레터를 이용한 네트워크
(Networks With Letters)

John Knottenbelt

2009. 8. 12

(http://www.gamasutra.com/view/feature/4052/networks_with_letters.php)

[본 기술자료에서는 Introversion의 Knottenbelt가 새로 출시될 Xbox Live Arcade용 게임인 Darwinia+의 멀티플레이어용 불연속 이벤트 시뮬레이션(discrete event simulation)에 대하여 논의한다. 이 방식은 수천 개의 게임 내 객체 위치가 아닌 플레이어의 동작과 버튼이 네트워크로 전달되는 방식이다.]

소개

많은 이들이 잘 모르고 있지만 IGF상을 수상한 Introversion의 Darwinia는 미완성으로 출시되었다. Darwinia는 원래 멀티플레이어 게임으로 계획된 게임이다.

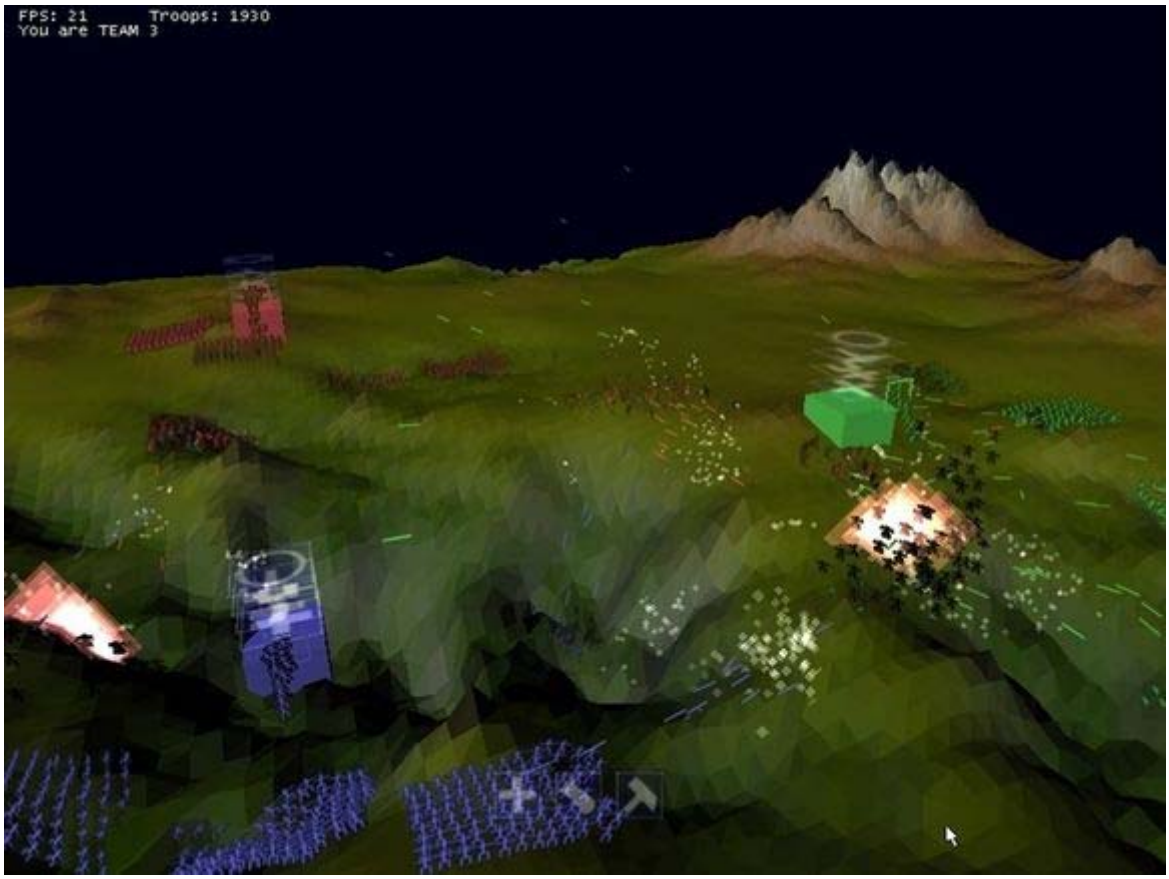
이 게임의 최초 코드네임은 “Future War”이었으며 수많은 귀신 부대를 거느린 플레이어가 또 다른 부대를 지휘하는 다른 플레이어와 격돌하는 대규모 전투로 설계되었다. 하지만 자금이 떨어지는 문제가 발생하였다.

Introversion의 재정이 줄어들게 됨으로써 최초의 멀티플레이어 요소는 Darwinia에서 사라졌다. 하지만 몇 개월이 지나 IGF 상을 수상하는 예상치 못한 영광으로 우리는 다시 멀티플레이어 개념을 부활시킬 수 있게 되었다.

