가상공간에서 실시간 네비게이션을 위한 셀 로딩 알고리즘

이 기 동 손 정 봉 최 창 은
영남대학교 컴퓨터공학과

Cell Loading Algorithm for the Purpose of Realtime Navigation in Virtual Space

Kidong Lee Jungbong Son Changeun Choi
YeoungNam Univ.
kdrhee@yu.ac.kr zec3490@yumail.ac.kr country1357@yumail.ac.kr

요 약

3차원 실세계를 2차원 평면상에 표현하기에는 화면의 복잡도, 정보의 왜곡, 생동감의 결핍 등의 한계가 뒤따른다. 또한 각 객체의 디자인 한계성과 지속적인 데이터 양의 갱신이 필수조건이지만 실시간 네비게이션의 복잡성과 많은 데이터양의 활동에 있어 속도감의 문제가 발생한다. 이를 해결하기 위해 실시간 네비게이션 셀로딩 알고리즘을 제안한다.

ABSTRACT

while they cannot overcome the limitation that arises in the process of representing the 3D real world to the 2D plane. Also, there exists requirements on performance to support realtime navigation capability.In this paper, therefore, we propose a cell loading algorithm for navigating virtual space that can support realtime visualization according to the multimedia objects in variety and the change of the view point by user, and that can accommodate the capacity imposed in the process of navigation regardless of the number of objects.

Ⅰ.서론

해마다 다양한 형태의 가상공간 네비게이션 뷰어가 소개되고 있지만 아직까지는 3차원 실세계를 2차원 평면상에 표현하기에는 화면의 복잡도, 정보의 왜곡, 생동감의 결핍 등의 한계가 뒤따른다. 또한 각 객체의 디자인 한계성과 지속적인 데이터 양의 갱신이 필수조건이지만 실시간 네비게이션의 복잡성과 많은 데이터양의 활동에 있어 속도감의 문제가 발생한다.

그래서 실시간 네비게이션을 하기 위해서 적절한 공간확보 및 데이터양을 고려하여 저작할 수 있는 네비게이션 알고리즘 개발이 중요하다.

본 논문에서는 이러한 문제점들을 인식하고 뷰어에 있어서 다양한 멀티미디어 객체들을 제작하

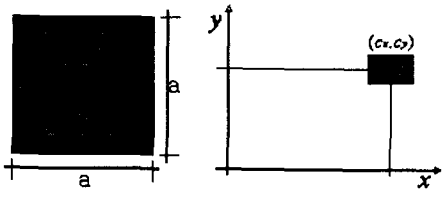
고 각 객체의 회전, 크기, 이미지, 색상 그리고 문자 등을 사용자와 상호작용 할 수 있도록 하고 이를 바탕으로 하여 복잡한 데이터 정보를 실시간으로 시각화하여 네비게이션을 볼 수 있도록 하며 방향성에 따른 네비게이션 셀 로딩 알고리즘 기법을 연구하는 것을 목적으로 한다.

먼저 기존의 네비게이션 셀 로딩 알고리즘 기법을 설명하고 실시간으로 시각화하는 네비게이션 셀 로딩 알고리즘 기법을 제안하도록 한다.

Ⅱ.본론

2. 정사각형 셀 로딩 알고리즘

기존의 셀로딩 알고리즘 방식은 정사각형방식이 있다. 각 정사각형 셀 영역의 크기는 이동 방향과 시각적 네비게이션에 의해 결정되는데 이것은 실제로 사람이 볼수 있는 시야에 의한 면적을 하나의 정사각형 셀로 가정한 것이다. 셀 영역의 위치 정보를 알려면 먼저 블록으로 잡은 셀의 위치, 즉 좌표 값을 알아야 한다. 셀 영역의 계산을 위해 그림 2-1(a)과 같이 한 변의 길이가 a인 정사각형 블록과 그에 해당하는 셀 영역 위치 좌표를 구하기 위해서 식(2-1)을 이용한다. 여기서 INT[x]란 x를 넘지 않는 최대 정수값을 나타낸다.



