한국시뮬레이션학회 논문지
Vol. 19, No. 2, pp. 1-8 (2010. 6)

건물 내에서 화재시의 대피 시뮬레이션 설계 및 구현

장병옥[1†]

Design and Implementation of Evacuation Simulation of Indoor Environment Fire

Byeong Ok Jang

ABSTRACT

With recent development of computer hardware and 3D graphic technique, a lot of people have concern for something to express as the 3D graphic that look the real environment. Because the request of users have increased, the 3D simulation is developed and popularized in the many field. In this paper, we design and implement the simulation system that humans evacuate a building fires using the 3D graphic techniques. In this paper, we use the A* algorithm to humans have the artificial intelligence at evacuating a building fires, calculate the evacuation speed of each human considering temperature damage and smoke damage. In this paper, we applied the real building to demonstrate the effect of proposed evacuation simulation. Experimental results showed that the evacuation speed is affected by the temperature condition and the smoke density.

Key words : Fire evacuation simulation, Smoke, Temperature

요 약

최근 컴퓨터 하드웨어 및 3D 그래픽의 기술 발전으로 많은 사용자들이 실제와 유사한 3D 그래픽으로 표현되는 것에 관심을 가지고 있다. 이러한 사용자들의 요구가 증대됨에 따라 많은 분야에서 3D 시뮬레이션이 개발 보급되고 있다. 본 논문에서는 3D 그래픽 기술을 활용하여 화재시의 건물 내에서 사람들이 대피하는 시뮬레이션 시스템을 설계하고 구현한다. 본 논문에서는 사람이 화재 시 건물을 탈출할 때 인공지능을 갖기 위해 A* 알고리즘을 사용하였으며, 화재 시에 사람의 탈출에 영향을 미치는 열과 연기를 고려하여 각 사람의 탈출 속도를 계산한다. 본 논문에서 제안하는 대피 시뮬레이션의 효과를 입증하기 위해 실제 건물환경을 모델링하여 적용하였다. 실험결과들을 통해서 본 논문에서 제안한 방법을 통하여 연기의 농도가 짙어질수록 사람의 탈출속도가 감소하는 것과 온도와 연기 농도에 의해 사람이 피해를 입고 사망하는 것을 확인할 수 있다.

주요어 : 화재대피시뮬레이션, 연기, 온도, 시뮬레이션

1. 서 론

최근 다른 공학분야와 마찬가지로, 급속한 하드웨어 및 3D 그래픽 기술의 발달에 따라 3D 시뮬레이션에 대한 연구가 활발히 진행되었다. 3D 시뮬레이션의 활발한 연구와 더불어 안전에 대한 관심의 급증에 따라 화재 시 대피 시뮬레이션에 대한 연구도 활발히 진행되고 있으며 국내외에서 다수의 소프트웨어가 개발되고 있는 중이다. 대표적인 시뮬레이션 도구들로는 SIMULEX[1], EXODUS[2], EXITT[3], BuildingEXODUS[4]가 있는데, 이러한 시뮬레이션 도구들은 모두 2D를 기반으로 하고 있기 때문에 수치 및 데이터 등의 활용으로 전문가들이 대피 성능 및 효율을 판단하는 지표로는 활용될 수 있지만, 가시성이 떨어지기 때문에 비전문가인 일반사용자들이 분석하고 이해하기에는 적지 않은 어려움이 있다.

본 논문에서는 이러한 2D 소프트웨어들을 보완하여, 일반사용자들에게 가시적인 효과를 높이면서 시뮬레이션

*이 논문은 2009년도 나사렛대학교 학술연구비 지원에 의해 연구되었음
2009년 2월 2일 접수, 2010년 5월 31일 채택
[1] 나사렛대학교 인터넷정보학과
주 저 자 : 장병옥
교신저자 : 장병옥
E-mail; bojang@kornu.ac.kr

의 정확도를 유지하기 위해서 3차원 렌더링 기술을 활용하여 화재 발생 시에 사람들이 제한된 탈출구로 대피하는 시뮬레이션 시스템을 제안한다. 본 논문에서 제안하는 시뮬레이션 시스템은 제안된 탈출구를 통해 효율적이고, 안전하게 탈출할 수 있는 대피로를 탐색하고, 출구까지 탈출하는 동안에 연기 및 열에 의해 입는 피해 및 연기 및 열에 의한 생존 유무, 탈출하는 동안의 사람의 속도를 계산할 수 있다. 경로를 찾기 위한 알고리즘으로는 B* 알고리즘[5], Bellman-Ford 알고리즘[6], Best-First 탐색 알고리즘[7], Bidirectional 탐색 알고리즘[8], Breadth-First 탐색 알고리즘[9], Depth-First 탐색 알고리즘[10], Dijkstra의 알고리즘[11], Floyd-Warshall 알고리즘[12-14], Hill Climbing 알고리즘[15], Johnson의 알고리즘[16] 등이 있으며, 본 논문에서는 사람들의 대피로를 탐색하기 위해 Hart 등(1968)에 의해 처음 제안된 A* 알고리즘을 사용하였다.

