이동 범위 예측을 통한 온라인 서버 성능 향상 기법※

김용오*, 신승호*, 강신진**
고려대학교 정보통신대학 컴퓨터학과*, 홍익대학교 게임학부**
koyanghi@korea.ac.kr, winstorm@korea.ac.kr, directx@korea.ac.kr

Server Performance Improvement with Predicted Range of Agent Movement

Yong O Kim*, Seung Ho Shin*, Shin Jin Kang**
Department of Computer Science and Engineering, Korea University.*
School of Games, Hongik University**

요 약

온라인 게임 시장의 규모가 커짐에 따라 온라인 게임의 서버 성능에 대한 중요성이 커지고 있다. 본 논문은 게임 속의 객체 상태에 대한 동기화 패킷을 줄여 서버의 성능을 높이는 방안을 제안한다. 제안하는 방식은 게임 속의 각 객체의 이동 범위에 대한 예측과 못가는 지역에 대한 처리를 통하여 지형간의 경계이동 횟수가 감소될 수 있도록 지형을 재구성 할 수 있는 해결책을 제공한다. 성능평가는 제안하는 지형분할방식이 기존방법에 비해 동기화 패킷에 대한 처리량이 줄어드는 것을 보여준다.

ABSTRACT

The performance of server becomes important issues in online game with the online game market expansion. This paper proposes a method of improving performance to decrease synchronized packets for each entity's informations in game. Our method provides adapted solution of reconstructing spatial subdivision to reduce a load of movement between boundary regions using prediction of entity's movement range and disabled regions where entity can not move to. It is shown through the experiments that proposed method outperforms existing method in terms of processing quantity of packets.

Keywords : Server Performance, Spatial Subdivision, Game Load Balance

접수일자 : 2010년 11월 03일 일차수정 : 2010년 11월 30일 심사완료 : 2010년 12월 08일
교신저자(Corresponding Author) : 강신진
※ 이 연구에 참여한 연구자의 일부는 '2단계 BK21사업'의 지원비를 받았음.
※ 이 논문은 2010년도 홍익대학교 학술연구진흥비에 의하여 지원되었음.

1. 서 론

온라인 게임에 접속하는 유저의 수가 증가함으로 인해 하나의 게임서버가 처리해야 할 접속자 수가 늘어난다. 많은 유저들이 게임에 대한 흥미를 잃지 않고 지속적인 플레이를 유발시키기 위해서는 게임 안에서 즐기기 위한 다양한 컨텐츠를 필요로 한다. 가장 중요한 컨텐츠 중 하나로 전투가 빠질 수 없는데, 전투에 필요한 객체의 수가 끊임없이 늘어가게 되면 게임 서버의 안정성과 성능뿐만 아니라 클라이언트의 성능에 큰 문제를 일으킬 수 있다.

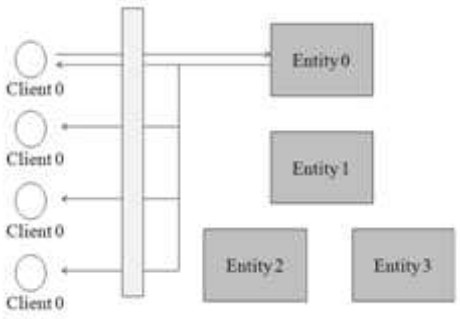
본 논문에서는 많은 수의 객체를 게임 서버에서 처리할 때 생기는 성능상의 문제를 객체의 이동 반경의 예측을 통하여 데이터를 선출하고, 그 데이터를 이용하여 경계 지역 간의 이동에서 발생하게 되는 broadcasting의 횟수를 줄일 수 있는 방안을 제안한다. 넓은 필드에 많은 수의 객체를 관리하기 위해서는 지형이 효율적으로 분할되어야 객체의 상태에 대한 참조 횟수를 줄일 수 있다. 그리고 클라이언트와 서버간의 패킷의 횟수가 감소되어야 주고받는 데이터양과 CPU사용량이 줄어들어 서버의 처리능력이 안정적으로 돌아갈 수 있다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 관련 연구를 통하여 접속하는 클라이언트의 수가 늘어남에 따라 성능이 저하되는 원인이 되는 부분을 분석하고 기존 게임에서의 지형분할의 방법에 대해서 설명한다. 3장에서는 게임 내 객체의 이동 예측 방법 및 기존 지형분할 방식을 개선한 알고리즘의 구현 방법에 대하여 설명한다. 4장에서는 제안한 방법으로 설계한 결과물의 성능분석결과를 보여준다. 마지막으로 5장에서는 결론 및 향후 연구에 대하여 언급한다.

