웹기반 가상공간에서 실시간 네비게이션을 위한 셀 로딩 알고리즘

이 기 동[†] · 하 주 한[††]

요 약

현실감이 고려된 대부분의 가상공간을 구성하는 데에는 많은 양의 저장공간이 필요하게 된다. 이러한 가상공간에서의 네비게이션은 현실감 뿐만 아니라 실시간 반응성을 요구하게 되는데, 오프라인 시스템이라면 보조기억장치를 이용하여 한꺼번에 많은 데이타를 미리 저장해 두면 실시간 반응을 할 수 있게 된다. 그러나 웹기반의 온라인 가상 시스템에서는 데이터 전송시간이 실시간 반응성에 영향을 주기 때문에 새로운 방식의 로딩이 필요하게 된다. 본 연구에서는 웹기반의 가상공간 네비게이션을 위한 셀로딩 알고리즘 두 가지를 제안하고, 그 성능을 비교한다. 컴퓨터 시뮬레이션 결과, 필요없는 셀의 로딩을 줄임여 실시간 반응성을 향상시킬 수 있는 육각형 셀로딩 방식이 사각형 셀방식 보다 셀 로딩 개수를 평균 30% 정도 줄일 수 있음을 알 수 있다.

A Cell Loading Algorithm for Realtime Navigation in the Web-Based Virtual Space

Kidong Lee[†] · Juhan Ha[††]

ABSTRACT

Most of the virtual space constructed sufficiently realistic need a lot of memory space to navigate smoothly. And this kind of virtual space also requires real-time responsibility for the navigation as well as realism. In the off-line virtual system, real-time responsibility can be resolved by using large scale of secondary memory. In the web-based on-line virtual system, on the other hand, real-time responsibility is highly related to the latency time of network data communication. This induces the necessity of the algorithm for fast data loading. In this paper, we propose and verify the validity of the two methodology for cell loading algorithm. According to the results of computer simulation, the algorithm using hexagonal type cell promotes the real-time responsibility over 30% than that of the rectangular type.

키워드 : 셀로딩(Cell Loading), 가상공간(Virtual Space), 실시간 네비게이션(Real-Time Navigation)

1. 서 론

오늘날 많은 사람들은 평면적인 2차원 가상공간 사용에서 벗어나 3차원 가상공간 구성에 많은 관심을 보이고 있다. 또한 해마다 다양한 형태의 가상공간 네비게이션 뷰어가 소개되고 있지만 아직까지는 3차원 실세계를 2차원 평면상에 표현하기에는 화면의 복잡도, 정보의 왜곡, 생동감의 결핍 등의 한계가 뒤따른다[1-5]. 또한 각 객체의 디자인 한계성을 극복해야 하며 특히 실시간 네비게이션에서는 지속적으로 요구되는 데이터의 갱신이 필수조건이다. 웹에서 실제감이 있는 가상공간을 구현하고자 하는 욕망은 인터넷을 사용해 본 사람이라면 누구든지 공통된 바람일 것이다. 특히 요즘 각광 받고 있는 각종 지리정보시스템(GIS) 서비스에 있어서도 마찬가지이다.

일반적으로 가상공간에서의 실제감이라는 것은 지각(perception), 몰입(immersion), 상호작용(interaction) 등과 같은 여러가지 요소들과 관계가 있는데, 몰입 방식이나 상호작용성을 고려한 가상공간 구성 및 그 운용에 관한 연구도 많이 진행되어 왔으나[6], 위의 세 요소 중에서도 오감을 포함하는 지각이 가장 중요한 요소이다. 지각 중에서도 시각이 가장 큰 비중을 차지한다고 할 수 있다[7]. 그렇지만 시각적인 상세도를 높이는 것만으로는 현실감을 증대 시킬 수 없다는 것이 알려져 있다[8-11]. 가상공간의 현실감은 시각적인 상세도 뿐만 아니라 시야, 화면갱신률(frame update rate) 등이 관계가 있다[12, 13]. 물론 모든 요소들을 충분히 정확하게, 그리고 넓은 시야를 확보하고 화면 갱신률도 높게 설계하면 현실감이 증가하겠지만, 사용 환경이 충분하지 않을 경우에는 이들 요소들을 적절하게 조정을 해야 한다.

※ 이 논문은 1999학년도 영남대학교 학술연구조성비 지원에 의한 것임.
† 정 회 원 : 영남대학교 전자정보공학부 교수
†† 정 회 원 : (주)이노지오 대표
논문접수 : 2003년 12월 2일, 심사완료 : 2004년 4월 8일

시각적인 상세도는 데이터량과 직접 관계가 있기 때문에 무제한 상세하게 제작할 수는 없다. 그리고 인간의 시야는 수평으로는 200°(좌우 각 100°), 수직으로는 130°(상 : 60°, 하 : 70°)이나 사물의 외관을 정확히 파악할 수 있는 주시야는 주시점(진행 방향)에서 좌우로 2°정도이다[14]. 따라서 시야는 진행방향을 기준으로 2°정도만 정확히 확보되면 된다. 화면갱신율은 실시간 네비게이션에서 얼마나 빨리 맵을 로딩할 수 있는가에 관련이 있다. 여기서 화면갱신률이 바로 가상공간 네비게이션시 실시간 반응성과 관련이 있다. 이것은 셀로딩 알고리즘의 성능에 좌우된다.

다른 요소들에 대한 연구는 많이 진행되어 왔으나, 실시간 반응성에 관련된 연구는 아직 미흡하다. 가상현실시스템이 이러한 실시간 반응성을 가지기 위해서는 제한된 시간내에 처리할 수 있는 데이터량이 문제가 된다. 오프라인 프로그램일 경우에는 시스템 메모리 용량이 충분히 크다면 미리 데이터를 적재할 수 있으므로 문제가 없으나, 웹기반 가상공간시스템과 같은 온라인 프로그램에서는 데이터 전송 시간이 문제가 된다.