실제 환경에서는 문에 도달하는 것이 탈출하는 것이 아니라 문을 빠져나가는 것도 문의 크기에 따라 다르기 때문에 본 시뮬레이션 시스템에서도 목표점인 출구에 도달하자마자 사람이 탈출한 것으로 간주하지 않고, 문의 폭, 사람의 폭에 따라 초당 빠져나갈 수 있는 사람의 수를 제한한다. 또한 실제 화재가 발생 했을 경우 사람에게 피해를 줄 수 있는 열과 연기를 적용한다. 온도는 사람에게 사망의 변수를 줄 수 있으며, 연기 농도는 사람의 사망 유무 및 이동 속도에 변수를 줄 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 시뮬레이션 시스템의 핵심 알고리즘들을 소개하고, 3장에서는 실험결과를 통하여 가시성을 확인 및 검증하며, 마지막으로 4장에서는 결론을 맺는다.

2. 본 론

본 논문에서 제안하는 시뮬레이션 시스템은 크게 6단계로 나누어 볼 수 있다. 이 시스템에서는 셀 단위로 구역을 구성하게 되는데 가장 먼저 이 셀을 편집한다. 셀을 편집하는 것은 제작한 편집 툴을 이용하여 마우스 클릭으로 셀을 편집할 수도 있고, 다른 시뮬레이션 툴과의 호환성을 위해 FDS파일 형식이 읽혀질 수 있도록 제작하였기 때문에 FDS 파일 포맷으로 건물을 편집할 수도 있다. 그 다음 단계로는 건물에 사람을 위치시킨다. 이 단계는 화재 발생시에 초기 사람의 위치로 활용되게 되며, 편집된 위치에서부터 화재 발생 시 탈출구로 대피하게 된다. 그 다음 단계는 각 사람마다 최근거리에 있는 탈출구로 최단거리를 이용하여 탈출할 수 있는 경로를 탐색하게 된다. 그 다음단계로는 연기와 열을 고려하여 탈출구를 향하여 대피하는 시뮬레이션을 시작하게 되고 모든 사람이 탈출하거나 사망했을 경우 시뮬레이션을 종료하게 된다. 마지막 단계로 시뮬레이션 결과를 파일에 저장하게 된다.

2.1 A* 알고리즘

사람의 대피 상황을 시뮬레이션 하기 위해서는 사람의 대피 경로를 계산하는 알고리즘이 필요하다. 본 논문에서는 사람의 대피 경로를 계산하기 위해서 인공지능 이론에서 자주 사용되는 A* 알고리즘을 사용하여 최적의 경로를 계산한다. A* 알고리즘의 특징은 두 지점 사이의 최적 경로를 빠른 시간 내에 찾을 수 있다는 것이다. A* 알고리즘은 최적의 탐색 방향을 평가하며, 뒤로 돌아와서 다른 경로를 찾기도 한다.

A* 알고리즘을 적용하기 위해서는 맵 상의 위치를 표현하는 자료구조인 노드(Node)를 정의해야 하며, 탐색되는 노드의 적합성을 평가하기 위해서는 거리(Distance)와 휴리스틱(Heuristic)을 정의해야 한다. 거리와 휴리스틱을 정의하기 위해서는 노드의 비용(Cost)을 목적에 맞게 추정할 수 있도록 해야 한다. A* 알고리즘의 목표는 비용이 가장 적게 드는 경로를 찾는 것이다. 비용의 정의는 응용에 따라 다르게 정의될 수 있기 때문에 각 응용에 맞게 적절한 비용을 정의하는 것이 매우 중요하다.