2. 관련연구

2.1 게임에서의 패킷 처리

게임 서버와 게임 클라이언트는 서로간의 패킷 전송을 통하여 상태에 대한 동기화를 처리한다. 동기화에 사용되는 패킷은 상태나 조건에 따라 여러 종류로 구분되는데, 그 중에 가장 빈번하게 사용되는 것은 게임 내 객체의 상태에 대한 패킷이다[1]. 패킷의 수가 많아지면 많아질수록 해당 게임이 처리해야 되는 작업량이 늘어나 I/O 처리와 CPU사용량이 증가하기 때문에 서버의 전체적인 처리 능력이 떨어지게 된다. 처리능력의 저하는 게임의 질적인 면에서 직접적으로 연관되어 있기 때문에 게임 서버를 개발하는 부분에 있어 중심적인 이슈가 된다[2].



[그림 1] 서버와 클라이언트 사이의 패킷 증가

서버에 접속하는 클라이언트수가 늘어나 게임속의 객체의 수가 증가하게 되면 서버와 클라이언트 양쪽에서 처리해야 되는 패킷의 수가 늘어난다. [그림 1]에서 하나의 클라이언트가 서버에 접속하게 되면 해당 객체에 대한 생성 정보를 클라이언트에게 전달하게 되는데 다수의 클라이언트가 접속해 있으면 본인의 객체 정보뿐만 아니라 접속된 모든 객체에 대한 정보도 접속한 클라이언트에게 보내주어야 서버에 접속한 모든 객체의 정보를 알 수 있다.

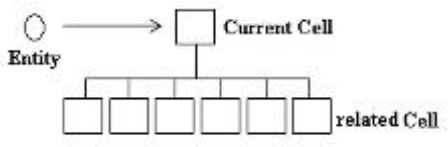
2.2 Geographic grid

온라인 게임서버에서 넓은 지형을 하나의 공간에서 처리하는 것은 대단히 비효율적이다[3]. 넓은

지형 속에 포함된 객체를 관리하기 위해 게임서버는 지형을 분할하여 관리하는 방식을 사용한다[4].

기본적으로 많이 사용되는 지형 분할은 지형데이터를 일정한 간격으로 분할하여 관리하는 방식이다. 분할하는 방식에는 quadtree[5], octree[6], 셀 분할방식[7]과 같이 여러 가지 방법이 있는데 기본적으로 구현 난이도가 높지 않고 접근이 쉬운 셀 분할 방식이 많이 사용된다.

지형분할 조건에 따라 다수로 분리된 지형의 하나의 단위를 셀이라고 부를 때, 한 셀이 관리하게 되는 정보는 해당 지형 안에 포함되어 있는 객체 정보가 기준이 된다. 셀 방식으로 지형을 나눌 때의 일정한 간격의 기준은 클라이언트에서의 시야범위가 기본이 되는데, 그 크기는 유저 PC의 이동속도에 따라 시야범위의 1에서 2배 정도로 설정된다.



[그림 2] 객체 정보의 주변 전달

분할된 지형 안에서의 객체 정보는 객체를 보유하고 있는 셀이 관리하게 되고 다른 셀 안의 객체가 해당 객체를 참조하기 위해서는 셀 데이터를 통해 접근하여야 된다. [그림 2]에서 객체의 정보는 Current Cell에 저장되어 있으며 시야 밖의 객체는 related Cell에 저장되어 있다. 서버는 해당 셀의 update를 통하여 객체 정보를 갱신하며 셀을 벗어나는 이동이나 다른 셀에서의 정보교환은 셀 간의 커뮤니케이션으로 통해 이뤄진다.

