영역패턴을 이용한 효과적인 사각형 이미지 절단 알고리즘

최경진*
*고려대학교 컴퓨터과학기술대학원 멀티미디어학과
e-mail:chankil@korea.ac.kr

An Efficient Rectangle Image Clipping Algorithm Using Region Pattern

Kyoung-Jin Choi*
*Dept of Multimedia, Graduate School of Computer Science & Engineering, Korea University

요 약

본 논문에서는 2차원 공간상에서 윈도우에 출력되는 사각형 이미지가 윈도우 영역을 벗어날 때 영역패턴을 이용한 절단 알고리즘을 제안한다. 제안하는 알고리즘은 Cohen-Sutherland의 알고리즘을 응용하여 사각형 이미지의 절단영역을 계산하는데, 약 4회의 조건문장과 1회의 분기문장을 사용하여 수행되므로 기존의 방법보다 전체적인 수행 성능이 약 12% 좋게 나타났으며, 특히 확장영역에 있어 기존의 방법은 처리하지 못하는 부분까지도 완벽하게 처리됨을 확인 할 수 있었다.

1. 서론

2차원 윈도우 영역에서 출력되는 데이터는 경우에 따라서 윈도우의 영역을 벗어나게 된다. 이 때, 윈도우의 영역을 벗어나는 부분을 적절하게 잘라내어 무효화를 시켜주어야 하는데, 이는 디스플레이 시스템의 오류를 방지하고 데이터의 출력 성능을 향상시키는데 있어서 매우 중요하기 때문이다.

Cohen-Sutherland의 Line Clipping 알고리즘과 Cyrus-Beck, Liang-Barsky의 알고리즘 등은 윈도우 영역을 벗어나는 선분의 효율적인 절단 방법을 제안하고 있으며[1,2,3], 임의의 다각형에 대한 효율적인 절단 방법들도 제안되었다[4,5,6,7].

최근에 사용되는 운영체제에서는 절단 방법에 대하여 다양하고 유연성 있는 인터페이스를 제공하고 있다. 그러나 게임분야와 같이 실시간으로 사각형 이미지 데이터를 입출력하는 경우에는 절단 알고리즘과 원시코드를 개발자가 직접 작성해야 하는 사례가 발생하게 된다. 이는 사용되는 알고리즘에 따라 결과물의 성능에 대한 편차가 발생함을 의미하며, 편차를 줄이기 위해서는 보다 객관성 있는 알고리즘의 필요성이 요구된다.

본 논문에서는 영역패턴을 이용하여 효율적으로 사각형 이미지를 절단하는 알고리즘을 제시하고자 한다. 이 알고리즘은 [1]에서 사용된 4비트를 영역패턴 코드로 사용하고 있다. 또한, 사각형 이미지에 대한 절단을 수행하므로 예외상황에 대한 처리를 위해 7가지의 영역패턴을 추가하였다. 이 방법은 약 4회의 조건문장과 1회의 분기명령을 가지고 수행되며, 알고리즘이 간단하다는 장점을 가지고 있다.

다음에 이어지는 2장에서는 절단 알고리즘에 대한 관련연구를 살펴볼 것이며, 3장에서는 영역패턴을 이용한 사각형 이미지 절단 알고리즘을 기술하고, 4장에서는 수행성능을 평가하고 결과에 대해서 기술할 것이며, 마지막으로 5장에서는 결론을 맺는다.

2. 관련연구

2차원 윈도우에서의 절단 알고리즘을 두 가지로 나누면, 하나는 선분을 절단하는 방법이고, 다른 하나는 다각형을 절단하는 방법이다. 그러나 다각형을 절단하는 방법은 본 논문의 논지로 볼 때 알고리즘의 복잡도가 높으므로 제외하였다[4,5,6,7].

선분을 절단하는 알고리즘은 Cohen-Sutherland의 방법과 Liang-Barsky의 방법이 가장 많이 알려져 있다[1,3]. 우선 [1]의 방법은 선분의 양 끝점에 대해

서 4비트의 2진 영역코드를 부여하고 절단 영역을 계산하는데 이용하고 있다. 이 알고리즘은 선분의 양끝 점의 좌표를 검사하여 영역코드를 판별하게 되며, bit1과 bit2는 x축 영역을, bit3과 bit4는 각각 y축의 영역에 할당되어 있다. 이렇게 두 점의 영역코드를 산출한 후, 논리곱 연산을 통해서 얻어진 결과값으로 선분의 출력여부를 결정할 수 있으며, 절단할 선분의 구간을 간단하게 검출 할 수 있다는 장점을 지니고 있다.

