 NPC를 목표 위치로 감속해가는 방식으로 조타하는 행동이다[8]. NPC가 목표에 도달하는데 얼마나 많은 시간을 쓰길 원하는가를 계산하기 위해 이 값을 사용한다. 이 값을 통해 NPC가 목표까지 원하는 시간 내에 도달하기 위해 얼마만한 가속도로 가야하는지를 계산할 수 있다. 이후, 계산 되어진 가속도를 현재 NPC의 속도 벡터와 목표 위치에 도달하기 위한 속도사이의 계산에 적용하여 감속하는 행동을 보이게 된다.

3.3 장애물 피하기(Obstacle avoidance)

장애물 피하기는 이동하는 NPC의 경로에 놓여 있는 장애물들을 피하도록 조타하는 행동으로 NPC의 앞쪽으로 확장시켜 얻은 감지상자인 정방형 지역 내에서 충돌이 없게 한다[8]. 여기서 장애물이란 원을 이용하여 비슷하게 만들 수 있는 모든 물체이다. 장애물 피하기는 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 3에서 확인할 수 있다.

① NPC는 자신의 감지 상자의 범위 내에 있는 장애물만을 고려한다. 반복적으로 게임 세계 내의 모든 장애물들을 검사하면서 좀 더 고려해야 하는 범위 내의 장애물에는 표시를 해둔다.(그림 3의 A)

② ①에서 표시된 장애물을 NPC의 중심을 원점으로 한 지역공간으로 변환 후, 음수 지역 x좌표를 갖고 있는 장애물을 버린다.(그림 3의 B)

③ 장애물의 경계 반경을 감지 상자폭의 절반으로 확대하고 해당 장애물의 지역 y값이 확대된 반경값보다 작지 않다면 감지 상자와 겹치지 않으므로, 고려 대상에서 제외한다.(그림 3의 C)

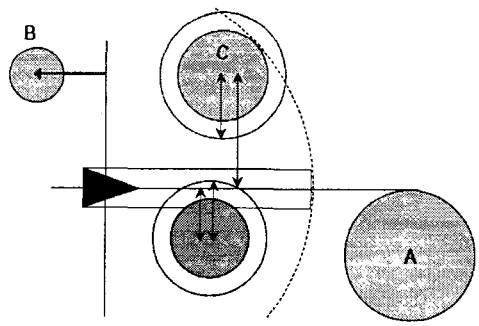


그림 3. 장애물 피하기의 A, B, C단계

4. 시뮬레이션 결과

본 논문에서 GVgraph에 의해 생성된 경로위에 실제로 이동하는 NPC의 효율적인 움직임을 위해 GVgraph에서 나타나는 몇 가지 특징을 이용해 실험하고 조타행동의 도착하기와 장애물 피하기를 적용하였다. 셀 기반 지형을 생성하기 위한 맵-에디터는 VC++를 이용해 제작하였다. 제작된 맵은 위치정보와 각 위치에 따른 셀의 정보를 맵 데이터로 저장하게 되며, 저장된 맵 데이터의 값에 따라 셀 테이블의 데이터들을 호출한다. 여기서 셀 테이블은 각 셀의 비트맵에 따른 고유 플래그를 저장하고 플래그의 종류에 따라 이동가능 여부를 결정하게 된다.

그림 4는 맵 에디터 위에 실행된 결과이다. 원형 중 빨간 삼각형이 있는 것은 출발점을 나타내는 것으로 NPC의 이동 방향을 삼각형의 꼭지점이 나타내며, 파란 원이 있는 것은 목표점을 나타내고 각 원의 반경은 r이라고 하자. NPC가 움직이는데 필요한 여유공간 c를 움직이는 NPC의 반지름 r의 2배, 즉 2r로 설정한다면 노란선 같이 원래 장애물들은 2r만큼 경계선이

확장되어 그 위에 Vgraph가 생성된다. A* 탐색 알고리즘으로 찾아낸 경로가 장애물 사이에 굵은 하늘색 선으로 표시된 것이다. 또한 빨간 삼각형을 안고있는 NPC가 하늘색 선을 따라 이동하는 중 화살표의 방향으로 왕복 이동하는 또 다른 NPC에 의해 진로에 방해를 받게 된다고 가정한다.

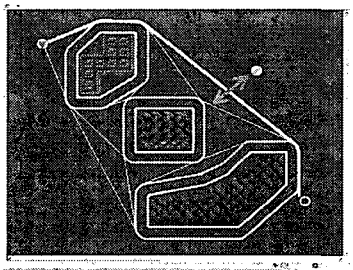


그림 4. GVgraph에 의한 경로계획 결과

이제 시작점에 위치한 NPC를 목표점까지 이동하는 실험을 시작한다. 위의 그림 4처럼 일반화 가시성그래프에 의해 계획된 경로는 직선과 곡선 형태만을 보인다. 따라서 A*알고리즘을 실행하기 전 노드들에 대한 정보를 저장할 때, 현재 노드에 연결된 선분이 직선의 형태를 가지는지, 곡선의 형태를 가지는지 미리 표시한다.

