3D 게임지도에서 에이전트 이동을 위한 경로표 활용

심 동 희†

요 약

게임지도상에서 에이전트 이동을 위한 경로탐색에 A*를 이용하는 경우는 실시간 게임진행에 지장을 주고 있다. 이러한 문제를 해결하기 위하여 2차원 게임에서 사용될 수 있는 경로표가 제안된 바 있다. 이 경로표를 게임개발시에 미리 작성하여 게임진행시에 이를 이용한다. 본 논문에서는 2차원게임을 위하여 설계된 경로표를 3차원 게임에서 사용할 수 있도록 이를 확장하였다. 이러한 경로표를 이용하는 경우 메모리를 많이 필요로 하는데 같은 경로표를 생략하는 방법을 이용해 데이터압축효과를 얻을 수 있음도 보여주었다.

Utilization of the Route Table for the Agent's Move in the 3D Game Map

Dong Hee Shim†

ABSTRACT

The use of the A* for the path search of the agent in the game map gives the computational overhead in real time game processing. To solve this problem, the route table was presented for the 2D game. This route table is made in the game development phase and utilized in game playing. The route table designed for 2D game is extended for 3D game in this paper. But the momory space is required too much. This problem can also be solved using the data compression by skipping the duplicated route table.

Key words: game map, route table

1. 서 론

게임은 장르에 따라 액션게임(Action Game), 전략게임(Strategy Game), 어드벤춰게임(Adventure Game), 스포츠게임, 시뮬레이션게임 등으로 구분되며 화면상의 플레이 투시기법에 따라 1인칭 투시화법(First-Person Perspective), 3인칭 투시화법(Third-Person Perspective), 하향(Top-Down)투시화법, 쿼터뷰(Quater View) 투시화법, 측면(Side-View)투시화법 등으로 구분된다[1-3]. 1인칭 투시화법은 에이전트의 시각을 통하여 플레이환경을 제공하는 기법이며, 3인칭 투시화법은 주인공의 에이전트를 직접 화면에 등장시켜 진행하는 기법이며, 하향 투시화법에서는 카메라를 하늘에 설치하여 지상을 비추는 것과 같은 기법이며, 쿼터뷰 투시화법은 3/4 정도로 비스듬하게 화면을 표현하는 기법이며, 측면 투시화법은 전통적인 액션게임에서 많이 사용되었는데 지금은 잘 이용되지 않고 있다. 그리고 게임은 이미지의 처리방식에 따라 2D와 3D로 대별된다[2,4]. 현대 게임의 구성요소중 기본기술은 에이전트의 이동기술이며, 그 비중이 점차 커져가는 요소는 3차원 그래픽, 네트워크, 가상현실 기술이다[3,5,6]. 에이전트의 이동은 거의 모든 게임에서 기본적이지만 특히 전략게임, 스포츠게임에서 중요한 요소가 된다. 이런 유형은 투시화법으로 대부분 3인칭 투시화법을 사용하게 된다. 에이전트의 이동은 이동을 가로막는 블록영역이 있는 경우와 블록영역이 없는 경우로 나누어 볼 수 있다. 에이전트의 이동은 다시 움직이는 방향에 따라 4방향, 8방향 등으로 분류할 수 있다. 그런데 블럭영역이 없는 경우에 에이전트가 주어