<그림 2-1 (a)> 셀 영역 이동 방향 및 위치좌표

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

이를 통하여 그 영역에서의 이동방향과 시각적 네비게이션을 위한 정사각형들로 한 셀 중심으로 8개의 정사각형 셀 영역을 로드하여 준다.

이렇게 해서 중심의 셀 영역을 포함한 <그림 2-1(b)>와 같이 8가지의 방향성을 가진다.

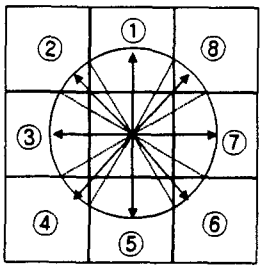


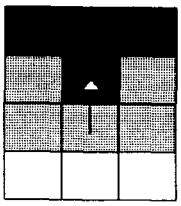
그림 <2-1 (b)>

2.1 정사각형 셀 로딩 정보

본 절에서는 이동 방향에 따른 8방향의 로드 순서와 시각적인 확보에 의한 셀 영역 확보에 대한 사항을 상세히 기술한다.

2.1.1 방향성에 의한 로드

위의 <그림 2-1 (b)>의 ①번 방향에서 ⑧번 방향까지 이동할 때 <그림 2-2>처럼 각각의 셀 인덱스가 로드되어지는 순서와 그에 따른 좌표 값을 구할 수가 있다. 먼저 ①번 방향으로 이동하면 <그림 2-2>처럼 로드되어진다. <그림 2-2>에서 ①번 셀 영역으로 이동하면 먼저 전진성에 의해 전방의 1번 셀 영역이 먼저 로드되고 다음은 시각적인 방향이 왼쪽에서 오른쪽으로 흐르는 것을 감안하여 2번을 로드하게 되고 마지막으로 3번째 영역을 로드하게 된다. 이에 따라 위치좌표를 이용하여 간단히 셀 영역의 좌표를 구할 수 있다.



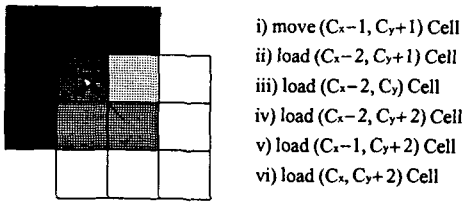
<그림 2-2> ①번 셀 영역 이동

셀로딩 방식은 전방과 같이 후방이나 우측방향의 로드순서도 같은 순서이다.

그러나, 대각선 방향의 셀 로딩방식은 이와는 조금 다르다.

<그림 2-3>에서는 ②번 셀 영역으로 이동하면 먼저 사용자 위치의 회전방향에 따라 로드되는 우선순위가 달라진다. 여기에서, 5개의 셀 영역이 로드되고, 로드되는 순서는 <그림 2-2>에 표시되

어 있는 ①번 셀 영역에서와는 달리 전방에 있는 셀 영역을 우선적으로 로드하지 않는다. 우선 시점이 왼쪽에서 오른쪽으로 흐르는 것을 감안하여 좌측 전방에 있는 셀 영역을 우선적으로 로드하여 주고 다음은 우측에 있는 셀 영역을 로드하게 된다. 이것은 회전 방향성에 따른 것이며 이에 문제점에 의해 새로 제안된 알고리즘에서 많이 언급할 것이다.



<그림 2-3> ②번 셀 영역 이동

이와 같은방식으로 각각의 대각선 방향의 셀 로딩은 이러한 순서로 로딩되어진다.

2.4 기존 네비게이션 셀 로딩 알고리즘 문제점

네비게이션 상에서 이동을 할 경우 정면이나 후면, 좌측, 우측의 경우에는 정사각형의 셀을 이용하여도 크게 무리가 없다. 그러나, 정면, 좌측, 우측이 아닌 기타 다른 각도로 시점을 이동할 경우 시점의 확보에 불필요한 많은 것을 볼 수 있다.