이러한 특징을 가진 가상공간에서 네비게이션을 할 때 셀 로딩시 많은 데이터양으로 인해 시스템에 부하가 많이 일어난다. 따라서 시스템 속도가 느린 낡은 사양 컴퓨터에서는 제대로 동작을 못하며, 진행 방향에서의 시각적인 왜곡성을 고려하지 못하고 있으나, 이에 대한 연구는 거의 없는 실정이다. 따라서 가상공간에서 현실감을 증가시킬 수 있는 실시간 네비게이션이 가능하기 위해서는 먼저 로딩해야 할 데이터량을 고려하여 가상공간에서 아바타(avatar) 시점에서 확보해야할 공간의 크기 및 방향을 결정하는 네비게이션 알고리즘 개발이 중요하다.

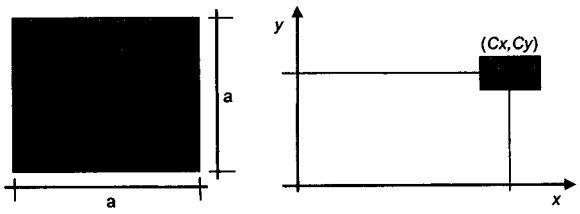
본 논문에서는 네트워크상에서 지원되는 가상공간을 네비게이션하는데 있어 현실감을 증가시킬 수 있는 셀 로딩 알고리즘 기법을 제안하고자 한다. 먼저 두 가지의 셀 로딩 알고리즘 기법을 제안하고, 이를 이용한 실시간 네비게이션 알고리즘을 설명한다. 컴퓨터 시뮬레이션을 통하여 제안된 알고리즘의 성능을 비교·분석한다.

2. 셀로딩 알고리즘

본 연구에서는 두 가지의 셀로딩 알고리즘에 제안한다. 첫 번째 알고리즘은 로딩할 셀의 형태가 사각형인 방식이며, 두 번째 방식은 첫 번째 방식보다 복잡하지만, 실시간 네비게이션에서의 반응성이 향상될 수 있는 알고리즘이다.

2.1 사각형 셀 로딩 알고리즘

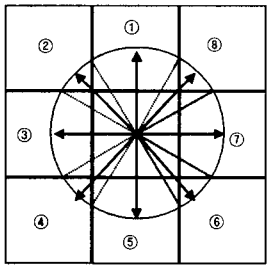
이 방법은 시스템이 로딩할 셀의 기본 형태가 사각형(여기에서는 정사각형으로 가정한다)으로 가정한 것이다. 전체 맵을 사각형 셀로 맵핑을 한 것으로 위치정보를 알기 위해서는 사각형 셀의 인덱스 정보를 가지고 있으면 된다. 셀의 인덱스 정보를 알려면 먼저 블록으로 잡은 셀의 위치, 즉 좌표 값을 알아야 한다. (그림 2-1)(a)에 나타나 있듯이, 한 변의 길이가 a인 정사각형 블록과 그에 해당하는 셀 영역 위치 좌표를 구하는 식은 식 (2-1)에 있다. 여기서 INT[x]란 x를 넘지 않는 최대 정수값을 나타낸다.



(그림 2-1) 셀 영역 이동 방향 및 위치좌표

$$C_x = INT[x/a]$$
$$C_y = INT[y/a] \qquad (2\text{-}1)$$

이러한 사각형 셀 알고리즘에서 네비게이션 방향에 따라 새로 로딩해야 할 셀이 달라지는데, 먼저 이동 방향에 따른 구분은 다음 (그림 2-2)와 같이 크게 8개의 방향으로 나뉜다. 그림에서 보듯이 ①,③,⑤,⑦ 방향으로 네비게이션 할 때는 면적의 연속성이 이어지지만, ②,④,⑥,⑧ 방향으로 이동할 때는 면적의 연속성이 깨진다. 따라서 실제 적용할 때에는 이부분에 대한 고려가 있어야 한다는 것이 이 알고리즘의 단점이지만, 셀 인덱스의 계산이 뒤에 소개하는 알고리즘 보다 간단하다는 장점이 있다.

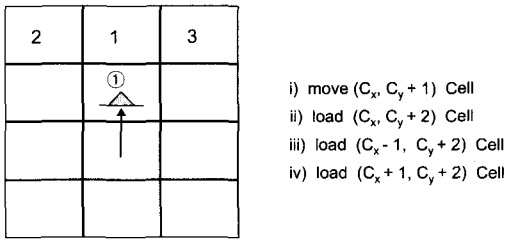


(그림 2-2) 셀 방향성

그러면 사각형 셀의 이동 방향에 따른 셀 로드 순서에 대해서 살펴본다. 위의 (그림 2-2)와 같이 ①번 방향에서 ⑧번 방향까지 각각 이동할 때, 현재 위치의 셀에서 로딩되어야 할 셀의 인덱스 및 그 순서에 대한 알고리즘이 아래에 설명되어 있다.

먼저 ①번 방향으로 네비게이션이 진행되면 (그림 2-3)처럼 로드되어진다. (그림 2-3)에서 보듯이 현재 위치에서 ①번 셀 영역으로 이동하면 먼저 전진성에 의해 전방의 1번 셀 영역이 먼저 로드되고 다음은 2번 셀을 로드하게 되고 마지막으로 3번 셀을 로드하게 된다. 3번 셀보다 2번 셀을 먼저 로딩하는 이유는 인간의 시각시스템은 일반적으로 왼쪽에서 오른쪽으로 흐른다는 것을 감안하였다. 이렇게 하면 새로 이동한 셀을 중심으로 사방으로 8개의 셀을 확보할

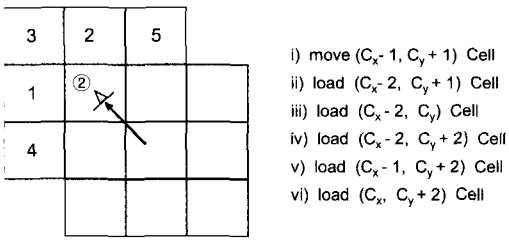
수 있게 되어 초기 상태와 같게 된다. 이렇게 로딩 셀의 우선 순위를 정해 두는 것은 실시간 네비게이션이 요구 될 때 최소한의 시야를 확보한 채 네비게이션을 진행해야하기 때문이다. 물론 새로운 셀이 3개 추가로 로딩되면 기존에 로딩 된 셀 중에서 아래 부분의 셀 3개가 자동으로 삭제(덮어 쓰기)된다. ③, ⑤, ⑦ 방향으로 네비게이션이 진행되는 경우는 앞에서 설명한 경우와 거의 유사하다. 단, 로딩해야 할 셀의 인덱스만 차이가 있다.