2.3 경계를 벗어나는 이동 처리

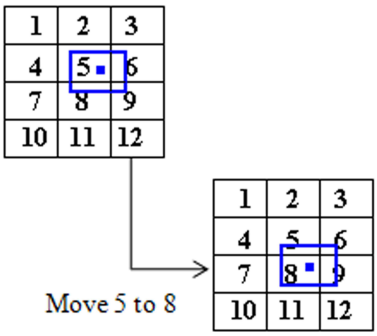
객체가 현재의 셀에서 다른 셀로 이동하게 된다면 셀 간의 커뮤니케이션을 통하여 객체 정보를 주고받는다[8,9]. 한 셀의 크기를 화면 시야만큼 분리하였을 때, 클라이언트의 입장에서는 본인이 위치한 셀과 주변 모든 셀의 정보를 알고 있어야 이동에 따른 화면 처리가 가능하다.

[그림 3]은 다른 영역으로 이동하는 객체에 대해 주변 영역의 변화를 보여준다. 패트롤이나 전투시의 이동으로 다른 셀로 이동하는 객체는 경계지역에 도달하기 까지는 주변영역에 변화가 없지만 경계지역을 통과하게 되면 해당 객체에 위치하고 있는 셀이 교체되기 때문에 주변영역의 정보도 변하게 된다[10]. [표 1]에서 5번에서의 Related 영역과 8번 Related 영역이 다르기 때문에 변경된 영역에 대해서는 객체의 정보를 전달해 주어야 한다.

[표 1] 위치에 따른 관련 영역 참조표

Current	Related Cell							
5	1	2	3	4	6	7	8	9
8	4	5	6	7	9	10	11	12

셀 간의 객체 동기화를 통하여 이전 영역에서 객체가 나가는 Pop이벤트와 새로운 영역에서 객체가 들어왔다는 Push이벤트에 대한 정보를 서버에 갱신함과 동시에 클라이언트들에게 broadcasting 한다.



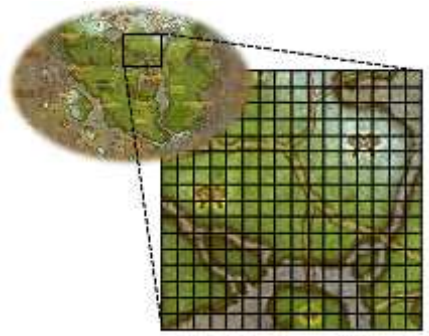
[그림 3] 경계 이동에 다른 객체 정보 전달

2.4 못가는 지역에 대한 처리

깊은 물속이나 올라 갈수 없을 정도로 높은 산과 같은 지형, 게임 상의 조건이나 기능에 따라 유저나 게임 객체가 가지 못하는 지형에 대해서 지형 분할과정에서 비 활성화된 셀로 세팅하면 메모

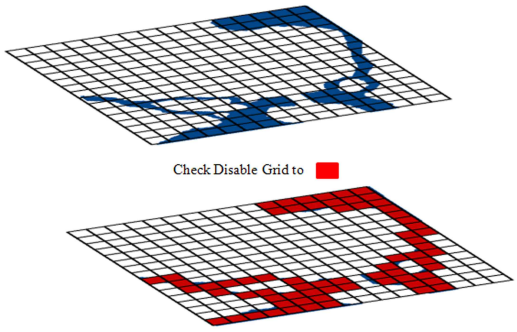
리와 셀 이동 간 작업처리를 줄일 수 있다.

[그림 4]는 전체 서버에서 한 영역을 선택했을 때의 모습을 그림으로 보여준다. 전체 지도는 하나 이상의 영역으로 나누어지고 해당 영역은 다시 Grid로 재분할된다. [그림 4]의 아랫부분은 선택된 영역을 Grid로 표현한 그림이다.



[그림 4] 게임 지형에서 선택된 영역

[그림 5]는 같이 하나의 지형을 기본적인 셀 규격에 맞춰 분할한 뒤 추가되는 정보를 통해 지형에 대한 정보를 재구성한 모습이다. [그림 5]의 아래 그림에서 붉은 사각형은 못가는 지역으로 확인된 영역으로서, 이런 지역을 검색의 범위에서 제외시키면 해당 영역에 대한 불필요한 메모리가 줄고 검색의 속도도 높아진다[11]. 이러한 데이터는 맵이 생성되는 시기에 같이 생성되기 때문에 실행시간에는 지형정보 제작에 따른 계산이 필요하지 않는다.