본 논문에서 A* 알고리즘을 선택하여 사용한 가장 중요한 이유로는 탐색으로 휴리스틱 알고리즘을 적용하여 사용하고 있으며, 최소의 계산으로 수행이 가능하기 때문이다. 과거 “A Formal Basis for the Heuristic Determination of Minimum Cost Paths in Graphs, 1968)논문에서 허용성과 최적성에 대한 증명이 이루어졌으며, Dijkstra's algorithm이나 Best-first Search(BFS)보다 처리속도가 훨씬 빠르기 때문이다. 따라서 최근 최단거리 최적화 알고리즘으로 가장 많이 사용되고 있는 A* 알고리즘을 선택하여 본 시뮬레이션을 구현하고 화재 발생시 대피 상황을 재현하여 실험을 진행하였다.

본 논문에서 노드의 속성은 각각 f, g, h로 정의한다. g(Goal)는 목표 노드를 의미하고 시작 노드에서 현재 노드까지 도달하는데 드는 비용이다. 시작 노드에서 현재 노드까지 도달하는 경로는 여러 가지가 있는데 각 경로에 해당하는 비용을 의미한다. h(Heuristic)는 휴리스틱을 의미하고 현재 노드에서 목표 노드까지 가는데 드는 추정된 비용을 의미한다. 추정 단계에서는 최단 경로를 모르고 있기 때문에 실제 비용은 알 수 없고, 최단 경로를 찾은

후에 실제 비용을 알 수 있다. f(Fitness)는 적합도를 의미하고 g+h의 값을 가진다. 현재 노드를 거쳐 진행하는 경로의 비용에 대한 최선의 추측이다. 각 노드는 속성 f, g, h를 가지도록 하고 노드들을 위한 두 개의 목록을 열린 목록(OL: open list)과 닫힌 목록(CL: closed list)로 관리하며 그림 1과 같이 계산된다.

2.2 Bellman-Ford Algorithm 알고리즘

Bellman-Ford algorithm은 가중 유향 그래프에서 최단 경로 문제를 푸는 알고리즘이다. 이때 간선의 가중치는 음수일 수도 있다. Dijkstra's algorithm은 Bellman-Ford algorithm과 동일한 작업을 수행하고 실행속도도 더 빠르다. 하지만 Dijkstra's algorithm은 가중치가 음수인 경우는 처리할 수 없으므로, 이런 경우에는 Bellman-Ford algorithm을 사용한다. V와 E가 각각 그래프에서 꼭지점과 모서리의 개수라고 한다면, Bellman-Ford algorithm의 실행시간은 O(VE)이다.

2.3 Floyd-Warshall 알고리즘

Floyd-Warshall Algorithm은 그래프에서 모든 정점간의 최단거리를 구하는 알고리즘이다. 음수 간선도 사이클만 없다면 잘 처리된다. 제일 바깥쪽 반복문은 거쳐가는 정점이고, 두 번째 반복문은 출발하는 정점, 세 번째 반복문은 도착하는 정점이다.

```
for(i=1;i<=N;i++) {
  for(j=1;j<=N;j++) {
    for(k=1;k<=N;k++) {
      if(cost[j][i]+cost[i][k]<cost[j][k]) {
        cost[j][k]=cost[j][i]+cost[i][k];
      }
    }
  }
}
```

```
Step 1.  n0 := 시작 지점 노드.
         - n0에 f, g, h 값들을 배정
         - OL := {n0}, CL := {}, n0.parent := null
Step 2.  n := OL에서 가장 첫 노드 (f가 가장 작은 노드), n을 OL에서 제거.
         - if (B == 목표 노드) then
             경로를 찾았음. 경로를 출력하고 알고리즘 종료함.
         - if (OL이 비어서 n이 없으면) then
             경로를 찾을 수 없음. 알고리즘을 끝냄.
Step 3.  n에 연결된 모든 유효한 노드(m)들에 대해서 반복.
         - new_g := n.g+cost(n,m)
         - if(m이 OL이나 CL에 있음 && m.g <= new_g) then
             m을 무시하고 다음 m에 대해서 진행
         - 경로를 설정(m.parent = n)
         - m.g := new_g
         - m.h := GoalDistEstimate(m)
         - m.f := m.g + m.h
         - if (m이 CL에 있음) then m을 CL에서 제거
         - if (m이 OL에 있음) then
             OL에서 m의 위치를 조정(f가 오름차순이 되도록)
           else m을 OL에 추가(f가 오름차순이 되도록)
Step 4.  n을 CL에 추가.
Step 5.  단계 2부터 다시 반복.
```