이에 반하여 [3]의 방법은 보이지 않는 선분의 빠른 제거를 통해 효율성을 강조하고 있다. 이 알고리즘은 매개 변수 방정식을 사용하고 있으며, 네 개의 부등식을 가지고 경계내부에 선분의 존재 여부를 판단하게 된다. 또한, 경계 내부에 있는 선분을 나타내는 u1과 u2의 값을 계산하여 경계선을 정의하게 되는데, 2차원, 3차원의 공간상에서 쉽게 일반화 될 수 있다는 장점을 가지고 있다.

마찬가지로, [5]에서 제시하고 있는 동차좌표계를 이용한 절단 알고리즘은 동차계수 값이 0보다 커야 한다는 조건아래 x와 y좌표의 범위를 정하고 동차계수와 x, y성분에 대한 조건들을 정하게 된다. 이 방법은 3차원 공간까지도 응용 될 수 있기 때문에 [1]의 방법보다 비교적 복잡한 과정을 거치게 된다.

이와 같이, 위에서 살펴본 선분절단 알고리즘들은 단순한 선분을 포함하여 기울기가 있는 선분까지도 고려하고 있다. 또한 차원(Dimension) 까지도 고려한 알고리즘도 있는데, 복잡도가 높은 알고리즘은 기울기가 없는 사각형 이미지의 절단에는 적합하지 않다. 따라서 활용가치와 효율성을 우선적으로 고려하여 볼 때, 사각형 이미지의 절단에는 [1]의 알고리즘이 가장 적합한 요건을 지니고 있다고 판단된다.

3. 영역패턴을 이용한 사각형 이미지 절단

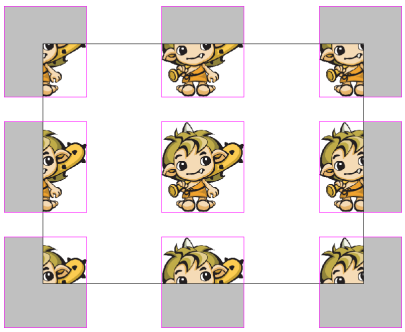
3.1 알고리즘의 적용방법

영역패턴을 이용한 사각형 이미지 절단 알고리즘은 [1]에와 같이 4비트를 사용한다. 비트 값들은 각각 0부터 15까지의 범위를 가지며, 이 값은 2차원 윈도우에 출력되는 사각형 이미지가 윈도우의 어느 영역에 출력되는가를 결정하게 된다. 즉, 이 값은 패턴코드를 의미하는 것이며<표 2>, 사각형 이미지 데이터의 무효영역을 어떻게 절단해야 하는지를 구분시켜 주게 된다.

<표 1> 조건식에 따른 비트별 조건 테이블

bit	조건식	조건만족	초기값	10진수
3	x < WinMinX	1	0	8
2	y < WinMinY	1	0	4
1	x+width ≥ WinMaxX	1	0	2
0	y+height ≥ WinMaxY	1	0	1

본 논문에서는 [1]에서 제시한 알고리즘에 착안하여 <표 1>의 테이블을 작성 하였다. (그림 1)은 <표 1>의 비트를 조합한 9가지 패턴을 보여주고 있으며, 가운데 영역은 절단이 필요 없음을 의미한다.



(그림 1) 9가지 형태의 절단영역

<표 2>는 (그림 1)과 같이 9가지 형태의 절단 영역에 대하여 할당된 패턴코드를 정리 한 것이다.

<표 2> Cohen-Sutherland 에 기초를 둔 코드 테이블

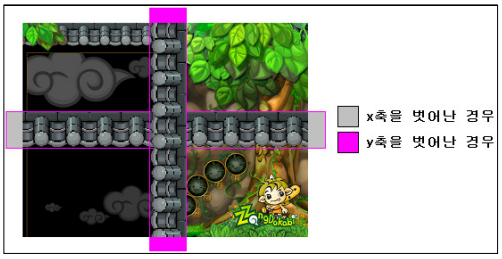
영역구분	패턴 코드	구분 숫자	비고
0	1100	12	Normal
1	0100	4	
2	0110	6	
3	1000	8	
4	0000	0	
5	0010	2	
6	1001	9	
7	0001	1	
8	0011	3	

3.2 예외영역에 대한 Cohen-Sutherland코드 보완

2차원 윈도우에 출력되는 사각형 이미지는 아래의 (그림 2,3,4) 에서와 같이 7가지의 예외적인 영역들이 발생하게 된다. 이러한 원인은 출력하는 사각형 이미지가 대상 윈도우의 일부 또는 전체영역을 뒤덮어버리기 때문이며, <표 2>에서 제시한 9가지 코드로는 예외적인 영역들을 해결할 수 없었다.