[그림 5] 못 가는 지역에 대한 처리

3. 구 현

본 논문은 기존 연구과제에서 제시되었던 이동불가 지역에 대한 지형분할 결과를 발전시켜, 객체의 이동범위를 예측한 결과 table로 재구성 하여 영역간의 경계를 벗어나는 객체의 수를 최소화 할 수 있는 지형 분할 기법을 제안한다. 셀 사이를 오고 갈 때 생기는 부하를 줄이기 위하여 지형에 생성된 객체의 정보를 통해 이동범위와 경로를 예측하고 그 데이터를 통하여 재구성된 새로운 Geographic grid를 생성한다.

3.1 이동불가 지역에 대한 처리

못가는 지역에 대한 참조 데이터를 서버 실행시간이 아닌 boot-up 시간에 생성하여 선 처리 데이터로 사용한다. Lui's 연구 결과 [12] 에서의 선처리된 데이터의 처리방식을 통하여 기존 지형분할 알고리즘으로 제작된 데이터와 지형데이터를 링크시킨다.

Checked disable - geogrid

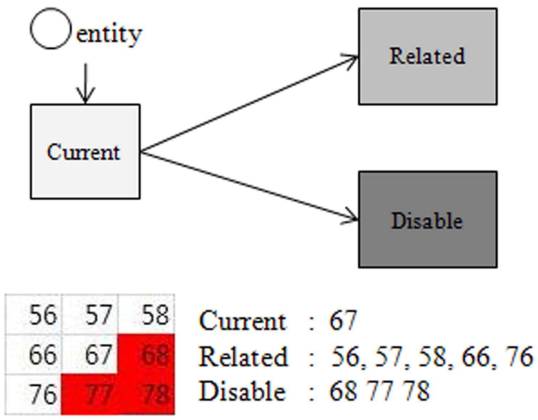
0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

[그림 6] 지형분할 그래프

실제 지형을 통해 생성된 지형데이터는 접근의 속도를 높이기 위해 위치 정보의 링크만 저장하며 데이터는 게임 서버안의 객체를 관리한다. [그림 6]은 기본 지형을 배열로 나눈 table에 대하여 이동불가 지역에 대한 선 처리 결과를 적용시킨 데이터에 대한 시각적인 모습이다. 각 셀 안의 정보는

실제 데이터에 대한 링크를 가지고 있다. 이동불가 지역에 대해 선 처리된 데이터는 기존 지형데이터와 마찬가지로 배열로 생성되고 지형 데이터의 메모리에 대한 key를 배열의 index를 통하여 참조할 수 있게 링크한다. 게임의 한 객체가 지역 간의 경계를 이동하게 되면 선 처리 데이터의 링크를 통하여 주변영역에 대한 broadcasting를 처리할 때 이동불가 지역은 자동적으로 해당 영역에서 제외된다.

[그림 7]에서 객체가 57번 셀에서 67번 셀로 경계를 넘어가는 이동시 셀의 경계를 통과하게 되는데, 67번 셀의 주변 영역 중 못가는 지역으로의 정보 전달이 필요 없다는 것을 선 처리데이터의 링크를 통해 파악할 수 있다.



[그림 7] 링크된 정보의 접근

3.2 이동반경 예측을 통한 MergeGrid 생성

게임 상에 생성되는 객체는 지형의 특성이나 의도에 맞춰 화산지형에는 화산지형에 맞는 객체가 호출되고 늪 지형에는 늪 지형에 적합한 객체가 호출되도록 기획된다. 각 객체는 서버가 시작될 때 이미 스폰 위치에 대한 정보를 가지고 있다. 스폰 위치는 점 좌표가 되기도 하고 반경 범위가 되기도 하는데, 일정범위를 넘기지 않는 범위 내부에 호출되어야 기획적인 의도와 일치된다. 생성된 객체는 타입에 따라 근거리 순찰형, 지역 방어형, 비선공형등 여러 가지 타입으로 분류되며 적의 탐색에 사용되는 시야거리도 타입에 맞춰 다르게 세팅되지만 일정 범위를 넘어 다른 지형으로까지 넘어가지 않도록 하기 위해 기획적 순찰범위를 한정시켜 놓는다.