그림 1. A* 경로 찾기 알고리즘

2.4 사람의 탈출 속도 계산

사람의 탈출 속도에 영향을 미치는 요소는 두 가지가 있다. 첫 번째로 사람들의 밀집도에 따른 속도 감소이다. 사람들이 좁은 구역에 많이 몰려 있을수록 탈출 속도는 감소하게 된다. 두 번째는 연기 농도에 의한 속도 감소이다. 연기 농도가 짙어질수록 사람의 속도는 감소하게 된다. 본 논문에서는 건물의 구조적 환경에 따른 화재 발생 시의 탈출 속도에 대한 관계성을 확인해 보는 것을 주요 목적으로 주안점을 두었으므로, 탈출자의 개인적 환경 사항(사람들의 나이, 걷는 속도) 등은 고려되지 않았으며, 출입문의 크기는 일반 표준형 건물의 표준 출입문을 기준으로 하였다.

2.4.1 밀집도에 따른 속도 계산

사람의 밀집도에 따른 속도 계산은 식 (1)에 의해 계산된다. 사람의 탈출 속도에 영향을 미치는 요소는 두 가지가 있다. 첫 번째로 사람들의 밀집도에 따른 속도 감소이다.

$$S = k - ak\frac{1}{d^2\pi} \tag{1}$$

S는 밀집도를 고려한 사람의 탈출 속도를 의미하고, k는 SIMULEX에서 지정한 표 1의 일정 탈출 속도를 의미하며, d는 가장 가까운 사람과의 거리를 의미한다. 사람의 탈출 속도를 계산하는 과정의 전체 흐름도는 그림 2와 같이 나타난다.

첫 번째로 사람의 위치 정보를 입력으로 하여 자신의 앞에 위치한 사람인지 판단하여 불필요한 계산을 피하기 위해 그 사람이 진행하려는 앞 쪽에 있는 사람만 비교대상으로 삼는다. 사람의 진행 하려는 방향 벡터인 V_p와 다른 사람의 위치인 H_i에서 현재 사람의 위치인 H_c로의 벡터 V_d의 내적으로 식 (2)로 현재 사람의 앞에 위치한 사람인지 아닌지를 판별할 수 있다. 식 (2)의 내적 값이 양수이면 H_i가 H_c의 앞에 있는 것으로 판단한다.

$$V_p \cdot (H_i - H_c) \tag{2}$$

두 번째 단계와 세 번째 단계와 네 번째 단계에서는 현재 사람의 위치에서 앞의 위치한 모든 사람에 대해서 거리를 계산하여 현재 사람과 최소거리에 있는 사람과의 거리를 계산하는 단계이다.

다음으로는 최소속도와 최고속도의 제한을 두기 위한 단계로서 계산된 d가 0.7678보다 크다면 일정탈출속도와 0.85636을 곱한 값을 사람의 탈출 속도로 설정하고, d가 0.7678과 0.29098 사이의 값을 가진다면 d의 값과 일정 탈출 속도 k를 입력으로 식 (1)을 사용하여 사람의 탈출 속도를 계산한다. 이때의 d의 값은 2008년 한국 소방안전관리공단의 사전 연구 결과에 의한 화재시의 가장 가까운 사람과의 인접거리 범위를 의미한다.

표 1. 사람의 종별 일정탈출속도

구분	k
성인남성	1.581481438
성인여성	1.357187892
아이	1.054320959
노인	0.9341865571

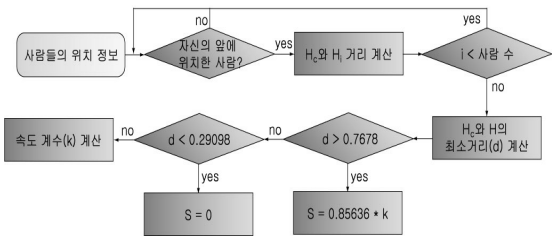

그림 2. 사람의 밀집도에 따른 속도 계산 흐름도

2.4.2 연기 농도에 따른 속도 계산

사람이 탈출할 때의 속도에 영향을 미치는 요소는 밀집도 외에 연기 농도도 있다. 연기의 농도가 짙어지면 사람의 시야가 흐려지기 때문에 사람의 탈출속도도 감소하게 된다. 연기 농도에 따른 사람의 속도는 식 (3)에 의해 계산된다.

$$S_{sm} = 1.1 - 8.8889 \times (7600 \times (smoke \times 10^{-6}) + (0.2)^2) \div 1.1 \tag{3}$$

식 (3)에서 S_{sm}은 연기 농도에 따른 사람의 속도이며, $smoke$는 연기 농도를 의미한다. 그림 3은 연기 농도에 따른 사람의 속도를 계산하는 전체 과정이다.