마이크로소프트는 Darwinia를 멀티플레이어 기능과 함께 XBLA로 발표하기를 원하였다. 이는 2008년 말 PC로 출시한 새로운 게임인 Multiwinia IV를 개발할 때였다. Darwinia와 함께 Multiwinia는 올 여름 Live Arcade에서 Darwinia+로 출시될 것이며 이는 처음으로 콘솔 게임을 출시하는 Introversion에게 중대한 사건이 될 것이다.

Darwinia+ 작업을 시작하던 시기에 멀티플레이어 네트워킹은 우리에게 새로운 것은 아니었

다. 언급한 바와 같이 우리는 처음 Darwinia를 개발할 당시 멀티플레이어 게임에 살짝 발을 담근 적이 있으며 성공을 거두지는 못했지만 소중한 교훈을 얻을 수 있었다. 그것은 바로 멀티플레이어기능을 초기에 게임의 핵심으로 통합하는 것이 좋다는 것이다.



우리가 Darwinia와 함께 시작했던 네트워킹은 2006년 Defcon과 작년에 발매된 Multiwinia 등 멀티플레이어 게임에서 유용한 것으로 입증되었다.

우리의 필자가 본문에서 설명할 멀티플레이어 네트워킹 방식은 장단점과 함께 핵심적인 프로그래밍 문제를 해결하기 위한 통찰력을 가지고 있다.

기존의 접근 방식

게임에서 가장 일반적인 멀티플레이어 네트워킹 접근 방식은 여러 플레이어가 연결하는 게임 서버를 마련하는 것이다.

게임 서버는 게임 세계를 시뮬레이션하고 플레이어들이 연결된 세계 내에서 객체들의 위치와 속도를 전파한다.

이 시스템이 제대로 작동하려면 다음 업데이트를 기다리는 동안 클라이언트가 객체의 위치와 속도를 예측해야 한다.

하지만 Defcon와 Multiwinia에서는 수 많은 객체들의 위치와 속도가 변화하기 때문에 위의 방식을 따를 경우 영국의 표준 광대역 연결보다 더 우수한 광대역을 필요로 한다.

Multiwinia의 예를 들면 게임 내에 각각의 속도와 위치가 변화하는 5000명의 Darwinian이 있을 경우 대략의 업데이트 규모는 다음과 같이 계산할 수 있다.

5000 Darwinians * SizeOf (
New Position (Represented as 3 32-bit floating point numbers - X, Y, Z)
New Velocity (Represented as 3 32-bit floating point numbers - DX, DY, DZ)
New State (byte)
)
= 5000 * (4*3 + 4*3 + 1) = 5000 * 25 = 125000 bytes

플레이어들은 서버와 상호교신을 필요로 하며 초당 10회 업데이트를 할 경우 우리는 다음과 같은 용량을 필요로 하게 된다.

125000 bytes * 10/s = 1 250 000 bytes/s

그리고 Multiwinia에는 4명의 플레이어가 접속하므로 실제로 다음과 같이 예상할 수 있다.

3 * 1 250 000 = 3 750 000 bytes/s 업스트림 대역(업로드 속도)

물론 압축 기술을 이용하여 이를 약간 최적화할 수는 있지만 참으로 방대한 수치임은 분명하다. 이와 같은 방식으로 세계의 상태를 전파하는 것은 Multiwinia의 게임 세계에는 적합하지 않다.

불연속 이벤트 시뮬레이션(discrete event simulation)

우리가 선택한 방식은 불연속 이벤트 시뮬레이션(discrete event simulation)의 일종이다. 이 방식의 개념은 플레이어의 입력이 없으면 Defcon과 Multiwinia의 세계는 동일한 시작 조건에서 같은 방식으로 진행하는 것이다.

플레이어의 입력이 세계의 진행에 영향을 미치도록 함으로써 모든 시뮬레이션은 서로 보조

를 맞추도록 한다. 이론적으로 우리는 플레이어의 입력(마우스 클릭과 키 입력)을 서버로 보내 모든 클라이언트로 복사하도록 해야 한다.

하지만 사용자 입력의 많은 부분이 세계에 영향을 미치지 않는다(예를 들어 정찰을 위한 비행). 따라서 우리는 세계에 영향을 미치는 행동을 추상화를 이용하여 나타내기로 결정하였다.

간단하게 표현한 게임 루프를 살펴보도록 하자.

```
MainGameLoop()
{
while( !GameOver() )
{
ProcessInput();
ProcessEvents();
Render();
}
}
```

ProcessInput()은 사용자의 입력을 담당한다. 세계의 상태를 직접 변경시키는 대신 우리는 이벤트를 생성하여 서버로 보낸다. 서버는 10Hz로 구동된다. 따라서 서버는 100밀리세컨드마다 플레이어로부터 받은 모든 이벤트를 취합하여 '네트워크 레터(network letter)'로 포장한다.