본 논문에서는 각 객체의 스폰 위치와 패트롤 범위, 시야 거리에 따른 한 객체의 행동반경을 이동반경을 예측하기 위한 데이터로 사용한다. 이동반경 데이터를 통하여 객체의 행동에 따라 주변 셀과의 밀집도를 계산한다. 경계의 라인을 통과한 수의 최소값을 $MinL$이라 두고, 주변 셀과의 밀집도의 최대값을 $MaxD$라고 둘 때 아래와 같은 밀집도 D가 산출되게 된다.

$$D = \max\left(\min\left(\frac{\sum_{t=0}^{N} Lt}{N_{npc}},\ MinL\right),\ MaxD\right)$$

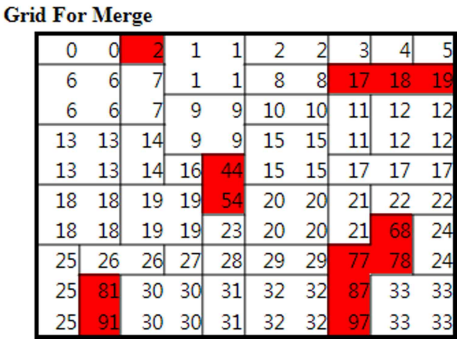
L_t = 시간에 라인을 통과한 NPC수

여기에서 전체 NPC에 대하여 해당 라인을 통과한 수를 $MinL$과 비교한다. 두 값 중의 최소값이 MaxD보다 클 경우에 해당 셀의 밀집도 값인 D를 계산하여 저장한다. 전체 셀에 대해, 각 셀의 밀집도가 다음 셀에 대한 밀집도 값보다 높다고 계산되면, 두 셀들에 대한 밀집도에 대한 flag값인 Df를 true로 세팅하고 각각의 index값을 반환한다.

$$Df = \begin{cases} true & \text{if } (D \geq theshold) \\ false & else \end{cases}$$

[그림 8]은 선 처리된 데이터와 이동반경 데이터를 이용하여 만들어진 Merge 테이블을 배열로 표현한 그림이다. 셀 사이에 Related Cell관계가 이뤄지면 같은 상자로 묶는다. 같은 상자로 묶여진 셀은 서로간의 경계가 없어지기 때문에 셀 간의 경계이동에 대한 broadcasting처리가 불필요하다. 데이터의 정렬과 메모리의 효율적인 사용을 위해 Related Cell 끼리 하나의 index로 연결하는 작업

이 이뤄진다.



[그림 8] Merge된 지형데이터

연결처리가 된 첫 번째 셀 번호를 기본 index로 설정하고 값을 순차적으로 증가시킨다. index가 설정된 cell은 Related Cell의 데이터를 참조하여 관련 셀을 찾아내서 같은 index를 부여하도록 한다.

[표 2] pseudocode for generating cell

```
index = getNextIndex();
do
    if  currentCellindex is not setted
        set currentCellindex = index
while
(null!=GetNextRelated(currentCell info));
```

[표 2]는 각 셀에 대하여 Related Cell에 대한 index를 부여하는 알고리즘을 의사코드로 작성한 내용이다. [그림 8]의 사각으로 둘러싸여 같은 index를 가진 셀은 이동반경 데이터를 통해 밀집도가 thredhold값 이상으로 계산되어 Related Cell 처리된 것이다.

[표 3]은 Merge 된 Cell위의 객체가 현재 셀에서 다른 셀로 이동처리 과정에 대한 알고리즘의 의사코드다. 이동함수가 호출될 때 현재의 셀이 변경되었다면 객체정보를 해당 셀에게 전달하도록 PushEntity와 PopEntity 함수를 호출한다. 함수 내부에서는 각 셀의 가상함수로 제작된 broadcast가 호출되어 주변 셀에게 객체의 정보를 전송한다.

[표 3] pseudocode for entity broadcasting

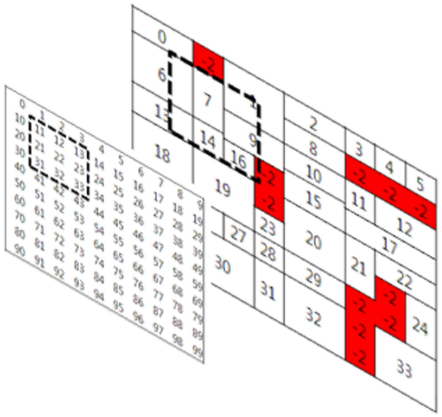
```
class ICell
{
virtual void Broadcast()=0;
virtual void Transfer()=0;
virtual void PushEntity()=0;
virtual void PopEntity()=0;
};
class IMergeCell : public ICell ;
class INoramlCell : public ICell ;

EntityUpdate(time_t time, entity)
{
 ICell *pPrev = entity->GetCell();
 entity->EntityMove(time);
 ICell *pCell =entity->GetCell();
 if pPrev is pCell then
     return;
 else
     pCell->Transfer( pPrev, entity );

pCell->PushEntity(entity);//bracdcast
pPrev->PopEntity(entity);//bracdcast
}
```