(그림 2-3) ①번 셀 영역 이동

이와 같이 네비게이션이 전·후, 좌·우 방향일 경우에는 셀 로딩방식이 거의 유사한 방식으로 되어 있다. 진행하는 방향의 전방에 있는 셀을 먼저 로드하고, 진행 방향을 기준으로 좌측에서 우측순으로 셀이 로드 된다.

그러나, 대각선 방향의 셀 로딩방식은 이와는 조금 다르다. (그림 2-4)에서는 보는 바와 같이 ②번 셀 영역으로 이동하면 먼저 사용자 위치의 회전방향에 따라 로드되는 우선순위가 달라진다. 여기에서, 5개의 셀 영역이 로드되고, 로드되는 순서는 (그림 2-3)의 경우와는 달리 전방에 있는 셀 영역을 우선적으로 로드하지 않는다. 인간의 시점이 왼쪽에서 오른쪽으로 흐르는 것을 감안하여 진행 방향을 기준으로 좌측 전방에 있는 셀 영역을 우선적으로 로드한 뒤, 우측에 있는 셀 영역을 로드하게 된다. 그 다음 진행 방향의 전방에 있는 셀을 로딩하고, 마지막으로 나머지 4, 5번 셀을 로딩한다. 이렇게 하는 이유는 진행방향의 전방에 있는 셀이 이전 셀과 공간의 연속성이 없기 때문이다. 따라서 공간의 연속성을 확보하기 위하여 좌우의 시야를 확보한 후, 전방의 시야를 확보해야 한다. 물론 새로운 셀이 5개 추가로 로딩되면 기존에 로딩된 셀 중에서 오른쪽 아래 부분의 셀 5개가 자동으로 삭제(덮어 쓰기가 된다)된다.



(그림 2-4) ②번 셀 영역 이동

물론 ④, ⑥, ⑧ 방향으로 네비게이션이 진행되는 경우는 앞에서 설명한 (그림 2-4)의 경우와 거의 유사하며 로딩해야 할 셀의 인덱스만 고려해 주면 된다. 이와 같이 네비게이션이 대각선 방향으로 이동할 때에는 전, 후, 좌, 우로 이동 할 때와는 달리 로딩해야 할 셀의 개수가 2개 많아지게 된다.

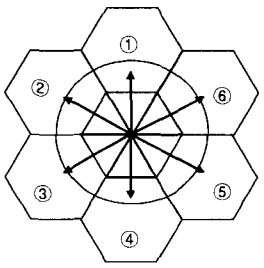
이상에서 볼 때 사각형 셀 알고리즘에서는 네비게이션이 진행 방향을 기준으로 전, 후, 좌, 우일 경우에는 실시간 진행에 크게 무리가 없이 진행 방향의 시야를 확보 할 수 있으나, 진행 방향을 기준으로 대각선 쪽으로 이동할 경우에는 실시간적으로 시야를 확보하는데 무리가 있음을 알 수 있다.

2.2 육각형 셀 로딩 알고리즘

2.1절에서 다룬 사각형 셀 알고리즘의 단점이 생기는 이유는 사각형의 회전 대칭성이 2축 밖에 없기 때문이다. 따라서 사각형 셀 알고리즘의 단점을 보완하기 위해서는 회전 대칭성이 좋은 구조의 셀을 사용하는 것이 좋다. 그러나 전체 맵을 중복 없이 표현 할 수 있는 구조는 정 육각형 밖에 없다. 따라서 여기에서는 정육각형 형태를 가진 셀의 로딩 알고리즘을 제안하고자 한다. 먼저 정육각형 셀 로딩 알고리즘을 설명하고 이어서 로딩해야 할 셀의 인덱스 정보를 구하는 방법을 설명한다.

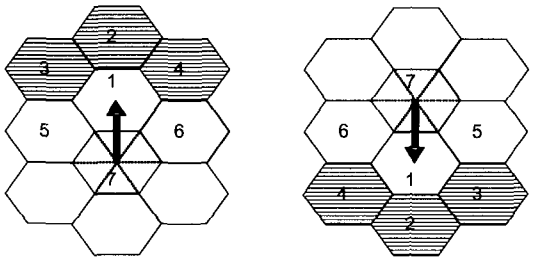
2.2.1 셀로딩 알고리즘

(그림 2-5)는 현재위치에서 메모리에 올라와 있는 정육각형 셀의 전체 형태이다. 전체 7개의 정육각형 셀이 사용되며, 영역의 중심에서 네비게이션 이동 방향성을 크게 6가지로 나눌 수 있다.



(그림 2-5) 정육각형 셀 영역 방향성

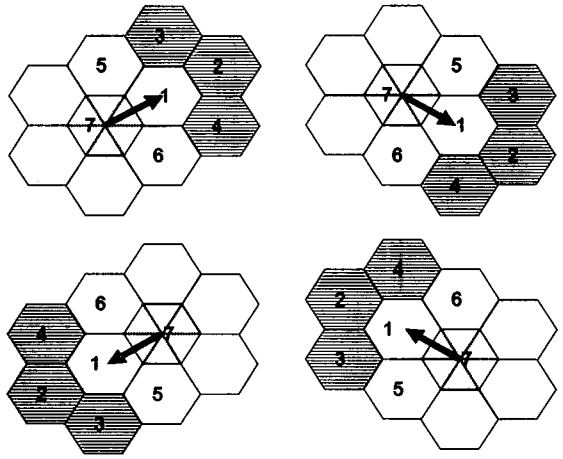
다음은 셀 로딩순서에 대하여 살펴보자. (그림 2-6)에서



(그림 2-6) 셀 영역의 로드순위(I)

보는 바와 같이 화살표 방향으로 이동하면, 진행하는 방향의 전방에 있는 2번 셀이 먼저 로드되고 다음은 시각적인 방향이 왼쪽에서 오른쪽으로 흐르는 것을 감안하여 3번을 로드한 후, 4번 셀을 로드하게 된다. 물론 기존의 셀에서 3개의 셀이 메모리에서 삭제된다. 이 경우는 (그림 2-3)에서 보는 것과 같이 사각형 셀의 경우와 유사함을 알 수 있다.