본 논문에서는 이러한 점을 해결하기 위해 패턴코드를 추가로 할당하였으며 <표 3>과 같이 예외영역을 쉽게 판별할 수 있도록 문제점을 개선하였다.

(그림 2) 상하좌우 영역을 벗어난 경우



(그림 3) x, y의 축을 벗어나는 경우

(그림 4) 전체를 벗어나는 경우

이와 같이 개선한 결과, 영역패턴을 나타내는 경우의 수는 일반적인 9가지의 경우와 예외적인 7가지 경우를 합하여 16가지가 됨을 알 수 있다<표 2,3>.

<표 3> 예외영역을 위해 보완된 확장 패턴 테이블

영역구분	패턴코드	구분 숫자	비고
좌측영역 벗어남	1101	13	Extended
우측영역 벗어남	0111	7	
상단영역 벗어남	1110	14	
하단영역 벗어남	1011	11	
x축 영역 벗어남	1010	10	
y축 영역 벗어남	0101	5	
전체영역 벗어남	1111	15	

본 알고리즘이 [1]의 방법과 다른 점은 두 가지가 있다. 첫째, [1]의 방법에서는 두 점의 영역코드를 가지고 논리연산을 수행하지만, 본 알고리즘은 4회의 조건검사로 얻어진 값을 이용하여 절단을 수행한다. 둘째, [1]의 방법에서는 예외처리를 위한코드가 없으나, 본 알고리즘은 예외처리를 위해 7가지의 영역패턴을 확장하여 주었다는 점이다<표 2,3>.

(그림 5)는 C 언어를 이용하여 작성된 예 이다.

```
unsigned char patt_code = 0;

if( x + width <= WinMinX ) return;    .....(5.1)
if( x < WinMinX ) patt_code |= 0x08;

if( y + height <= WinMinY ) return;   .....(5.2)
if( y < WinMinY ) patt_code |= 0x04;

if( x >= WinMaxX ) return;            .....(5.3)
if( x+width  >= WinMaxX ) patt_code |= 0x02;

if( y >= WinMaxY ) return;            .....(5.4)
if( y+height >= WinMaxY ) patt_code |= 0x01;
```

(그림 5) C 언어로 작성된 알고리즘의 예

(그림 5)의 (문장 5.1-5.4)는 출력영역을 완전하게 벗어날 경우의 버림 조건 코드이다. 변수 patt_code는 조건식을 만족하는 경우에만 해당하는 비트를 1로 설정해 주고, 4번의 영역 검사를 통해 값이 완성된다<표 1>. 이제 완성된 값을 이용하여 <표 2>를 참조하면서 사각형 이미지를 절단하면 된다.

4. 실험 및 평가

실험은 Intel Pentium-III 750Mhz Notebook PC 에서 Windows 2000 운영체제, Microsoft Visual C++ 6.0을 사용해서 시행되었다. 실험 방법은 제안된 알고리즘의 객관성을 유지하기 위해 웹에 게시된 알고리즘을 이용했으며[8], 수행범위는 <표 4>와 같이 일반, 외부, 확장영역으로 나누었다. 수행성능은 16가지 영역(그림 1,2,3,4) 데이터를 입력하고 500,000회를 실행하여 1/1000ms 단위로 시간을 기록하고 이를 10회 반복하여 평균값을 가지고 측정하였다.

<표 4> 알고리즘의 수행범위 비교

구분 \ 범위	일반영역	외부영역	확장영역
[8]의 방법	O	처리못함	처리못함
본 연구 방법	O	O	O

<표 4>에서 보는바와 같이 본 연구에서 제시한 방법은 모든 경우에 대해서 완전한 수행능력을 보여주고 있다. 그러나 [8]의 방법에서는 외부, 확장영역에 대하여 절단영역을 계산하지 못하는 단점이 나타났다. 이러한 원인은 사각형 이미지 데이터가 화면의 출력영역보다 큰 경우와 화면 밖을 완전히 벗어났을 경우에 발생되는 것으로 분석되었다.

아래의 <표 5>는 [8]의 알고리즘과 본 연구방법의 수행성능을 비교한 것이다.

<표 5> 500,000회 수행결과 (단위 1/1000 ms)

방법 \ 영역	일반영역	외부영역	확장영역
[8]의 방법	945.4	처리못함	처리못함
본 연구방법	870.6	629.2	678.1

본 연구에서는 보다 객관적인 실험 결과를 얻기 위해 [8]의 방법에 외부영역 처리코드를 추가하였다.