3.3 결과 Grid의 생성

선처리 데이터와 이동반경 데이터를 통하여 생성된 Merge Grid와 기존 지형분할을 통해 만들어진 Grid를 연결하여 계층적 구조의 테이블을 작성한다. 각 객체는 기존 지형분할 Grid로 다른 데이터에 접근하는 것과 동일한 interface를 사용하여 계층적 구조로 만들어지기 때문에 두 가지 데이터를 원하는 시기에 적용 시킬 수 있다.



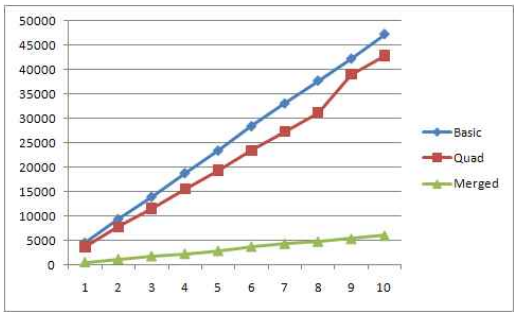
[그림 9] GeoGird의 계층적 그래프

[그림 9]의 위에 올려진 그림은 기본 지형분할 데이터고 바닥에 깔린 그림은 선 처리 데이터와 이동 반경 데이터로 생성된 Grid다. 기본 지형의 12번 cell에서 22번 cell로 이동할 때, 두 cell의 상위값의 index가 7번으로 동일한 부모 값을 가지기 때문에 broadcasting이 발생하지 않는다. 11번 cell에서 12번 cell로 이동하게 되면 부모의 index가 6과 7로 다르기 때문에 broadcasting 작업이 수행된다.

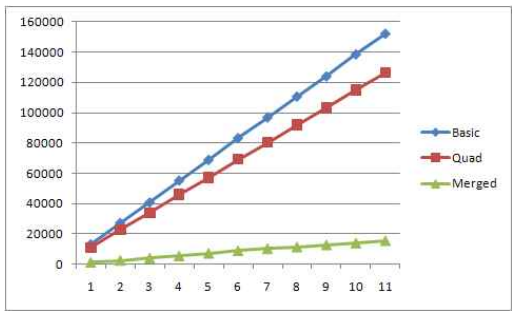
4. 성능 및 실험

본 시스템은 Windows XP 운영 체제, Intel Core2 CPU 2.40GH, 2GB 메모리의 환경에서 C++와 Visual Studio 2008을 사용하여 구현하였다. 기존의 지형분할 방식으로 제작된 서버와 3장의 구현 방법에 따른 이동 예측으로 생성한 지형분할 방식에 따라 제작한 게임서버의 성능을 비교하였다. 기본 셀의 최대 수는 100으로 제한하였으며 한 셀의 규격은 가로, 세로 1m로 설정하였다. 셀 분할방식과 QuadTree방식, 제안한 Merged방식을 비교하여 실험하였으며 Octree는 높이 값이 0일때 QuadTree와 결과 값이 동일하기 때문에 측정에서 제외하였다.

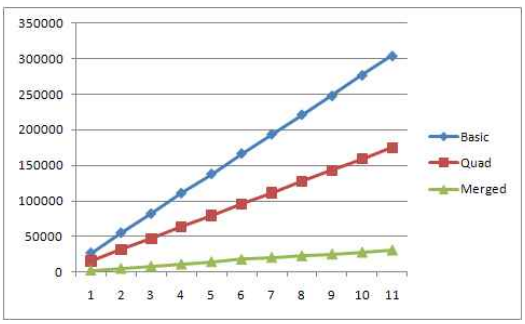
[그림 10]은 객체의 수가 작을 때 기본 지형 분할과 제안한 방식을 측정한 결과 값이다. 두 데이터의 결과 값으로 미루어 볼 때 제안한 방식이 성능 면에서 많은 효과가 있는 것을 알 수 있다. [그림 11]과 [그림 12]는 객체의 수를 다르게 테스트한 결과를 보여준다. [그림 10]과 마찬가지로 기존 방법에 비해 정보교환 패킷의 수가 줄어드는 것을 확인할 수 있다.