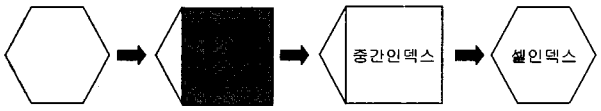
또한, 대각선 방향으로 이동하면 (그림 2-7)에서와 같이 로드한다. 이 경우에도 3개의 셀이 새로 로드되는 것을 볼 수 있다. 그러나 앞에서 설명한 사각형 셀의 경우에는 5개가 로드된다. 그리고 사각형 셀의 경우에는 공간적인 연속성을 위하여 진행하는 방향의 셀을 우선적으로 로딩하지 못하는 것에 비하여 육각형 셀의 경우에는 항상 전방의 셀을 확보할 수 있다는 장점이 있다.



(그림 2-7) 셀 영역의 로드순위(Ⅱ)

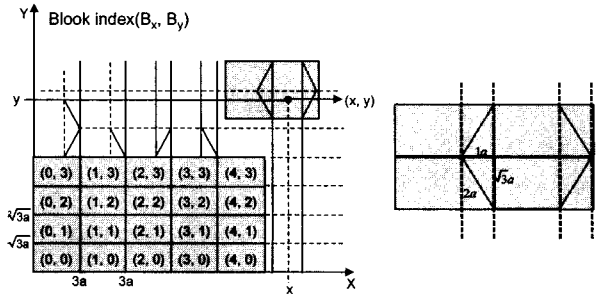
2.2.2 셀 인덱스 계산

다음은 현재의 위치 정보(x, y 좌표값)를 가지고 로딩해야 할 셀의 인덱스를 구하는 방법을 살펴보자. 육각형 셀의 경우에는 사각형 셀의 경우와 달리 정방형이 아니므로 셀 인덱스를 구하는 방법이 간단하지 않다. 제안하는 방법은 (그림 2-8)과 같이 3단계로 나누어 셀 인덱스를 구한다. 먼저 블록인덱스(block index)를 구하고, 그다음 중간인덱스(intermediate index)를 구한 후, 최종적으로 셀인덱스(cell index)를 계산한다.



(그림 2-8) 셀인덱스 구하는 순서

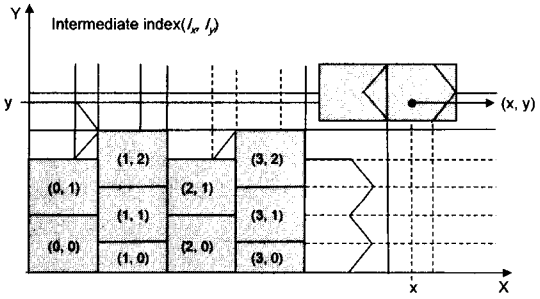
먼저 블록인덱스를 구하는데 (그림 2-9)에서 보는 바와 같이 전체 맵을 블록으로 나눈다. 정육각형의 한변의 길이가 2a이면 블록의 형태는 가로가 3a이고 세로가 $\sqrt{3a}$인 직사각형이 된다. 그러면 현재 위치가 (x, y)일 경우, 블록 인덱스는 다음 식 (2-2)에 의해서 구해진다. 예를 들면, 현재 위치 좌표값이 (10, 6)이고 정육각형의 한변의 길이를 1이라고 가정하면 현재 위치의 블록인덱스는 B(3, 3)이 된다.



(그림 2-9) 블록인덱스 셀 영역

$$B_x = INT[x/3a]$$
$$B_y = INT[y/\sqrt{3a}] \qquad (2\text{-}2)$$

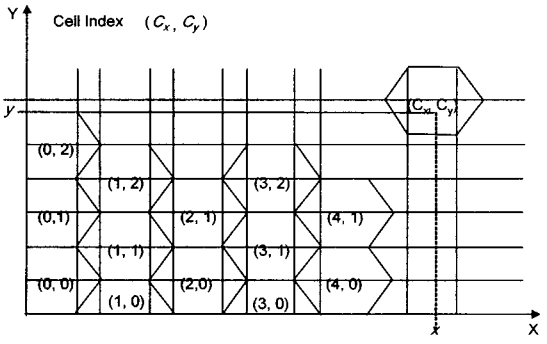
블록인덱스를 구하고 나면 두 번째로 중간인덱스 값을 구해야한다. 중간인덱스란 (그림 2-8)에 나타나 있듯이, 앞에서 구한 블록인덱스를 통합하여 육각형 셀의 기본 형태를 만드는 일이다. 구하는 방법과 식은 (그림 2-10), 식 (2-3)에 나타나 있다. 앞의 예에서 좌표값이 (10, 6)인 경우, 블록인덱스는 B(3, 3)인데, 중간인덱스는 I(3, 2)가 된다.



(그림 2-10) 중간 인덱스 셀 영역

$$I_x = B_x$$
$$I_y = \begin{cases} INT[B_x/2] & if\ I_x = even(include\ 0) \\ INT[(B_y + 1) & if\ I_x = odd \end{cases} \qquad (2\text{-}3)$$

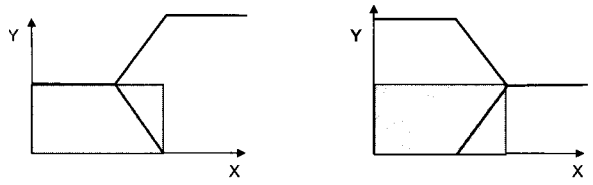
마지막으로 (그림 2-11)과 같은 셀인덱스 값을 구해야 한다.