접수일 : 2002년 10월 4일, 완료일 : 2002년 11월 7일
† 정회원, 전주대학교 정보기술컴퓨터공학부 교수

진 게임지도상에서 위치를 이동할 때 좌표를 이용하여 직선으로 이동할 수 있으므로 경로탐색이 필요하지 않다. 그러나 블럭영역이 있는 경우에는 에이전트는 이동을 위한 경로를 탐색해야 한다. 이러한 블럭영역은 다시 게임의 실행중에 동적으로 위치가 변하는 경우와 정적으로 고정되어 있는 경우로 나누어 볼 수 있다. 그런데 대부분의 게임에는 이러한 블럭영역이 정적이든 또는 동적이든 존재하기 때문에 경로탐색이 필요하다. 그래서 에이전트의 지능적인 이동을 위해서는 이러한 경로탐색이 효과적으로 이루어져야 한다. 상용화되어 있는 많은 게임에 이러한 경로탐색을 위하여 A* 알고리즘[7]이 이용되고 있는 것으로 알려지고 있다. 그런데 최적의 경로를 탐색하기 위한 A*와 같은 알고리즘은 수행시간을 많이 요구하기 때문에 실시간을 위한 게임에는 다소 부적합하다. A*가 갖는 수행시간에 대한 이러한 단점으로 인하여 많은 비디오 게임에서는 최적경로에 근접한 경로를 구하기 위해 게임 도메인의 지식으로부터 도출한 함수를 이용하여 Greedy 탐색방법을 사용하고 있다[8-10]. 그러나 이 탐색방법은 최적경로를 도출해 주지 않는다는 단점을 갖고 있다[11]. 다른 게임도 어떤 유형의 탐색방법을 이용하고 있지만 게임내부의 프로그램 및 그 사양을 상업적인 이유로 인하여 공개하지 않고 있기 때문에 정확히 어떤 탐색을 이용하는지 모르는 경우가 많다. 블록영역이 동적으로 변하는 경우에는 게임진행중에 A*나 Greedy 탐색을 이용하여 경로를 탐색할 수 밖에 없지만 블록영역이 정적인 경우에는 게임 개발시에 에이전트의 이동을 위하여 최적경로를 미리 작성하여 실시간게임진행에 활용할 수 있을 것이다.

이러한 문제를 해결하기 위하여 블록영역이 정적인 경우 4방향으로 이동할 수 있는 2D 게임에서 실시간게임진행에 지장을 주지 않으면서 최적경로를 구하여 이용할 수 있는 방안이 제안된 바 있다[12]. 이 방법은 2D에서 대각선의 이동은 고려하지 않은 4방향의 이동에 대한 것이었다. 그래서 이 방법에서는 먼저 주어진 게임지도상에 존재하는 모든 위치들간의 최적경로를 미리 탐색한 후 이 최적경로를 게임에서 바로 이용함으로써 수행시간을 줄이게 된다. 그리고 모든 최적경로를 저장하는데 소요메모리를 최소화하기 위하여 중복되는 경로는 하나로 저장하는 방법을 제안하고 있다. 본 논문에서는 이 방법을 2D의 4방향 이동에서 3D의 6방향 이동의 경우로 확장하여 3D게임에서 사용할 수 있도록 하고자 한다.

이 논문의 2장에서는 최적경유필수위치 개념을 소개하고 이에 기반한 경로표를 제안하였으며, 또한 이를 이용한 에이전트의 이동알고리즘을 설계하였다. 3장에서는 경로표를 메모리에 효율적으로 저장하기 위한 경로표 압축방법을 제시하였다. 4장에서는 경로표의 성능평가를 위하여 A*를 이용하는 경우와 경로표를 이용하는 경우에 대한 비교를 하였다.

2. 게임지도상의 경로와 이동

2.1 최적경로와 최적경유필수위치

게임이 수행되는 지도상의 모든 위치는 이동을 제어하기 위하여 식별자를 가져야 하는데 이 식별자는 게임지도상의 좌표를 사용하는 것이 바람직하다. 그래서 게임에서는 보통 픽셀단위로 부여하는 좌표를 이용하여 위치를 식별하게 된다. 또한 이러한 좌표를 이용할 때 에이전트는 상하좌우전후 6방향으로 이동할 수 있게 된다. 3차원 게임의 경우 현위치가 (a, b, c)이면 이동가능위치는 (a+1, b, c), (a, b+1, c), (a, b, c+1), (a-1, b, c), (a, b-1, c), (a, b, c-1)이 된다. 그런데 현위치에서 목표위치로 이동하는 최적경로는 이러한 지도좌표상에서 도출이 된다.