[그림 10] case 1 (객체 수 : 하)



[그림 11] case 2 (객체 수 : 중)



[그림 12] case 3 (객체 수 : 상)

[표 4] 결과 테이블

	Basic	Quad	Merge
Case1	45,777	42,874	4,099
Case2	162,165	126,272	6,328
Case3	304,330	174,936	14,322

[표 4]는 모든 테스트에 대한 결과를 정리한 모습이다. 표 안의 숫자는 각 case에서의 broadcast의 수를 합친 결과이다. 제안된 방식이 셀 분할방식과 QuadTree의 결과 값과 비교해서 높은 성능을 보이는 것을 확인할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 객체의 이동 범위 예측을 통하여 서버의 성능을 향상시킬 수 있는 알고리즘을 제안하였다. 본 연구 결과는 기존 방식에 비하여 제시한 방식이 성능 면에서 높은 성능적인 차이를 나타냈다. NPC뿐만 아니라 PC의 이동을 데이터마이닝을 통하여 예측하는 알고리즘을 향후 연구할 예정이다.

참고문헌

[1] Abdelkhalek A and Bilas A, “Parallelization and performance of interactive multiplayer game servers”, IPDPS, pp72, 2004.

[2] 문성원, 조형제, “온라인 게임의 서버 메시지 동기화 분산에 대한 연구”, 한국 게임학회 논문지, 제 9권, 제 2호, 2009.

[3] Fengyun Lu, et al., “Load Balancing for Massively Multiplayer Online Games”, Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, 2006.

[4] Roger Smith, Modelbenders, LLC, et al. “Geographic Grid Registration of Game Objects”, Game Programming Gems 6, Charles River Media, 2006.

[5] Helge Backhaus, Stephan Krause, “QuON: a quad-tree-based overlay protocol for distributed virtual worlds”, International Journal of Advanced Media and Communication, 2010.

[6] J. Revelles, C. Urena, and M. Lastra, “An Efficient Parametric Algorithm for Octree Traversal”, WSCG'2000 conference, pp212-219, 2000.

[7] G. Huang, M. Ye, and L. Cheng, “Modeling system performance in MMORPG”, GlobeCom, pp512-518, 2004.

[8] G. singh, et al., “BrickNet: Sharing object behaviors on the net”, In Proc. of the IEEE Virtual Reality Annual International Symposium, pp19-25. Los Alamitos CA, IEEE Computer Society Press, 1995.

[9] J. Chim, R. Lau, H.V. Leong And antonio Si, “CyberWalk:A Web-based Distributed Virtual Walkthrough Environments”, PhD Thesis, Newcastle University, 2006.

[10] B. Hariri, S. Shirmohammadi, M.R. Pakravan, “LOADER: A Location-Aware Distributed Virtual Environment Architecture”, IEEE Conference on Virtual Environments, July 2008.

[11] 장수민, 유재수, “효율적인 MMORPG 분산 게임 서버”, 한국 콘텐츠 학회 논문지, Vol.7, No.1, pp32-39, 2007.

[12] J. C. S. Lui and M. F. Chan, “An efficient partitioning algorithm for distributed virtual environment”, IEEE Transactions on systems Parallel and Distributed Systems, Vol.13, pp193-211, 2002.

김 용 오 (Kim, Yong O)

2005 고려대학교 정보통신대학 미디어공학과 공학석사

관심분야 : 컴퓨터 프로그래밍, 네트워크

신 승 호 (Shin, Seung Ho)

2007 고려대학교 정보통신대학 컴퓨터학 이학석사
2011 고려대학교 정보통신대학 컴퓨터전파통신학과 공학박사

관심분야 : 컴퓨터 프로그래밍, 컴퓨터 그래픽스, 가상 현실

강 신 진 (Kang, Shin Jin)

2003 고려대학교 정보통신대학 컴퓨터학과 이학석사
2011 고려대학교 정보통신대학 컴퓨터학과 이학박사
2003-2006 소니컴퓨터엔터테인먼트코리아 (Sony Computer Entertainment Korea)
2006-2008 엔씨소프트(NCsoft)
2008-현재 홍익대학교 게임학부 전임강사

관심분야 : 게임 기획, 컴퓨터 그래픽스