네트워크 레터는 연결된 각 플레이어에게 전송된다. 서버는 각각의 레터에 순번을 지정하여 클라이언트들이 이벤트를 순서대로 평가할 수 있도록 한다. 예를 들어 클라이언트가 Darwinian그룹을 특정 위치로 이동시키도록 명령을 하면 이는 이벤트를 통해 인코딩된다.

```
MOVE_DARWINIANS
  TEAM_ID: unsigned int
  GROUP_POS: Vector3
  GROUP_RADIUS: unsigned int
  DEST_POS: Vector3
```

TEAM_ID는 플레이어 팀의 식별자(identifier)이다. 연결된 모든 플레이어가 이벤트를 처리하기 때문에 팀을 구별하는 것은 중요하다. GROUP_POS는 세계 내에서의 위치를 나타내며

GROUP_RADIUS는 선택 반경의 크기를 나타낸다(모든 Darwinian들은 GROUP_POS에서 지정한 원 안에 위치하며 GROUP-RADIUS가 선택된다). DEST-POS는 Darwinian들이 향하는 곳을 의미한다.

ProcessEvents()는 서버로부터 네트워크 레터를 수신하고 서버에서 정한 순서에 따라 처리한다.

100ms 동안 이벤트를 수령하지 않으면 서버는 이벤트가 포함되지 않은 빈 네트워크 레터를 보낸다. 이는 유효한 표시 이벤트로써 클라이언트가 사용자의 입력 없이도 세계와 게임 시간을 진행할 수 있도록 한다.

장점

이 방식의 장점은 시스템에 필요한 대역을 크게 줄일 수 있다는 점이다. Defcon과 Multiwinia에서 우리는 클라이언트 당 2KB/s만을 필요로 한다.

따라서 약 25KB/s인 영국 업로드 속도에서 서버는 최대 12대의 클라이언트를 지원할 수 있으며 Defcon의 6인 게임과 Multiwinia의 4인 게임을 지원하는데 충분하다. 실제로 Defcon에서 우리는 클라이언트를 추가하여 게임을 관람할 수 있도록 하는 관람 모드를 구현하였다(물론 게임 조작은 할 수 없다!).

이 방식의 또 다른 장점은 이론 상 매우 간단하게 게임을 기록하고 재생한다는 점이다. 이 방식에서는 이벤트의 내역을 저장하고 클라이언트에서 다시 재생을 하기만 하면 된다.

이 아이디어를 조금 확장하면 일단 이벤트를 기록한 후 다른 카메라 위치와 움직임을 시도함으로써 품질이 매우 우수한 영상을 만들어낼 수 있다. 그 다음 두 차례 재생하여 한 번은 소리를 기록하고 프레임 별로 고해상도 영상을 기록할 수 있다.

단점

이 멀티플레이어 네트워킹 방식의 단점 중 한 가지는 게임이 진행 중일 때 들어올 플레이어는 지금까지의 전체 게임 이벤트 내역을 전송 받은 다음 이벤트를 평가해야 한다는 점이다.

게임이 오랫동안 진행된 경우 전송될 데이터의 양도 상당할 뿐만 아니라 이벤트를 평가하는 시간도 오래 걸리게 된다.

이 문제의 해결책은 서버에 서버에 최근 세계 상태를 기록하여 다음 이벤트와 함께 전송함으로써 대역을 아끼는 것이다. 이 방식은 비디오 코덱에 이용되었던 i-프레임 시스템을 떠

오르게 한다.

다른 단점으로는 게임코드를 주의 깊게 작성하여 모든 클라이언트들에게 정확히 같은 방식으로 이벤트 평가가 이루어지도록 해야 한다는 점이 있다. 따라서 게임 후반 업데이트에서 이벤트의 평가가 변경된 경우(예를 들어 버그의 수정) 클라이언트들이 함께 게임을 하기 위해서는 모두 함께 새 버전으로 업그레이드를 해야 한다.

Defcon과 Multiwinia는 이벤트 평가에서 부동소수점 연산을 이용한다. 따라서 대부분의 클라이언트가 IEEE-754에 따르는 반면 이와 다른 컴파일러를 이용하여 다른 방식으로 컴파일을 할 경우 약간 다른 부동소수점 연산 결과를 나타내는 문제도 있다.

이는 부동소수점 오류가 수학적 연산 순서에 따라 발생하며 컴파일러들이 각기 다른 방식으로 최적화를 하기 때문에 일어난다. 또한 PowerPC와 인텔 프로세서 사이의 상호운용에 관련된 문제도 존재한다.

결론

우리가 Defcon과 Multiwinia에서 취한 네트워킹 방법은 우리에게는 잘 작동하였다. 레터 시스템은 확장성을 입증하였으며 복수의 게임 유형에 적용될 수 있다.

전반적인 시뮬레이션 방법론은 예측 결과에 대한 의존으로 인하여 문제를 나타냈으며 우리는 상당한 시간을 들여 동기화 문제를 해결해야 했으나 최종 결과는 문제점을 해결하고도 남았다.

우리가 만일 이 게임을 다시 작업한다면 부동소수점 연산에 대한 의존을 어떻게 피하고 중간 참여 준비시간을 줄일 수 있는 '빠른 따라잡기' 기능을 어떻게 구현할 수 있을지를 생각해볼 것이다.