게임지도에서 각 경로의 이동비용은 보통 똑 같다. 다만 에이전트가 접근할 수 없는 영역(경로비용이 ∞에 해당)이 있는데 이를 블록영역이라 한다. 만일 어떤 게임지도에서 블록영역이 없다면 최적경로는 좌표에 의하여 1회의 계산에 의하여 산출될 수 있다. 예를 들어 현위치 (a, b, c)에서 목표위치 (d, e, f)로의 이동을 위한 최적경로는 직선상의 이동을 하면 되는 것이므로 표 1에 나타낸 바와 같이 모두 6가지의 경로가 있다. (a, b, c)에서 표 1에 열거된 경로중 하나를 통하여 (d, e, f)에 도달할 수 있다. 이러한 경로상에 있는 위치를 최적경유필수위치라고 하자. 즉 게임지도상의 경로를 서로 연결하여 작성한 선분에서 같은 선분에 있는 끝 위치를 최적경유필수위치라고 하자. 2D의 4방향, 3D의 6방향 게임에서 최적경로가 주어지면 이 최적경로를 구성하는 각 선분의 끝위치가 바로 최적경유필수위치에 해당한다. 대분분의 게임지도에는 블록영역이 있기 때문에 최적경로를 도출하기 위한 탐색이 필요하게 된다. A*

표 1. (a, b, c)에서 (d, e, f)로 이동시 이동경로

처음 이동좌표	두 번째 이동좌표
(d, b, c)	(d, e, c)
(d, b, c)	(d, b, f)
(a, e, c)	(d. e. c)
(a, e, c)	(a, e, f)
(a, b, f)	(d, b, f)
(a, b, f)	(a, e, f)

와 같은 알고리즘을 이용하여 최적경로를 도출하면 이 최적경로는 유한한 개수의 선분으로 구성된다. 이때 각 선분의 끝점은 최적경유필수위치에 해당하게 된다.

2.2 에이전트의 최적경로 도출방법

게임진행시 에이전트가 이동하기 위해서는 최적경로를 구해야 한다. 앞서 서론에서도 설명한 바와 같이 게임의 진행시에 최적경로를 구하면 실시간 진행에 부담을 주기 때문에 바람직하지 않다. 이 단점을 해결하는 방법으로 게임 설계시에 각 위치마다 모든 목표위치에 대해 최적경로를 미리 작성하여 이를 저장한 후 게임시에 이를 이용할 수 있다. 이 방법은 게임의 실시간 진행에는 거의 부담을 주지 않는 방법이다. 그러나 모든 위치에서 모든 위치로의 최적경로를 저장하는 것은 많은 메모리를 요구하기 때문에 이 또한 적절한 방법이 될 수 없다. 이러한 메모리 부담을 경감할 수 있는 방법으로 최적경로를 모두 기억시키지 않고 선분의 단위로만 기억시키는 방법을 활용할 수 있다. 그래서 각 위치마다 목표위치에 대한 최적경유위치를 기억시켜 이를 반복적으로 이용하면 이러한 메모리 부담을 경감할 수 있다.

2.3 경로표

지도상의 모든 위치는 표 2와 같은 경로표를 갖도록 한다. 표 2에서 목표위치는 현위치에서 이동하고자 하는 위치이며 최적경유필수위치는 목표위치로 이동하기 위한 최초의 최적경유필수위치를 나타낸다.

표 2. 경로표

목표위치	최적경유 필수위치
(x1, y1, z1)	(x2, y2, z2)
.	.
.	.

2.4 최적경유필수위치의 성질

표 2에 표기된 최초의 최적경유필수위치는 현위치에서 최적경유필수위치로 가는 경로가 바로 계산될 수 있도록 일직선상에 위치해야 한다. 즉 최적경로가 도출되면 이 최적경로를 선분단위로 분할하여 각 선분의 끝위치가 최적경유필수위치가 되게 해야 한다. 이렇게 생성된 최적경유필수위치는 다음과 같은 성질을 갖게 된다.

(1) 목표위치가 현위치와 같은 선분에 있는 경우

목표위치가 현위치와 같은 선분에 있으면 최적경유필수위치는 목표위치와 동일하게 된다.

(2) 최적경로가 선분 n개로 구성되는 경우

최적경로가 선분 n개로 구성되면 최적경유필수위치는 n개(각 선분의 끝)가 되며 경로표에는 이중 최초의 것을 나타내는 것이다.

(3) 목표위치도 최적경유필수위치에 해당

목표위치도 마지막 최적경유필수위치에 해당한다고 볼 수 있다.