가장 먼저 식 (3)을 이용하여 연기 농도에 따른 속도 S_{sm}을 계산한다. 그 다음으로 계산된 S_{sm}이 0.3보다 작으면 사람의 속도는 0.3으로 설정하는데 밀집도에 따른 속도가 0.3보다 작다면 사람의 탈출속도는 밀집도에 따라 계산된 속도를 적용한다. 반대로 S_{sm}이 0.3보다 작지 않다면 밀집도에 따라 계산된 사람의 속도와 비교한다. 밀

집도에 따른 사람의 속도인 S보다 S_{sm}이 작으면 사람의 속도는 S_{sm}으로 설정되고 반대일 경우에는 사람의 속도는 S로 설정된다. 밀집도에 따른 사람의 속도와 연기 농도에 따른 사람의 속도는 매 프레임마다 계산하여 사람의 속도를 갱신하여 준다.

2.5 온도 및 연기 농도에 따른 피해도 계산

화재 시 사람에게 피해를 주는 요소로는 온도와 연기 농도가 있다. 온도가 높아지면 고온에 의해 화상을 입어 피해를 입게 되고, 연기 농도가 짙어지면 질식할 수가 있기 때문에 대피 시뮬레이션에서는 온도와 연기 농도에 따른 사람의 피해도를 계산하여 사망 유무를 판단할 수 있어야 한다. 온도에 의해 사람이 입는 피해식은 식 (4)와 같이 계산된다.

$$F_h = (thick/60)/e^{5.185-0.0273\times temp} + F_{h-1} \tag{4}$$

F_h는 온도에 따른 사람의 피해도를 의미하며 $thick$는 렌더링 시간 간격을, $temp$는 온도를, F_{h-1}은 이전의 온도에 의한 피해를 의미한다.

연기에 의해 사람이 입는 피해식은 식 (5)와 같이 계산된다.

$$\begin{aligned} F_{\in} = & ((((8.2925\times 10^{-4}\times mono^{1.036}))/30) \\ & +0.0045)\times e^{0.1903\times dio\times 100\times 2.0004)/71} \\ & +e^{8.13-0.54\times(20.721-oxy\times 100)}))) \\ & \times(thick/60)+F_{\in -1} \end{aligned} \tag{5}$$

$F_{\in}$은 연기에 의한 피해를 의미하며, $mono$는 일산화탄소의 농도, dio는 이산화탄소의 농도, oxy는 산소의 농도, $F_{\in -1}$은 이전의 연기에 의한 피해를 의미한다. 이러한 모든 연기의 요소를 고려하여 연기 농도에 따른 사람의 피해를 계산한다.

온도 및 연기 농도에 따른 사람의 피해량을 계산하는 전체 과정이 그림 4에 나타나 있다. 가장 먼저 연기 농도에 따른 피해를 식 (5)를 이용하여 계산한다. 다음으로는 온도에 따른 피해를 식 (4)를 이용하여 계산한다. 다음으로 연기농도에 따른 피해가 사망기준치(2008년 소방안전공학회의 사망기준치 연구결과)를 초과하게 되면 그 사람은 사망한 것으로 판단하고 그렇지 않으면 온도에 따른 피해를 사망기준치와 비교하여 사망기준치 보다 크면 사망한 것으로 판단한다. 사망 기준치는 에디터 프로그램에서 설정해 준다. 두 가지 모두에 해당하지 않는 다면 다음 대피 절차로 넘어가서 온도와 연기 농도에 따른 피해도를 다시 계산한다. 식 (4)와 식 (5)에서 보면 이전의 피해도를 계속 더해가며 사용하기 때문에 사람이 탈출하면서 입는 피해도는 누적된다는 것을 알 수 있다.

3. 시뮬레이션 결과

제안된 화재 시 대피 시뮬레이션 시스템의 가시성과 효율성을 입증하기 위해 본 논문에서는 윈도우 플랫폼에서 C++과 DirectX 9.0c를 사용하여 구현하였다. PC는 Athlon 64*2 Dual Core 프로세서로 2GB RAM의 메모리를 장착하였고, Geforce 7300 GPU의 그래픽 카드를 사용하였다.

그림 5는 사람이 탈출하기 위한 시뮬레이션의 초기 단계의 모습으로서 가시성을 평가하기 위해 사람이 사망하였을 경우 탈출 하는 중에 바닥에 눕는 것을 평가하기 위한 배치이다. 좀 더 가시성을 높이기 위해 사람은 소수의 인원만 배치해 보았다.