일반영역에서는 본 연구의 방법이 78.4/1000 ms 만큼 성능이 좋았지만, 외부영역에서는 [8]의 방법이 18.2/1000 ms 만큼 좋게 나왔으며, 확장영역은 본 알고리즘에서 약 678.1/1000 ms 가 나왔지만, [8]의 방법에서는 실행 결과를 측정할 수 없었다<표 6>.

<표 6> 500,000회 수행결과 (단위 1/1000 ms)

방법 \ 영역	일반영역	외부영역	확장영역
[8]의 방법	945.4	611	처리못함
본 연구방법	870.6	692.2	678.1

<표 7>은 일반, 외부, 확장영역을 일괄적으로 수행한 결과이며, 전체적으로 [8]의 방법보다 본 연구방법이 약 11.85% 가 좋게 나타났다.

<표 7> 전체영역에 대한 일괄수행 (단위 1/1000 ms)

방법 \ 영역	대상영역 : 일반, 외부, 확장영역
[8]의 방법	2553
본 연구방법	2250.6

이와 같이 본 연구에서 제시한 방법과 [8]의 방법을 비교해 본 결과 본 알고리즘은 모든 경우의 데이터에 대해서 완전한 수행 결과를 보여주고 있으며, 확장영역을 처리하는데 있어서도 별도의 알고리즘을 추가하지 않아도 되기 때문에 기존의 방법보다 효율과 성능이 우수하다는 것을 확인 할 수 있었다.

5. 결론

2차원 윈도우 영역에서 사각형 이미지 데이터를 절단하는 방법은 크게 소프트웨어적인 방법과 하드웨어적인 방법으로 나눌 수 있다. 하드웨어적인 방법은 성능과 효율성이 매우 좋으며 원시코드를 작성하는데 있어서 부담을 덜어주고 있다. 그러나 소프트웨어적인 방법을 사용하게 될 경우에는 보다 효율적인 처리를 위해서 연구가 필요하게 되었다.

본 연구에서는 영역패턴을 이용하여 사각형 이미지 데이터를 절단하는 방법을 제시하고 있으며 기존의 방법보다 완전한 수행능력을 보여주었다. 또한 알고리즘을 적용하는데 있어서 단순히 비트만을 설정하여 수행능력을 향상시켰으며, 영역을 벗어나는 데이터에 대해서도 조건검색을 최소화 하였다.

따라서 본 연구는 사각형 이미지 데이터를 직접 절단하는 경우에 효과적일 것이며, 게임등과 같은 분야에서 많이 응용될 것이라 보여 진다.

향후 이미지 데이터의 효율적인 처리에 관한 다양한 연구가 지속되어야 할 것이며, 게임, 컴퓨터 그래픽스 분야에서 많은 활용가치가 있기를 기대한다.

참고문헌

[1] Andre Lamothe, TRICKS OF THE WINDOWS GAME PROGRAMMING GURUS, SAMS, 1999. pp.420-427.

[2] Cyrus, M., And J. Beck, Generalized Two and Three-Dimensional Clipping, Computers and Graphics, 3(1), 1978, pp.23-28.

[3] Y-D Liang and B.A. Barsky, A new concept and method for line clipping, ACM Transactions on Graphics Vol 3, pp.1-22, 1984.

[4] Y-D Liang and B.A. Barsky, An Analysis and Algorithm for Polygon Clipping, Communications of the ACM Vol 26, Number 11, November, pp.868-877, 1983.

[5] Blinn, J.F., And Newell, M.E. Clipping using homogeneous coordinates. In SIGGRAPH '78 Conference Proceedings(Atlanta,Ga.,Aug.1978). ACM, New York, pp.245-251, 1978.

[6] Zhang Mingjun, and Sabharwal, C. L., An Efficient Implementation Of Parametric Line And Polygon Clipping Algorithm, ACM SAC'02, pp.796-800, 2002.

[7] Maillot Patick-Gilles, A New, Fast method for 2D polygon Clipping: Analysis and Software Implementation, ACM Transactions on Graphics, vol. 11, pp.276-290, 1992.

[8] http://www.lynn3686.freeserve.co.uk/clipper.html

[9] Nicholl, T. M., Lee, D. T., and Nicholl, R. A. An Efficient New Algorithm For 2-D Line Clipping: Its Development And Analysis. In Proceedings of the ACM Computer Graphics, Siggraph '87.21, 4. ACM, pp.253-262, 1987.

[10] http://www.sronline.co.kr