(그림 2-11) 셀 인덱스 영역

먼저 해당 블록을 좌측 하단의 원점으로 식 (2-4)와 같이 평행 이동 하면 좌표의 위치에 따라 (그림 2-12)와 같은 두 가지의 패턴으로 나눌 수 있다.

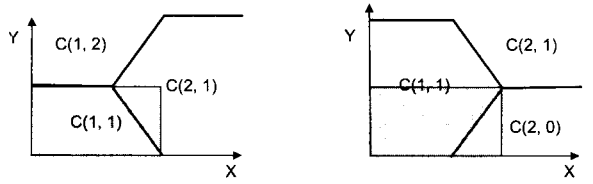
$$x_t = (x - 3aB_x)$$
$$y_t = (y - \sqrt{3}aB_y) \qquad (2\text{-}4)$$



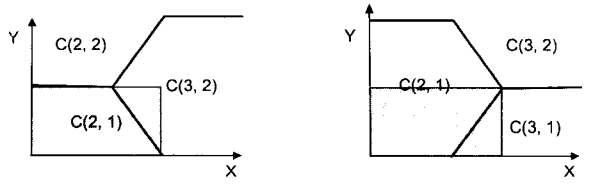
(a) 패턴a : B_y가 홀수 (b) 패턴b : B_y가 짝수

(그림 2-12) 기본 패턴

이 기본 패턴의 경계부분에서 셀인덱스가 달라지는 부분이 생긴다. 이 부분을 조정해 주는 것이 셀인덱스를 구하는 방법이다. 기준 셀의 블록인덱스에 따라서 다음 (그림 2-13)과 같은 셀인덱스를 가진다. 즉 I_x가 홀수이고 패턴a일 경우에는 셀인덱스가 (C_x, C_y) 또는 (C_x+1, C_y)중의 하나가 되고, I_x가 홀수이고 패턴b일 경우에는 셀인덱스가 (C_x, C_y) 또는 (C_x+1, C_y-1) 중의 하나가 된다((그림 2-13)(a) 참조). 그리고 I_x가 짝수이고 패턴a일 경우에는 셀인덱스가 (C_x, C_y) 또는 (C_x+1, C_y+1)중의 하나가 되고, I_x가 짝수이고 패턴b일 경우에는 셀인덱스가 (C_x, C_y) 또는 (C_x+1, C_y)중의 하나가 된다((그림 2-13)(b) 참조).



(a) I_x가 홀수 일 때



(b) I_x가 짝수 일 때

(그림 2-13) 셀인덱스 예

그러면 이 네 가지 경우에 적용되는 판별식은 다음 식 (2-5)와 같다.

CASE I : I_x가 홀수이고, B_y가 홀수인 경우

$$\begin{cases} C_x = I_x + 1, C_y = I_y & if(y_t - \sqrt{3x_t} - 3\sqrt{3a}) > 0 \\ C_x = I_x, C_y = I_y & else \end{cases}$$

CASE II : I_x가 홀수이고, B_y가 짝수인 경우

$$\begin{cases} C_x = I_x + 1, C_y = I_y - 1 & if(\sqrt{3x_t} - y_t - 2\sqrt{3a}) > 0 \\ C_x = I_x, C_y = I_y & else \end{cases}$$

CASE III : I_x가 짝수이고, B_y가 홀수인 경우

$$\begin{cases} C_x = I_x + 1, C_y = I_y + 1 & if(y_t - \sqrt{3x_t} - 3\sqrt{3a}) > 0 \\ C_x = I_x, C_y = I_y & else \end{cases}$$

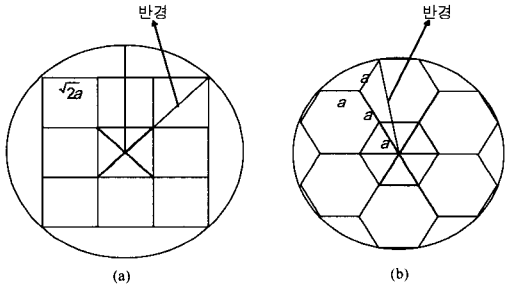
CASE IV : I_x가 짝수이고, B_y가 짝수인 경우

$$\begin{cases} C_x = I_x + 1, C_y = I_y & if(\sqrt{3x_t} - y_t - 2\sqrt{3a}) > 0 \\ C_x = I_x, C_y = I_y & else \end{cases} \qquad (2\text{-}5)$$

앞의 예에서 좌표값이 (10, 6)이면 블록인덱스는 B(3, 3)이고, 중간인덱스는 I(3, 2)이 되었다. 이 경우 좌표를 평행 이동하면 (1, 0.8)이 되고 이 좌표를 이용하여 셀 인덱스를 구해보면 중간인덱스와 같은 C(3, 2)가 된다. 물론 사각형 셀의 경우보다 계산량이 많지만 전체 계산식은 간단하여 실시간 계산에 문제가 될 정도는 아니다.

2.2.3 육각형 셀 방식의 장점

셀로딩 개수로 판단할 때에는 육각형 셀 방식이 더 좋다는 것을 직관적으로 알 수 있다. 그리고 모든 방향으로의 시야 확보성을 비교해 보면 다음과 같다. (그림 2-14)에 나타나 있듯이 사각형 셀의 한 변이 $\sqrt{2}a$라면 원의 반경은 $3a$가 되고, 셀의 전체 면적은 $18a^2$이 된다. 육각형의 한 변이 a라면, 원의 반경은 $\sqrt{7}a$가 되고, 셀의 전체면적은 $\frac{21}{2}\sqrt{3}a^2$이 된다. 따라서 원의 면적에서 셀영역이 차지하는 비율을 보면 사각형셀은 약 64%이고, 육각형셀은 83%가 되어 육각형 셀 방식이 사각형 셀 방식 보다 훨씬 더 충실하게 시야를 확보할 수 있다는 것을 알 수 있다. 또한 셀영역의 최외곽에서 원둘레에 이르는 최대거리비율이 30%이고, 육각형셀은 24%로써 방향성이 훨씬 좋다는 것을 알 수 있다.