시점이 정면과 좌측, 우측을 제외한 다른 방향으로 이동하는 경우는 네비게이션 시스템에서 확보해 줘야 할 시야가 왜곡되어 질 수 있다. 꼭 확보되어야 할 시야가 나타나지 않는 경우도 있으며, 불필요한 시야가 확보가 될 수도 있다. 또한, 진행하는 방향의 정면에 있는 셀 영역이 우선순위에서 지연됨으로써 빠르게 로딩 되어지지 않는 문제점이 나타날 수 있다.

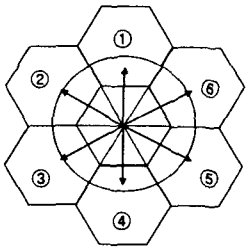
3. 개선된 네비게이션 셀 로딩 알고리즘

정사각형에서의 장점은 먼저, 위치정보의 계산이 쉽다는 점을 빼놓을 수 없다. 그러나, 정육각형 셀은 시각적인 확보는 물론 실시간 네비게이션을 하기에는 아주 적합하지만 계산량이 복잡한 단점이 있다. 그래서 미리 셀 영역의 인덱스를 계산하기 때문에 실시간 네비게이션 하기에 무리가 없다.

3.1 네비게이션 셀 로딩 알고리즘 기법 제안

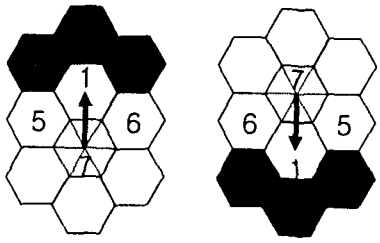
정육각형 셀 영역 네비게이션 셀 로딩의 특징과 방향에서의 시각적 확보를 살펴보고 이어서 정 육각형 셀 영역의 위치정보를 구하는 식을 제안 해 본다. <그림 3-1>은 정육각형 셀 영역의 중심에서 각 방향성을 나타낸 것이다.

그림에서 보듯이, 한 셀을 중심으로 여섯 개의 셀영역을 로드하여 준다.



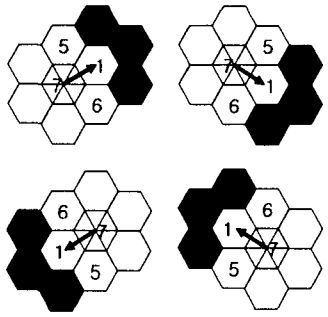
<그림 3-1> 정육각형 셀 영역 방향성

다음은 셀 영역 로딩순서에 대하여 살펴보자



<그림 3-2> 셀 영역의
로드순위(I)

<그림 3-2>에서 화살표 셀 영역으로 이동하면 전진성에 의해 전방의 2번 셀 영역이 먼저 로드되고 다음은 시각적인 방향이 왼쪽에서 오른쪽으로 흐르는 것을 감안하여 3번을 로드하게 되고 마지막으로 4번째 영역을 로드하게 된다. <그림 3-2>에서 보는 것과 정사각형 셀 영역 에 대한 전방의 이동방향과 후방의 이동방향 로드순위는 크게 차이 나는 것이 없음을 볼 수 있다.



<그림 3-3>셀 영역의 로드순위(II)

또한, <그림 3-3>에서 대각선 셀 영역으로 이동하면 셀 영역이 3개가 로드되어지는 것을 볼 수 있다. 그러나 정사각형의 셀 영역에서 5개가 로드되어지는 것을 볼 수 있었다. 그리고 정사각형 셀 영역의 우선순위를 회전각을 이용해야 하는 문제점과 방향에 의해 로드되는 순위가 달라지는 것과 비교하면 정육각형이 가지고 있는 회전각과 방향성의 특성을 이용하여 위의 <그림 3-3>셀 영역과 마찬가지로 전방에 있는 셀 영역을 우선적으로 로드되기 때문에 셀 영역의 대각선 이동 부분은 항상 전방성을 가지고 있어 정육각형 셀 영역이 적당하다고 보여진다.