그림 6은 그림 5의 배치상태에서 시뮬레이션 하는 중간의 모습이다.

그림 7은 시뮬레이션이 종료되기 직전의 모습으로 두 명의 사람이 누워 있는 모습을 볼 수 있다. 그림 5는 나머지 사람은 모두 대피한 경우이고, 두 명의 사람은 피해도

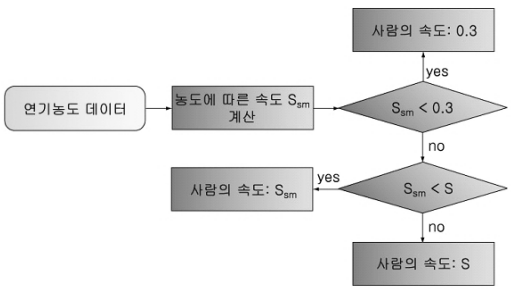

그림 3. 연기 농도에 따른 속도 계산 흐름도

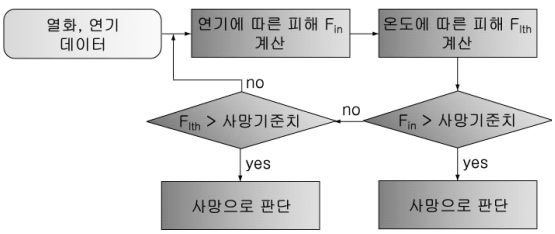

그림 4. 온도 및 연기 농도에 따른 피해도 계산 흐름도

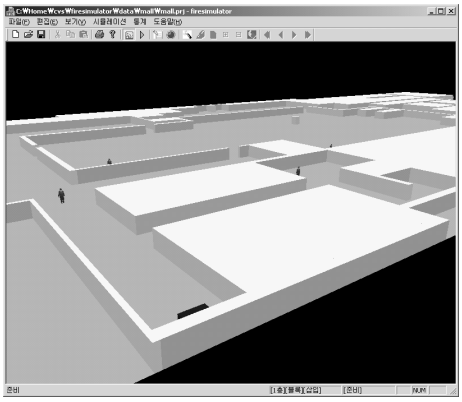

그림 5. 시뮬레이션 초기 단계의 모습

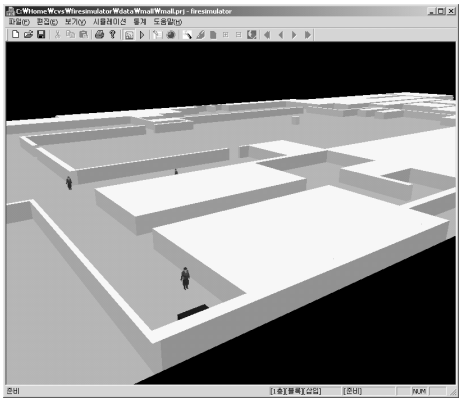

그림 6. 시뮬레이션 중간 단계의 모습

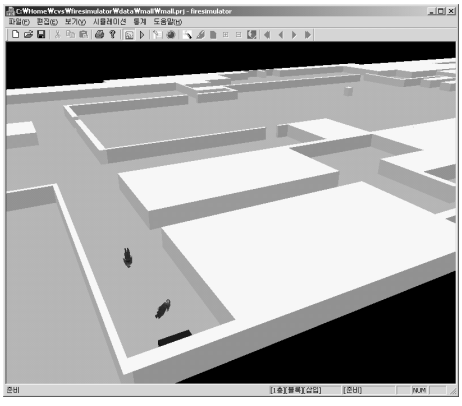

그림 7. 시뮬레이션 종료 직전의 모습

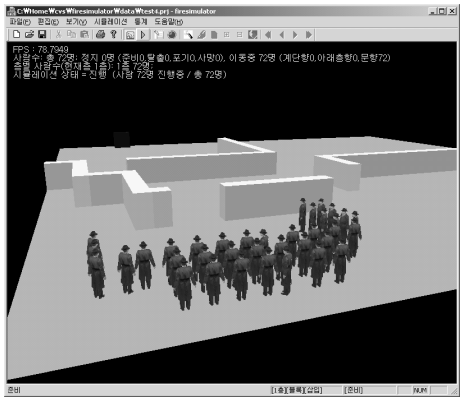

그림 8. 시뮬레이션 초기 모습

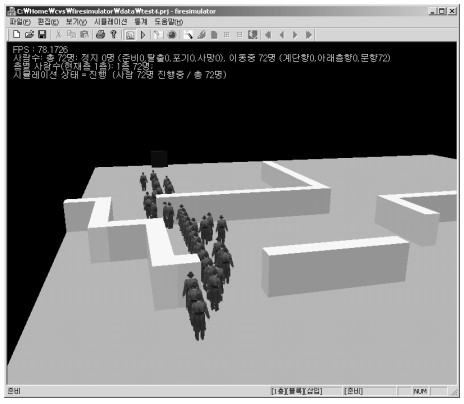

그림 9. 시뮬레이션 중간의 단계 모습

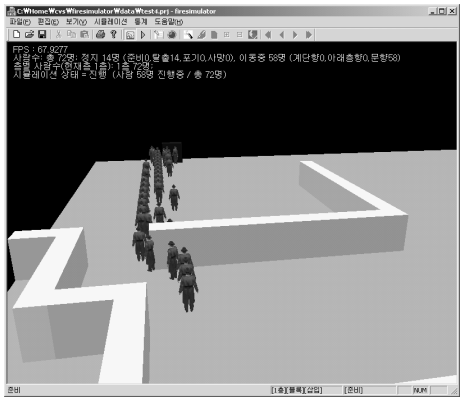

그림 10. 시뮬레이션 중간 단계의 모습

가 누적되어 사망했기 때문에 그림 7과 같은 모습을 보인다.

그림 8, 그림 9, 그림 10, 그림 11은 단층 건물에서 여러 사람이 하나의 출구를 향해 대피하는 것을 시뮬레이션하는 과정의 모습을 보여준다. 그림 8은 72명의 사람을 배치한 시뮬레이션 초기 단계의 모습을 나타낸다.

그림 9에서는 사람들이 좁은 통로를 통해 서로 같은 방향으로 진행하려 하기 때문에 진행한 거리가 서로 차이가 많이 나는 것을 확인할 수 있다.