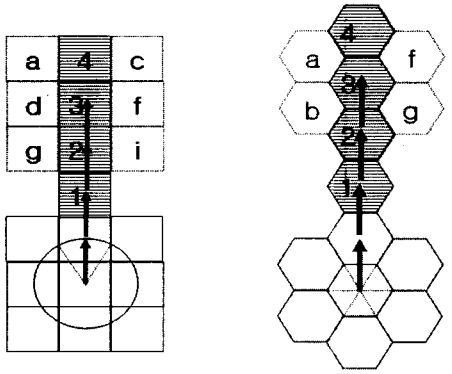


(그림 2-14) 셀 영역 면적량 비교

3. 실시간 네비게이션

앞에서 제안한 두 가지의 셀알고리즘을 실제 네비게이션에 적용한다면 이동하는 속도가 문제가 될 수 있다. 특히

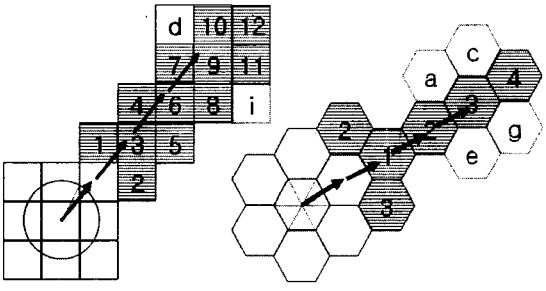
실시간 네비게이션이 요구되는 상황에서는 로딩하는데 걸리는 시간이 중요한 요소가 된다. (그림 3-1)과 (그림 3-2)에 실시간 네비게이션 알고리즘을 보였다. 먼저 (그림 3-1)에 나타나 있는 것과 같이 현재 위치에서 전방으로 진행하는 경우에는 두 알고리즘 모두 전방에 있는 셀 하나만 새로 로딩하면 된다. 그리고 만일 3의 위치에서 충분한 시간을 가지고 머물러 있다면 나머지 셀들도 차례대로 로딩하면 된다. 물론 로딩이 끝나기 전에 네비게이션이 진행되면 나머지 셀들은 로딩하지 않고 계속 진행한다. 사각형셀에서는 전, 후, 좌, 우의 네 방향으로 이동할 때에는 같은 방법이 적용되며, 육각형셀에서는 전, 후 두 방향으로 이동할 때에 같은 방법이 적용된다.



(a) 사각형 셀 (b) 육각형 셀

(그림 3-1) 전방으로 연속적 셀 영역 이동

그런데 (그림 3-2)와 같이 네비게이션이 기준 위치에서 대각선 방향으로 진행하면, 사각형셀은 진행 방향의 시야를 확보하기 위하여 최소한 3개의 셀을 새로 로딩해야 한다. 그러나 육각형 셀의 경우에는 1개의 셀만 새로 로딩하면 된다.

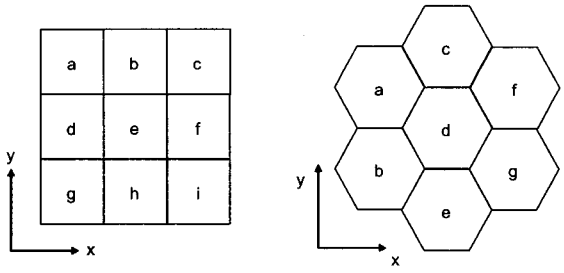


(a) 사각형 셀 (b) 육각형 셀

(그림 3-2) 대각선 방향으로 연속적 셀 영역 이동

4. 시뮬레이션 및 결과

셀 로딩 시뮬레이션을 통해 두 방법의 장단점을 살펴보고자 한다. 먼저 (그림 4-1)과 같이 네비게이션 셀 로딩의 셀 영역 블록의 이름을 부여한다. 그리고 네비게이션 방향에 따른 셀 로딩 순서가 <표 4-2>와 <표 4-3>에 있다. 표에서 밑줄 친 부분은 실시간 네비게이션시 최소한의 시야를 확보하기 위하여 로딩해야 하는 셀이다.



(그림 4-1) 셀 영역 블록의 이름

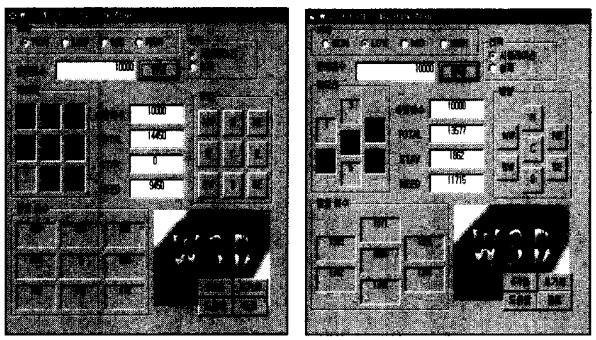
〈표 4-1〉 정사각형 셀 로드 순서

이 동	로드 순서	배열명
↑	e→ b → a → c → d → f → g → I → h	NN[9]
↗	e → b → f → c → a → I → d → h → g	NE[9]
→	e → f → c → I → b → h → a → g → d	EE[9]
↘	e → f → h → I → c → g → b → d → a	SE[9]
↓	e → h → I → g → f → d → c → a → b	SS[9]
↙	e → h → d → g → I → a → f → b → c	SW[9]
←	e → d → g → a → h → b → I → e → f	WW[9]
↖	e → d → b → a → g → c → h → f → I	NW[9]

〈표 4-2〉 정육각형 셀 로드 순서

이 동	로드 순서	배열명
↑	d → c → a → f → b → g → e	NN[7]
↗	d → f → c → g → a → e → b	NE[7]
↘	d → g → f → e → c → b → a	SE[7]
↓	d → e → g → b → f → a → c	SS[7]
↙	d → b → e → a → g → c → f	SW[7]
↖	d → a → b → c → e → f → g	NW[7]

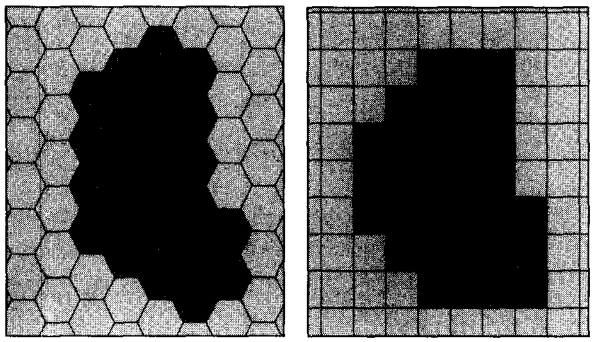
제안된 모델에 대한 네비게이션 셀 로딩 시뮬레이션을 위하여 다음 (그림 4-2)와 같은 프로그램을 작성하여 그 성능을 비교 분석하였다.