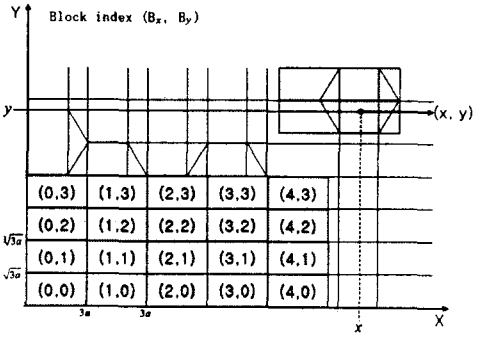
다음은 구체적으로 셀 영역의 위치 정보와 블록단위를 계산할 수 있는 방법 및 정육각형이 가지고 있는 특성을 잘 파악하여 계산식을 구해 보고자 한다.

3.1.1 블록 인덱스

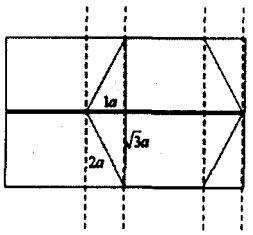
셀 영역을 구하기 위해서는 3가지 방법을 이용해야만 구할 수 있다.

<그림 3-4>에서 블록을 x좌표 값과 y좌표 값을 기준을 (0, 0)을 한 블록단위로 영역을 잡게 된다. 그러면 각각의 블록 영역들은 (1, 0) (2, 0) (3, 0)...등과 같이 x좌표의 값으로 블록 영역이 설정된다. 그리고 y좌표 값들은 (0, 1) (0, 2) (0, 3)... 등으로 블록이 설정된 것을 볼 수 있다.

먼저 정육각형 블록 인덱스를 구하는 방법은 <그림 3-5>와 같이 한 변의 길이를 2a를 둔다. 그러면 직각 삼각형의 피타고라스 정의를 이용하여 각각의 값을 구할 수 있다.



<그림 3-4> 블록인덱스 셀 영역



<그림 3-5> 셀 영역 값

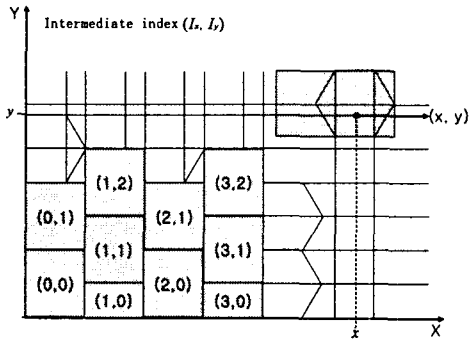
이러한 처리과정에 따른 한 블록을 식(3-1)을 이용하여 쉽게 구할 수 있다. 예를 들면, <그림 3-4>에서 위치 좌표값이 (10, 6)라는 좌표값을 가진 위치를 가지고 있다면 해당 블록영역을 식(3-1)으로 구해보면 Bx값은 3이라는 좌표값이 나오고 By값은 3이라는 좌표값을 가진다. 그러면 해당 블록은 (3, 3)이라는 블록영역의 좌표값을 구할 수 있다.

3.1.2 중간 인덱스

블록 인덱스를 구하고 나면 두 번째로 중간 인덱스 값을 구해야한다. 구하는 식과 방법은 다음 <그림 3-6>과 같다.

블록 인덱스를 구할 때 정육각형의 특성상 <그림 3-6>과 같이 셀 영역의 정육각형들이 짝수 홀수의 두가지 값을 가지는 특서을 볼 수 있다. 그

렇게 해서 식(3-2)을 이용해 블록단위에서의 중간 영역 인덱스 값을 구해야 한다.



<그림 3-6> 중간 인덱스 셀 영역

$$I_x = B_x$$

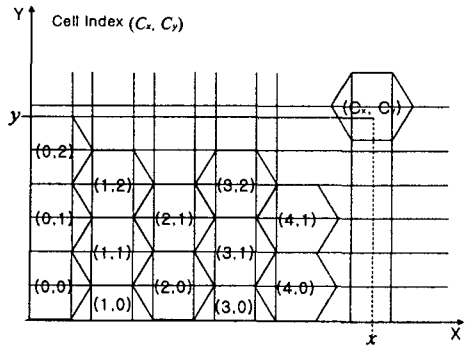
$$I_y = \begin{cases} INT\,[B_y/2] & f\ I_x = even\ (0\text{포함}) \\ INT\,[(B_y+1)/2] & \text{if}\ I_x = odd \end{cases}$$

식 (3-2)

<그림 3-6>과 같이 식(3-2)를 이용하여 간단히 블록 인덱스 값을 가지고 중간영역 인덱스 값까지 구할 수가 있다. 최종적으로 셀 인덱스 값을 구해주어야한다. 셀 인덱스 값을 구하는 것은 각각의 블록에 대해서 셀 영역으로 구한 것이기 때문에 어느 위치의 정육각형 셀 영역인지를 구해야 하기 때문이다.