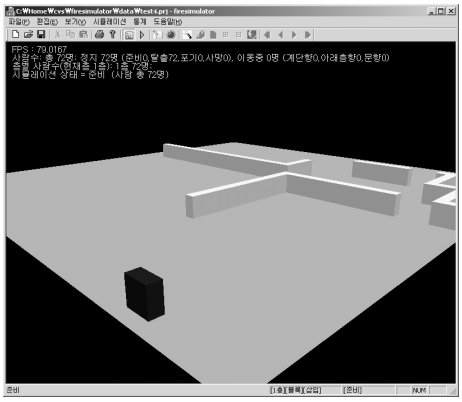

그림 11. 시뮬레이션 종료 모습

그림 10은 이제 막 사람들이 출구를 통해 탈출을 시작하는 모습을 나타낸다. 그림 11은 하나의 출구를 향해 72명의 사람이 모두 탈출하고 난 후의 모습이다.

본 논문에서 제안한 시뮬레이션 도구의 정확성을 높이기 위해서 사람이 출구까지 탈출하는 동안 탈출 속도에 영향을 미치는 사람들의 밀집도와 연기 농도, 온도의 요소들을 계산하여 사람이 탈출하는 동안, 대피 속도를 적용하였고, 탈출하는 동안 온도와 연기 농도에 의해 입는 피해도를 계산하여 사망기준치를 넘어서면 그 사람은 사망한 것으로 처리하였다. 이 사망 기준은 2008년 소방안전공학회의 사망기준치 연구결과를 적용하여 현실성 있는 시뮬레이션 도구를 개발하는데 주안점을 두었다.

4. 결 론

시뮬레이션의 효율성을 높이기 위하여 화재 발생 시 사람이 대피할 때 실제 환경과 같은 조건을 만족시켜 가시성을 높이고, 정확성을 높이기 위하여 온도와 연기를 고려한 대피 시뮬레이션 시스템을 제안하였다.

본 논문의 시뮬레이션 결과에서 확인할 수 있는 바와 같이, C++를 이용, 기본적 최단거리 알고리즘을 통하여 화재 발생시 대피 시뮬레이션을 3D그래픽으로 재현하고, 그 결과를 UI로 산출함으로써, 가상적 상황을 통해 화재가 발생되었을 때 안전사고 방지를 위한 목적으로 활용할 수 있도록 하였다. 본 연구는 한국 소방안전관리공단에서 3회에 걸쳐 유효성 및 적합성 테스트를 수행하였으며, 이후 수정 및 보완 연구를 통하여 소방안전을 고려한 건축물 설계 기준 방안 및 대피 운영 시나리오 작성에 적극적으로 활용할 계획이다.