2.5 에이전트의 이동 알고리즘

에이전트는 자신의 위치를 알 수 있으며, 목표위치도 기억하고 있으며, 상, 하, 좌, 우, 앞, 뒤 6방향으로만 이동할 수 있는 환경을 고려하기로 하자. 에이전트가 현위치에서 목표위치로 이동하기 위하여 다음과 같은 알고리즘을 이용한다.

이동알고리즘

① 현위치가 목표위치인지 확인한다. 목표위치면 단계⑤로 간다.

② 현위치의 경로표에서 목표위치에 대한 최적경유필수위치를 가져온다.

③ 현위치에서 최적경유필수위치로 이동하고 현위치를 최적경유필수위치로 갱신한다.

④ 단계 ①로 간다.

⑤ 종료한다.

즉 에이전트는 현위치에서 목표위치로 이동하기 위하여 현위치에 대한 경로표에서 목표위치의 최적

경유필수위치를 찾아서 최적경유필수위치로 이동한다. 현위치에서 최적경유필수위치로의 이동을 위한 경로는 실행시간에 대한 부하없이 좌표에 의한 1회의 계산에 의해 바로 결정될 수 있다. 에이전트는 이 최적경유필수위치를 새로운 현위치로 간주하며 목표위치에 도달했는지 확인하여 목표위치가 아니면 다시 목표위치로의 이동을 경로표를 이용하여 반복적으로 수행하게 된다.

2.6 경로표의 생성

경로표를 작성하기 위해서는 A* 알고리즘[3]을 함수로 이용하여 다음과 같은 경로표작성 알고리즘을 이용하였다.

(1) 경로표작성 알고리즘

```
node = 지도상의 처음 지점
do
{
node1 = 지도상의 처음 지점
do
{
path[] = AStarSearch(node,node1)
length = LengthOfPath(path[])
pathlength = 1
while(IsParallel(node,path[pathlength]))
{ pathlength++ }
pathlength --
Route-Table[node][node1] = path[length]
node1 = 지도상의 다음 지점
}until (node1 != 지도상의 마지막 지점)
node = 지도상의 다음 지점
} until (node != 지도상의 마지막 지점)
```

AStarSearch는 주어진 첫 번째 노드와 두 번째 노드 사이의 최적경로를 배열로 반환하는 함수이다. LengthOfPath는 주어진 경로의 길이를 반환하는 함수이다. ShortestPath는 주어진 첫 번째 노드와 두 번째 노드 사이에 장애물이 없다고 가정했을 때 최단거리(대각선은 제외)를 반환하는 함수이다. IsParallel은 노드의 x좌표, y좌표, Z좌표가 같으면 true을 반환하고 다르면 false를 반환하는 함수다.

(2) A*에서의 휴리스틱 함수

A* 알고리즘에서 사용한 휴리스틱함수 h'는 (두 지점간의 거리) + 3*Max(두 지점간의 직선거리 사이에 포함된 블록영역 거리)를 사용한다. 이러한 함수를 사용하면 이 함수는 admissible을 만족하게 되며 그리고 다른 휴리스틱 함수보다 상당히 dominant하게 된다. 그림 1에 나타낸 간단한 2차원에서의 예를 살펴보자.

이 그림에서 S에서 G로 가는 경우 직선거리는 4이며, 블록영역(그림의 검은 부분)거리는 2(1.5칸이지만 셀 단위로 측정하므로 2)가 된다. 그리고 2차원의 경우는 (두 지점간의 거리) + 2*Max(두 지점간의 직선거리 사이에 포함된 블록영역 거리)를 사용하므로 함수값은 6이 된다.

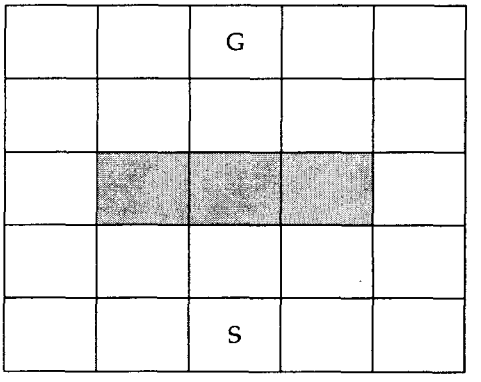

그림 1. 휴리스틱 함수 h' 예

2.7 경로표의 크기

가로 크기가 L, 세로의 크기가 M이고 높이의 크기가 N인 3차원 지도에서 각 지점을 표시하는 인덱스는 $\log_2^{L*M*N}$의 bit를 사용하여 표현할 수 있게 된다. 그러므로 각 경로표는 $\log_2^{L*M*N}$ L*M*N의 bit에 해당하는 크기를 가지게 된다. 또 지도의 모든 지점은 경로표를 갖게 되므로 경로표는 $(L*M*N)^2*\log_2^{L*M*N}$에 근접한 크기를 갖게 된다.