3.1.3 셀 인덱스

셀 인덱스 값을 구하면 정육각형의 셀 영역을 구할 수가 있다. <그림 3-7>는 셀 인덱스 구하는 방법이다.



<그림 3-7> 셀 인덱스 영역

좌측 하단을 원점으로 설정한 후 평행 이동을 하면 몇 가지 공통점을 보인다. 그것은 정육각형 특성상 원점을 기준으로 평행 이동을 하게 되면 4가지의 정육각형 선분의 기울기를 알수 있다. 먼저 평행이동을 해주는 식을 정의하면 평행이동(x, y)는 (Xt, Yt)와 같다. 다음 식(3-3)을 이용하여 평행이동 값을 구한다.

$$x_t = (x - 3a\,B_x)$$

$$y_t = (y - \sqrt{3a}\,B_y)$$

식(3-3)

그리고, 4가지의 선분의 방향을 각각 대입하면 어느 정육각형 셀 영역으로 이동했는지를 판단할 수 있다. 다음들의 그림들과 식들은 4가지 선분 방향에 대하여 구하는 계산식을 정의하고 살펴본다. 첫 번째는, Bx 값이 짝수이고 이때 0도 짝수로 포함한다. 그리고, By값도 짝수일 때 선분 방향에 의한 구하는 식(3-4)를 정의한다.

$$If\ (B_x = even(0\text{포함})\ AND\ B_y = even)$$

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

식 (3 - 4)

두 번째는, Bx값이 짝수이고 이때 0도 짝수로 포함한다. 그리고, By값이 홀수일 때 선분방향에 의한 구하는 식(3-5)를 정의한다.

$$If\ (B_x = odd(0\text{포함})\ AND\ B_y = odd)$$

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

식(3-5)

세 번째는, Bx값이 홀수이고 이때 0도 포함한다. 그리고, By값도 짝수일 때 선분방향에 의한 구하는 식(3-6)을 정의한다.

$$If\ (B_x = odd\ AND\ B_y = even)$$

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

식(3-6)

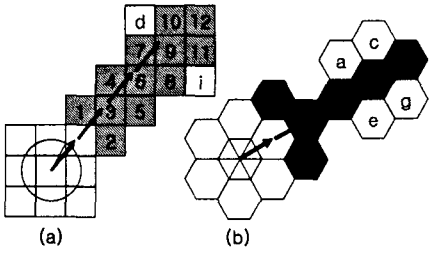
네 번째는, Bx값이 홀수이고 그리고, By값도 짝수 일 때 선분 방향에 의한 구하는 식(3-7)을 정의한다.

$$If\ \ (B_x = even(0포함)\ AND\ B_y = odd)$$

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

식(3-7)

3.4 정육각형 셀 영역의 네비게이션 셀 로딩 알고리즘 평가



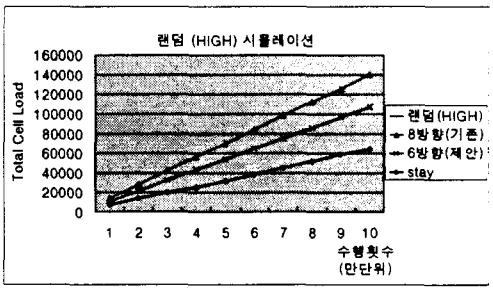
<그림 3-8> 연속적 셀 영역 이동(II)