(a) 사각형셀 (b) 육각형셀

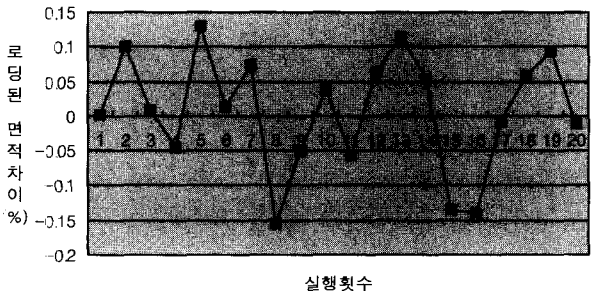
(그림 4-2) 시뮬레이션 프로그램

먼저 실시간 네비게이션이 아닌 일반 네비게이션(모든 셀을 전부 로딩한 후 진행하는 경우) 경우를 살펴보면 다음 (그림 4-3)과 같은 결과를 볼 수 있다. 즉 똑같은 경로로 네비게이션 했을 경우의 로딩셀 영역을 볼 수 있다. 여기서 네비게이션의 방향은 랜덤으로 선택한다. 여러 가지 경로에 대하여 시뮬레이션 한 결과, (그림 4-4)에서 보듯이 전체 로딩셀 영역의 크기는 사각형셀 방식이나 육각형 셀 방식이 거의 같음(오차가 평균 0.007%)을 알 수 있고, (그림 4-5)에는 셀로딩 회수에 있어서는 육각형 셀 방식이 사각형셀 방식에 비하여 약 23% 정도 적어짐을 알 수 있다. 즉, 같은 면적의 시야를 확보한다고 가정하면 사각형셀 한 개의 크기가 육각형셀 한 개 크기보다 약 23% 정도 작기 때문이다. 따라서 육각형 셀 방식은 셀 로딩에 따른 부하를 줄일 수 있는 장점이 있다.

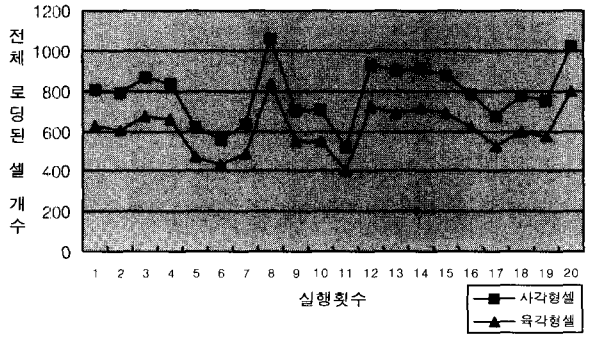


(a) 사각형셀 (b) 육각형셀

(그림 4-3) 네비게이션 실행 화면

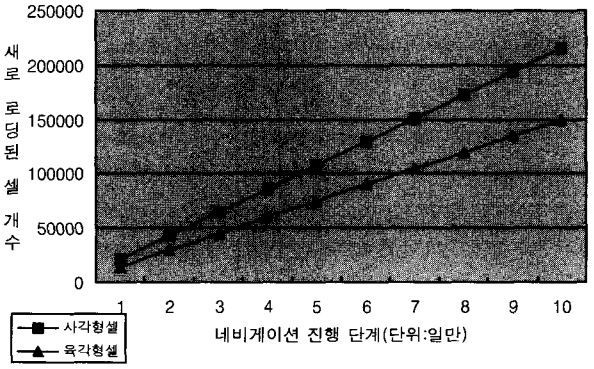


(그림 4-4) 전체 로딩된 셀 영역 차이

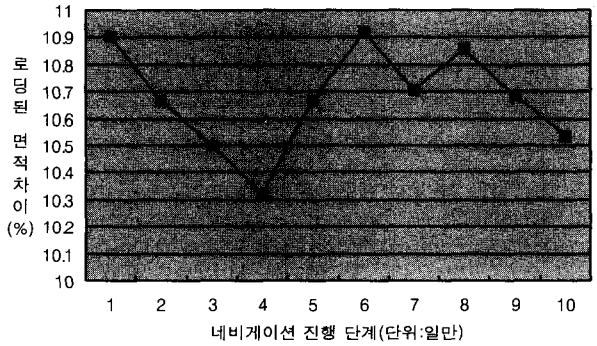


(그림 4-5) 전체 로딩된 셀 개수

그 다음으로 실시간 네비게이션이 진행될 경우를 살펴보자. 사각형셀과 육각형셀의 각각에 대하여 네비게이션을 매 10,000단계씩 최대 100,000단계를 진행했을 경우의 결과를 살펴본다. 실시간 네비게이션을 진행하면서 새로 로딩된 셀의 개수는 (그림 4-6)에 나타나 있듯이 육각형셀 방식이 평균 30% 정도 적음을 알 수 있다. 또한 이를 영역으로 환산해 보면 (그림 4-7)에 있듯이 평균 10.6% 정도 줄어짐을 알 수 있다.



(그림 4-6) 새로 로딩된 셀 개수



(그림 4-7) 전체 로딩된 셀 영역 차이

따라서 육각형셀 방식이 사각형셀 방식보다 일반적인 네비게이션에서 뿐만 아니라, 실시간 네비게이션에서도 더욱 효율적인 방식이라 할 수 있다.