3. 경로표의 압축처리

지도상의 경로표들은 상당 부분 중복될 수 있다. 따라서 각각의 경로표에 현위치 기준으로 인덱스를 붙이고 중복된 경로표를 생략한 후 지도상의 각 지점들은 경로표의 인덱스를 보유하게 함으로서 경로표 개수를 줄여 데이터압축효과를 기할 수 있다.

3.1 압축효과의 실험

얼마만큼의 압축 효과가 발생하는 지를 알아보기 위해 여러 가지 데이터를 사용해서 실험해 보았다. 압축효과를 실험하기 위해서는 주어진 비율만큼의 블록영역을 갖는 임의의 게임지도를 생성해야 한다. 그런데 실제 게임에서는 블록된 영역이 랜덤하지 않고 일반적으로 연속적으로 나타나게 된다. 그림 2), 그림 3에는 YS-Eternal 게임[13]에서 나타나는 지도를 보여주고 있는데 블록된 영역이 연속적으로 나타나 있음을 알 수 있다. 압축효과실험을 위하여 주어진 비율만큼의 연속적인 블록영역을 포함하는 게임지도를 생성하는 프로그램을 작성하였다. 이 프로그램을 이용하여 지도크기와 지도상에서 블록된 영역의 비율을 변경해 가며 데이터 압축후 몇 개의 경로표가 생성되는 지를 검사하였다.

표 3은 게임지도 크기 4가지 경우에 대하여 블록영역이 각각 5%, 10%, 20%, 30%인 경우에 대한 실험결과를 나타낸다. 이 표의 세로축은 지도의 가로, 세로, 높이의 크기를 나타내며 가로축은 지도상에 포함되어 있는 블록영역의 비율을 나타낸다. 이 표에

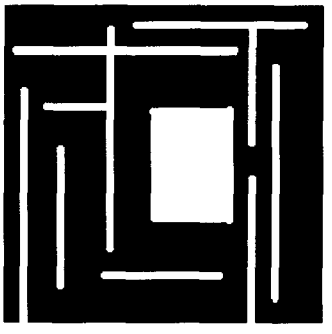

그림 2. 비디오 게임 지도1 (흰색이 블록영역)

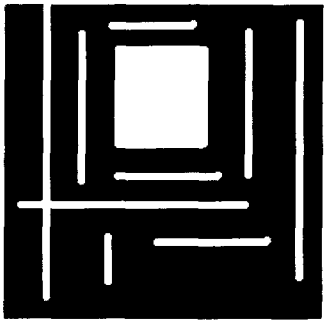

그림 3. 비디오 게임 지도2 (흰색이 블록영역)

표 3. 블록된 영역이 연속적인 경우 경로표 갯수

단위:픽셀	5%	10%	20%	30%
10*10*10	6.5 (.993)	13.0 (.987)	31.0 (.969)	40.5 (.951)
100*100*100	107.5 (1.0)	232.5 (1.0)	301.5 (1.0)	341.5 (1.0)
500*500*500	235.5 (1.0)	312.5 (1.0)	489.5 (1.0)	588.0 (1.0)
1000*1000*1000	228.0 (1.0)	357.0 (1.0)	553.5 (1.0)	1166.5 (1.0)

제시된 수치는 해당 지도상에서 생성되는 경로표의 개수를 나타내는데 이는 실험을 30회 반복한 결과의 평균치이다. 이 표에서 괄호안의 수치는 압축율(1.0-경로표수/맵크기)을 의미한다. 위 실험 결과 데이터 압축효과는 블록영역 비율이 적을수록 크게 나타남을 알 수 있으며 압축율은 거의 100%에 육박하고 있다. 이것은 일반적으로 게임분야에 사용되는 지도의 경우에는 훨신 적은 메모리 공간을 사용하여 경로표를 저장할 수 있음을 나타낸다. 만약 경로표를 사용하지 않고 최적경로를 그대로 저장하는 경우에는 현위치와 목표위치가 모두 다르기 때문에 모든 경로가 다르게 된다. 즉 이와 같은 압축방법을 사용할 수 없게 된다.