<그림 3-8(a)>에서 심각한 이유는 즉, 대각선으로 빠르게 이동하면 사용자는 대각선 방향으로 전진하는 것으로 정사각형 셀 영역에서의 방향성에 따른 로드정보에서 보면 알 수 있듯이 시각적인 확보에 따라 좌측에서 우측으로 셀 영역을 확보함에 따라 빨리 이동할 경우에는 셀 영역을 로드하여 주는데 무리가 많이 따른다. 왜냐하면 사용자의 시각방향은 대각선의 전방 방향을 보고 이동하지만 사실상 셀 확보하는 부분은 좌측 셀부터 확보하고 있기 때문에 항상 전방에 셀 영역은 로딩되는 시간이 지연되는 문제가 있다. 반면 <그림 3-8(b)>에서는 정사각형 셀처럼 대각선 방향으로 연속적으로 이동했을 경우에도 항상 전진성을 갖고 있기 때문에 전방 셀을 로드하고 좌측 또는 우측 셀 영역들도 시각적으로 같이 확보하여 로드하기 때문에 네비게이션을 실시간으로 수행하는 경우 사실상 사용자가 많이 기다려야 할 필요가 없다. 그리고 시각적인 왜곡면들이 정사각형 셀 로딩 보다 시간적으로 로딩되는 속도가 빠르다.

4. 네비게이션 셀 로딩 시뮬레이션 실험 및 결과

각각의 시뮬레이션에 의한 셀 영역에 대한 성능평가를 시뮬레이션으로 통해 비교 설명 하고

자 한다. 네비게이션 셀 로딩의 시뮬레이션을 각각 선택버튼에서 10000번씩의 임의의 값으로 주어질 때 수행한 결과를 STAY부분에서 선택되었을 때 결과와 그림에 대한 비교차트를 보여준다.



<그림 4-9> 랜덤의 Stay="HIGH" 시뮬레이션

STAY선택영역에서 HIGH를 선택했을 때 반복횟수에 따른 시뮬레이션 한 결과는 기존의 네비게이션 할 때와 제안된 알고리즘으로 네비게이션 할 때 제안된 네비게이션의 로드 개수가 약 25.9% 정도가 줄어들었다..

<그림4-10>은 결과를 바탕으로 차트를 보여준다.

III 결론

셀 로딩 알고리즘을 제안하여 실험 성능 평가를 함으로서 기존의 정사각형 네비게이션 셀 로딩보다 정육각형 셀 영역의 네비게이션 셀 로딩이 공간확보 및 데이터량에 상관없이 실시간 네비게이션을 수행하기에 적합하였다. 또한 방향성에 따른 사용자의 시점에 따라 시각적인 확보 및 각 객체들의 확보까지도 표현이 가능하였다.

본 논문에서 제안한 네비게이션 셀 로딩 알고리즘을 이용하여 실시간 게임 분야, 실시간 역사 학습 교과 코스웨어, 실시간 박물관 등 과 같은 다양한 적용사례에 생동감 있는 가상공간을 저작할 수 있고 시간적 속도성에 따른 사용자의 다중참여도 많아질 것으로 기대된다.

6. 참고 문헌

[1] 고범석외, "Internet과 VRML", 한국정보처리학회 1996 9v.3 n.5 pp.69-81

[2] 송경준외, "분산협동 가상현실 미들웨어 개발", 정보과학회지 제15권 제11호, p.20-25,1997.11.

[3] 하주한, 이기동, "VRML을 이용한 Web기반의 가상공간 저작도구 구현", 99 한국정보과학회 가을 학술발표 논문집, pp632-634, 1999년
[4] Laura Lemay, Kelly Murdock, Justin Couch. "3D Graphics & VRML 2.0"
[5] K. Matsuda, Y. Honda and R. Lea, "Sony's approach to behavior and scripting aspects of VRML:an Object Oriented Perspective" http://www.csl.sony.co.jp/project/VS/proposal/behascri.html
[6] VRML org. VRML Library, QvLib, http://www.vrml.org
[7] Ken Brodlie, Jason Wood and Helen Wright, "Scientific visualization - some novel approaches to learning," ACM SIGCSE Bulletin, Vol.28, pp.28--32. (1996)