본 논문에서 제안한 시뮬레이션 시스템을 이용하면 건물의 화재 발생 시 사망자 및 피해 정도를 예측할 수 있으며, 건물 설계 당시 출구를 어디에 두었을 때 가장 적은 피해를 입을 수 있는지 확인해 볼 수 있다는 장점이 있다. 특히 최근 자주 발생되고 있는 건물화재에 대비하여 안전대책 수립 및 건물의 최적화 설계를 위해 높은 활용 효과를 기대할 수 있다. 아직 연구 초기 단계에 있기 때문에 많은 프로그램들과 호환성이 적다. 앞으로 실제 건물의 설계를 할 때 사용되는 CAD와 같은 프로그램과 연동될 수 있도록 유연한 시뮬레이션 시스템을 만드는 것을 향후 연구과제로 하고 있다.

참 고 문 헌

1. Thompson P. A. and Marchant E. W., “Testing and Application of the Computer Model ‘SIMULEX’”, Fire Safety Journal, Vol. 24, No. 2, pp. 149-166, 1995.
2. E. R. Galea and J. M. P. Galparsoro, “EXODUS: An Evacuation Model for Mass Transport Vehicles”, Fire Safety Journal, Vol. 22, pp. 341-366, 1994.
3. Levin, B. N., “EXITT - A Simulation Model of Occupant Decisions and Actions in Residential Fires: Users Guide and Program Description”, US Department of Commerce, Gaithersburg, MD, 1987.
4. Gwynne S., Galea E. R., Lawrence P., J. and Filippidis L., “Modeling Occupant Integration with Fire Conditions Using the Building EXODUS Evacuation Model”, Fire Safety Journal, Vol. 36, No. 4, pp. 327-357, 2001.
5. B. Hans, “The B* Tree Search Algorithm. A Best-First Proof Procedure”, Journal of Artificial Intelligence, Vol. 12, No. 1, pp. 23-40, 1979.
6. R. Bellman, “On a Routing Problem, In Quarterly of Applied Mathematics”, Vol. 16, No. 1, pp. 87-90, 1958.
7. J. Pearl, “Heuristics: Intelligent Search Strategies for Computer Problem Solving”, Addison-Wesley, pp. 48, 1984.
8. D. Champeaux and L. Sint, “An Improved Bi-Directional Heuristic Search Algorithm”, Journal of ACM, Vol. 24, No. 2, pp. 177-191, 1977.
9. D. Champeaux, “Bi-Directional Heuristic Search Again”, Journal of ACM, Vol. 30, No. 1, pp. 22-32, 1983.
10. C. H. Thomas, L. E. Charles, R. L. Ronald and S. Clifford, “Introduction to Algorithm, Second Edition”, MIT Press and McGraw-Hill, pp. 540-549, 2001.
11. C. H. Thomas, L. E. Charles, R. L. Ronald and S. Clifford, “Introduction to Algorithm, Second Edition”, MIT Press

and McGraw-Hill, pp. 595-601, 2001.
12. R. W. Floyd, “Algorithm 97: Shortest Path”, Communications of the ACM, Vol. 5, No. 6, pp. 345, 1962.
13. S. C. Kleene, “Representation of Events in Verve Nets and Finite Automata, In C. E. Shannon and J. McCarthy: Automata Studies”, Princeton University Press, pp. 3-42, 1956.
14. S. Warshall, “A Theorem on Boolean Matrices”, Journal of ACM, Vol. 9, No. 1, pp. 11-12, 1962.
15. S. J. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach, Second Edition, Upper Saddle River, NJ: Prentice Hall”, pp. 111-114, 2003.
16. D. B. Johnson, “Efficient Algorithms for Shortest Paths in Sparse Networks”, Journal of ACM, Vol. 24, No. 1, pp. 1-13, 1977.
17. D. Rina and J. Pearl, “Generalized Best-First Search Strategies and The Optimality of A*”, Journal of the ACM, Vol. 32, No. 3, pp. 505-536, 1985.
18. P. E. Hart, N. J. Nilsson and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, IEEE Transactions of System Science and Cybermetics SSC4, No. 2, pp. 100-107, 1968.

장 병 옥 (bojang@kornu.ac.kr)

1990 서울산업대학교 전자계산학과 학사
1995 동국대학교 정보관리학과 석사
1999 경기대학교 전자계산학과 이학박사
2001～현재 나사렛대학교 교수

관심분야 : 모델링&시뮬레이션