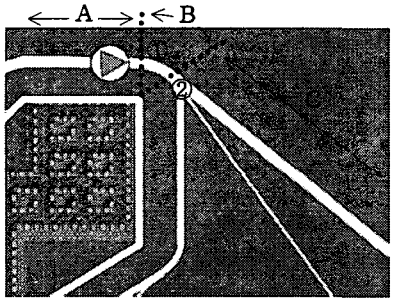
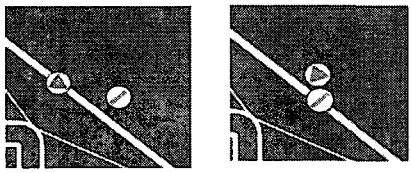
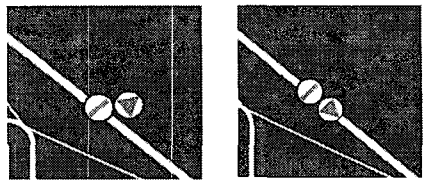


그림 5. 이동 중 곡선경로 확대 그림

그림 5의 A영역과 같이 직선경로를 이동할 때에는 다음 노드①을 만날 때 까지 NPC의 이동 방향은 변하지 않는다. 따라서 NPC는 처음 계획되어진 방향벡터대로 속도를 유지하면서 이동한다. 이제 B영역과 같이 원 운동을 하면서 방향이 변하는 노드①을 만나게 되면 노드①에 저장되어 있는 정보를 이용해 방향과 운동 방법이 변하게 되는 것을 인지하고, 다음 노드를 만날 때까지 원운동에 따른 방향벡터의 변환을 계산하여 이동한다. 이때 보통 사람의 원운동 시, 속도를 줄여가며 이동한다는 것을 고려하여 직선운동의 속도와 차이를 두며 실험한다. 마찬가지 방법으로 원운동이 끝나고 노드②를 만나게 되면 노드②에 저장된 정보를 통해 이후 나타나는 경로가 직선형태 임을 인지하고 일정한 방향으로 속도를 유지하며 이동하게 된다.



(a) (b)



(c) (d)

그림 6. 장애물 피하기 실행결과

그림 6은 앞 장에서 언급했던 장애물피하기 조타행동을 적용한 결과이다. 방해 NPC가 나타났을 때, 해당 NPC의 형태에 따라 원운동을 실행하며 잠시 정해진 경로를 이탈했다가 돌아오는 모습을 보이고 있다. 이 경우 방해 NPC의 반경에 대한 정보나 현재 위치좌표, 그리고 NPC들이 원의 형태를 가지고 있다는 가정 하에 원운동을 하게 된다.

5. 결론

본 논문에서는 로보틱스 분야에서 운동계획을 위해 많이 이용되는 가시성그래프를 일반화 다각형 환경으로 확장한 일반화 가시성그래프 위에서 생성된 경로를 효과적으로 구현하기 위한 조타행동을 정의하고 구현하였다. 일반화 가시성그래프는 게임의 캐릭터들에게 자연스럽고 최적에 가까운 경로를 제공하는 것이 장점인데 이를 최대한 살릴 수 있도록 경로 따라가기를 구현하였다. 뿐만 아니라 계획된 경로를 따라 가다가 움직이는 다른 캐릭터를 만나는 경우에도 성능을 저하시키지 않고 처리할 수 있도록 하였다. 일반화 가시성그래프에 의해 생성된 경로의 특징상, 적은 수의 조타행동만을 정의해도 움직임을 사실적으로 표현할 수 있었으며 같은 방식으로 움직인다는 가정하에 다른 캐릭터의 움직임을 감지하고 피할 수 있었다. 본 논문에서는 정적인 장애물만을 가정하고 같은 성질의 캐릭터에 대한 대응만을 고려하였으나 향후에는 임의의 동적인 장애물을 피하도록 하는 조타행동을 정의하는 것이 필요하다.

6. 참고문헌

[1] 이재홍, "게임시나리오 작법론", 도서출판정일, 2004
[2] S. Woodcock, "Game AI: The state of the industry", Game Developer Magazine, August, 2000.
[3] J.C. Latombe, "Robot Motion Planning", Kluwer, Academic Publishers, 1991.
[4] 유견아, 전현주, "컴퓨터 게임환경에서 일반화 가시성그래프를 이용한 경로찾기", 한국시뮬레이션 학회 논문지 14(3), 2005.
[5] 전현주, 유견아, "가시성그래프에 의해 최소 여유공간을 보장하는 길찾기", 한국정보과학회 추계학술발표회 논문집, pp.739-741, 2005.
[6] J. Rossignac and A.G. Requicha, "Offsetting operations in solid modelling", Computer Aided Geometric Design, vol. 3, pp129-148, 1986.
[7] Steve Ravin, "AI Game Programming Wisdom2", Charles River Media, 2003.
[8] Buckland Mat, "Programming Game AI by Example", Natl Book Network, 2004.