4. 경로표의 성능 측정

4.1 성능비교 방법

경로표를 사용했을 경우의 속도 향상을 A* 알고리즘[5]을 사용하는 경우와 비교하기 위하여 PC에서 C언어를 이용하여 실험하였다. 앞서 3장에서 사용한 프로그램을 이용하여 블록된 영역이 연속적으로 30%가 나타나도록 10*10*10, 100*100*100, 500*500*500, 1000*1000*1000 픽셀크기의 게임지도를 생성하였다. 일반 모니터의 해상도를 1024*768 픽셀로 가정하고 게임에서 사용되는 게임지도가 모니터의 가로, 세로 10배라고 가정하면 10240*7680 픽셀이 된다. 이렇게 큰 맵을 사용하면 오버헤드가 더 많기 때문에 일반적으로 게임에서는 8*8 픽셀을 하나의 타일(tile)로 처리한다. 그래서 10240*7680 픽셀 좌표계가 128*96 타일 좌표계가 된다. 그리고 게임상의 개체는 이 8*8 정사각형 타일에 위치하게 된다. 본 연구에서 실험대상으로 설정한 1000*1000*1000맵을 타일로 간주하면 픽셀로는 8000*8000*8000이 되어 웬만한 상용게임보다 큰 공간이 된다.

그리고 각각의 게임지도에서 10, 20, 30, 40개의 에

이전트가 동시에 임의의 출발점에서 임의의 목적지에 도달하는 시간을 측정하였다. 게임지도부터 다시 생성하는 이 실험을 각각 40회 반복하였으며 이 결과는 표 4부터 표 7에 각각 나타냈다. 이 실험의 결과 경로표를 이용하는 경우 A*를 이용하는 경우보다 1%의 유의수준에서 다 효율적이다고 말 할 수 있다. 이 실험에서 에이전트 여러 개를 이동하는 경우 서로 충돌할 수 있는데 A*의 경우는 충돌여부를 무시하고 이동시켰으며, 경로표 방법에서는 이동위치에서 충돌이 발생하는 지 미리 구간별로 검토하여 충돌이 발생하는 경우는 에이전트의 인덱스 순으로 이동을 시켰다.

4.2 성능비교결과

성능비교결과를 각 지도 크기별로 표 4, 표 5, 표 6, 표 7에 각각 나타냈다. 이 표에 나타낸 수치는 A*를 이용한 경우와 경로표를 이용한 경우에 이동에 소요된 시간의 산술평균을 나타낸다. 맵이 작은 경우는 큰 차이가 없지만 맵이 커질수록 많은 차이를 보이고 있다. 이 소요시간은 실시간 시스템의 구현에서 이는 매우 중요한 요소이다. 이를 통해 경로표를 이용한 시스템이 최적 경로를 이동하는 데 매우 우수한 수행 시간을 갖고 있고, 특히 실시간 시스템을 구현하는 데 있어 장점을 갖고 있다는 것을 알 수 있다.

5. 결론 및 향후 연구 방향

게임에서 에이전트의 이동은 중요한 기술요소중 하나이다. 에이전트의 이동경로는 최적성이 요구되며, 게임의 실시간 진행을 위하여 최단시간에 이동경로가 결정되어야 한다. 게임지도에 블록영역이 없는 경우에는 에이전트의 최적이동경로는 좌표에 의하여 결정될 수 있다. 그러나 블록영역이 있는 경우 에이전트의 이동경로를 찾기 위하여 탐색기법이 이용되어야 한다. 탐색기법중 최고의 성능을 보여주는 A* 알고리즘을 이용하는 것은 실시간게임진행에 부담을 주며, 다른 휴리스틱 탐색기법을 이용하는 경우 최적경로에 대한 보장이 없다. 그리하여 정적인 블록영역을 갖는 비디오 게임에서 경로표를 이용한 경로탐색 방법이 2D 게임에 대하여 제안되었으며 그 성능이 입증되었다. 이 논문에서는 2D 게임에서 활용한 이 경로표 활용방법을 Admissible한 휴리스틱 함수를 사용하여 3D 게임으로 확장하였으며 그 성능을 평가하였다.