5. 결 론

가상공간에서의 네비게이션이 원활하게 진행되려면 전체 맵을 저장공간에 한꺼번에 올려 놓으면 좋으나 이 방법은 저장공간이 많이 필요하게 되는 단점이 있다. 또한 이러한 맵 자료를 네트워크 상에서 받아야 한다면 전체 자료를 한꺼번에 받는 방식은 좋은 방법이 아니다. 그리고 대부분의 가상공간 네비게이션은 대부분 실시간 반응성을 요구하게 되는데, 오프라인 시스템이면 보조기억장치를 이용하여 많은 데이타를 저장해 두고 필요에 따라 주기억장치나 캐시를 사용하면 된다. 그러나 웹기반의 온라인 시스템이면

데이터 전송시간이 실시간 반응성에 영향을 주기 때문에 새로운 방식의 로딩 방식이 필요하게 된다. 본 연구에서는 두 가지 방식의 셀 로딩 기법을 제시하고, 특히 웹기반의 가상공간 네비게이션에서 실시간 반응성을 향상시킬 수 있는 육각형 셀로딩 방식을 제안한다. 사각형셀 방식은 기준 공간을 9개의 영역으로 표현하는 대신 육각형셀 방식은 7개의 영역으로 표현하므로 셀로딩 개수로 판단할 때에는 육각형 셀 방식이 더 좋다는 것을 알 수 있다. 그리고 모든 방향으로의 시야 확보성을 비교해 보면, 원의 면적에서 셀 영역이 차지하는 비율이 사각형셀은 약 64%이고, 육각형셀은 83%가 되어 육각형 셀 방식이 사각형 셀 방식 보다 훨씬 더 충실하게 시야를 확보 할 수 있다는 것을 알 수 있다. 또한 셀영역의 최외곽에서 원둘레에 이르는 최대거리비율이 30%이고, 육각형셀은 24%로써 방향성이 더 좋다는 것을 알 수 있다. 그리고 실시간 네비게이션에서도 정사각형 방식에 비해 로드되는 셀 개수가 평균적으로 약 30% 정도 줄어, 큰 성능향상을 보였다. 물론 육각형 셀 방식은 인덱스 계산이 약간 복잡하다는 단점이 있으나, 이 또한 실시간 계산에 지장이 있을 정도는 아니다. 본 논문에서 제안한 네비게이션 셀 로딩 알고리즘을 이용하여 실시간 게임분야, 실시간 역사 학습 교과 코스웨어, 실시간 박물관 등과 같은 다양한 가상공간을 생동감 있게 네비게이션을 할 수 있고, 시간적 속도성으로 인한 사용자의 다중참여도 가능하게 될 것으로 판단된다.

향후 과제로는, 현재의 알고리즘을 2차원적인 맵에서 3차원 맵 로딩에 대한 알고리즘으로 확장하는 연구가 필요하다.

참 고 문 헌

[1] 원광연, "전산학으로서의 가상현실", 정보과학회지, 제15권 제11호, pp.5-13, 1997.

[2] 김남국, 김철영, 김영호, 강석호, "VRML을 이용한 3D 형상정보 Web Server 및 Browser 개발", 한국 CAD/CAE 학회 학술발표회 논문집.

[3] Naim Alper, Mar, Inc. "Geospatial Metadata Querying and Visualization on the WWW Using Java Applets," Proceedings IEEE Symposium on Information Visualization '96 1996.10 ; IEEE Computer Society, pp.77-84, 1996.

[4] 박병주, 이진호, 최윤철, "VRML 저작도구의 설계 및 구현", 한국정보과학회 봄 학술발표논문집, Vol.25, No.1, 1998.

[5] 최진성, 박찬용, 최정단, 김동현, 조맹섭, "저가형 가상현실 저작도구 개발에 관한 연구," 한국정보과학회 봄 학술발표논문집B, Vol.24-B, 1997.

[6] Ramloll R. and Mowat D., "Wayfinding in virtual environments using an interactive spatial cognitive map," *Proceedings, Fifth International Conference on Information* Visualization, pp.574-583, 25-27, July, 2001.

[7] Heilig M. L., "The cinema of the future," Presence, 1(3), pp.279-294, 1992.

[8] 강상우 외 4인, "가상환경하에서 현실 체감도를 높이기 위한 다양한 형태의 감각 변수들의 영향도 평가", HCI '99 Conference, Vol.8, No.1, pp.356-363, 1999.

[9] Birebent G, Coiffet et al, "Effect of Frame Rate and Force Feedback on Virtual Object Manipulation," Presence, 5(1), pp.95-101, 1996.

[10] 강상우, "가상환경하에서 시선정보를 이용한 LOD 적용에 관한 연구", 한림대학교 컴퓨터공학과 석사학위논문, Feb., 2001.

[11] 김양수, "지형데이터를 위한 효율적인 단계별상세(LOD) 표현방법", 부산대학교 전자계산학과 석사학위논문, Feb., 2000.

[12] Barfield W. and Hendrix C., "The effect of update rate on the sense of presence in virtual environments," Virtual Relity : Reaearch, Development, Applications, 1(1), pp.3-15, 1995.

[13] Barfield W. and Hendrix C., "Presence within virtual environments as a function of visual display parameters," Presence : Teleoperators and Virtual Environments, 5(3), pp.274-289, 1996.

[14] 신문균, "눈의생리학", 현문사, 1995.

이 기 동

e-mail : kdrhea@yu.ac.kr

1985년 서울대학교 공과대학 제어계측공학과(학사)

1987년 서울대학교 공과대학 제어계측공학과(석사)

1994년 서울대학교 공과대학 제어계측공학과(박사)

1995년~현재 영남대학교 전자정보공학부 부교수

관심분야 : 인공지능, 로보틱스, 멀티미디어 정보처리, 정보보안

하 주 한

e-mail : innogeo@daum.net

1998년 경일대학교 공과대학 전자공학과(학사)

2000년 영남대학교 공과대학 컴퓨터공학과(석사)

2000년~현재 (주)이노지오 대표