최근의 게임에서는 대각선의 이동을 포함한 16방향, 32방향도 이용하는 추세이므로 이런 경우에 이용가능하도록 좀더 일반화된 경로표가 요구된다고 하겠다. 또한 실제 게임을 구현하면 정적 블록영역의 경우에도 에이전트가 존재하는 위치에 대해서는 동적인 블록영역이 되는데 이의 처리에 대해서도 정밀한 처리방법이 보완되어야 하겠다. 그리고 경로표의 데이터들을 좀 더 압축 시킬 수 있는 방향에 대한 연구가 필요하며 많은 실험을 통해서 지도의 크기와 경로표의 크기 사이의 함수 관계를 밝혀 내는 것도 병행되어야 하겠다.

표 4. 10*10*10 지도의 경우 단위:ms

	A*	경로표
10 에이전트	8.5	0.3
20 에이전트	15.2	0.3
30 에이전트	26.4	0.3
40 에이전트	36.9	0.4

표 5. 100*100*100 지도의 경우 단위:ms

	A*	경로표
10 에이전트	327.6	1.1
20 에이전트	623.5	1.2
30 에이전트	901.4	1.2
40 에이전트	1274.5	1.4

표 6. 500*500*500크기 지도의 경우 단위:ms

	A*	경로표
10 에이전트	4025.4	2.4
20 에이전트	7539.3	2.8
30 에이전트	10056.7	3.0
40 에이전트	12964.4	3.2

표 7. 1000*1000*1000 지도의 경우 단위:ms

	A*	경로표
10 에이전트	0.5964*106	4.2
20 에이전트	0.2389*107	5.1
30 에이전트	0.7896*108	5.8
40 에이전트	0.4357*1010	6.4

참 고 문 헌

[1] 주정규, "게임디자인(설계) 기법에 관한 연구", 전자공학회지, 제27권 제9호, 66~77쪽, 2000년 9월.

[2] 조성삼, 정문경, "온라인 게임 개발 현황", 한국정보처리학회지, 제9권, 제 3호, 24~33쪽, 2002년 5월.

[3] 최성, "게임 산업과 기술 전망", 한국정보처리학회지, 제9권, 제 3호, 11~23쪽, 2002년 5월.

[4] Marc Saltzman, Game Design : Secrets of the Sages, Brady Publishing, 1999.

[5] L. Bishop, "Designing a PC Game Engine", IEEE Transaction on Computer Graphics and Application, Vol 18, No 1, 1997.

[6] 신동일, 신동규, "다수 사용자 기반의 온라인 게임서버의 설계 및 제작", 전자공학회지, 제27권 제9호, 78~83쪽, 2000년 9월.

[7] N.J. Nilsson, Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, New York, 1971.

[8] Command and Conquer-Red Alert, WestWood사, 미국, 1996.

[9] Dune2, WestWood사, 미국, 1993.

[10] War Craft 2 - Tides of Darkness, Blizzard사, 미국, 1995.

[11] Stuart. Russel and Peter Norvig, Artificial Intelligence, Prentice-Hall, 1995.

[12] 심동희, 강혁, "게임지도에서 에이젼트 이동을 위한 경로표 활용", 한국정보처리학회 논문지 제7권 제10호, 2000년 10월.

[13] YS Eternal, Falcom사, 일본, 1987.

심 동 희

1980년 2월 서울대학교 산업공학과 (공학사)
1982년 2월 서울대학교 산업공학과 (공학석사)
1994년 2월 고려대학교 전산학과 (이학박사)
1982년~1985년 국토개발연구원
1985년~1990년 해양수산개발연구원
1990년 9월~현재 전주대학교 정보기술컴퓨터공학부 교수
E-mail: dhshim@jeonju.ac.kr
관심분야 : 게임공학, 네트워크 관리 및 보안, 기계학습

교 신 저 자

심 동 희 560-759 전북 전주시 완산구 효자동 1200 전주대학교 정보기술컴퓨터공학부