Dead Reckoning 기법을 활용한 네트워크 게임의 동기화 효과 분석

현광빈, 이환용
아주대학교

bio8641@ajou.ac.kr, hwan@ajou.ac.kr

Analysis of synchronization effects of network games using the Dead Reckoning technique

Kwangbin Hyun, Hwanyong Lee
Ajou University

요 약

온라인 게임에서는 사용자들 간에 다양한 상호작용이 일어난다. 어떤 사용자가 특정 행동을 하였을 때 이에 대한 다른 사용자의 화면에서의 동기화가 필요한데, 그 중 사용자의 이동에 대한 동기화는 사용자의 행동을 일일이 통신하면 통신량이 너무 많아지고, 이로 인한 지연시간의 차이로 인해 유저들 간의 동기화가 원활하지 않게 되기 때문에 단순히 처리하기 까다로운 문제이다. 이러한 문제를 완화하기 위한 방법이 Dead Reckoning 기법이다. Dead Reckoning 은 물체의 현재 위치 정보를 기반으로 미래의 위치를 추측하는 방법이다. 본 논문에서는 시뮬레이션을 진행할 간단한 게임을 만들고, Dead Reckoning 기법을 활용하여 다양한 환경에서 테스트해보면서 이동 동기화 기법의 효과를 분석해볼 것이다.

Ⅰ. 서 론

온라인 게임에서는 사용자들 간에 다양한 상호작용이 일어난다. 어떤 사용자가 특정 행동을 하였을 때 이에 대한 다른 사용자의 화면에서의 동기화가 필요한데, 그 중 사용자의 이동에 대한 동기화는 사용자의 행동을 일일이 통신하면 통신량이 너무 많아지고, 이로 인한 지연시간의 차이로 인해 유저들 간의 동기화가 원활하지 않게 되기 때문에 단순히 처리하기 까다로운 문제이다. 이러한 문제를 완화하기 위한 방법이 Dead Reckoning 기법이다. Dead Reckoning 은 물체의 현재 위치 정보를 기반으로 미래의 위치를 추측하는 방법이다. 본논문에서는 시뮬레이션을 진행할 간단한 게임을 만들고, Dead Reckoning 기법을 활용하여 다양한 환경에서 테스트해보면서 이동 동기화 기법의 효과를 분석해볼 것이다.

Ⅱ. 본 론

2.1. 실험 내용

- 게임 서버 환경 구축

본논문에서 사용할 게임 서버를 만들기 위해 C#의 .NET 프레임워크를 사용하였다. 반응성이 중요한 게임서버 특성상 소켓 입출력을 동시에 빠르게 처리해야 하므로 Asynchronous Multi-Thread 방식을 채택하였다.

- Dummy Client 개발

동시의 다수의 클라이언트를 접속하고 이동하기 위해 C#을 통해 더미 클라이언트를 개발하였다. 해당 클라이언트에서는 더미 클라이언트의 수를 정할 수 있고, Poisson Distribution 을 통해 Random 한 움직임을 만들어 더미 클라이언트의 이동 방식을 결정하는 Event Timeline 을 입력 받아 클라이언트들의 움직임을 실제로 Simulate 할 수 있다.

- Unity Client 개발

그림 1 과 같이 클라이언트들의 플레이어의 움직임을 실제로 확인하고 이동 경로의 결과를 얻기 위해 Unity3D 를 사용하여 클라이언트를 개발하였다. 해당 클라이언트에서 자신의 플레이어 및 다른 클라이언트들의 플레이어의 움직임을 확인할 수 있고, 더미(Dummy) 플레이어의 이동 Event Timeline 을 딜레이 없이 수행하는 Ghost 가 있어 동기화가 잘 이루어졌는지 확인해 볼 수 있다.

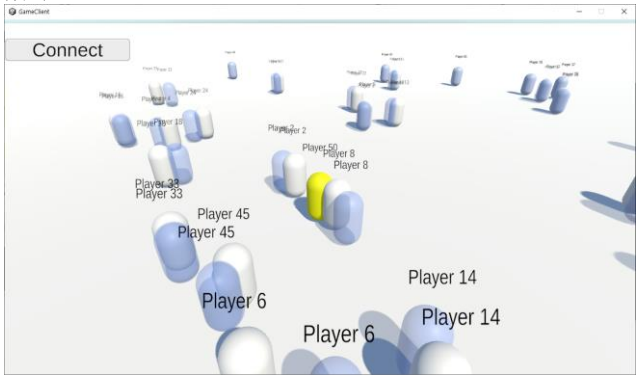


그림 1 Unity Client 화면

- 동기화 테스트

이동 동기화를 테스트하기 위해 서로 다른 동기화 기법을 사용하는 Protocol 을 준비하고 이를 동일한 환경에서 테스트하여 결과를 비교 및 분석하였다. 사용된 프로토콜은 아래의 3 가지이다.

Move_V1 은 단순히 플레이어의 위치만 전달하며, 패킷을 전달받으면 클라이언트는 해당 플레이어를 패킷의 위치로 순간 이동시킨다.

Move_V2 는 플레이어의 속도 정보만들 전달하며, 패킷을 전달받으면 클라이언트를 해당 플레이어의 속도를 패킷의 속도로 설정하여 동기화를 진행한다.

Move_V3 는 기본적으로 속도 정보만 전달받으나, 4 번째 패킷에는 위치 정보를 추가로 전달받아 다음 패킷을 받기 전에 해당 위치로 이동할 수 있도록 속도를 설정하여 동기화를 진행한다.

2.2. 실험 결과

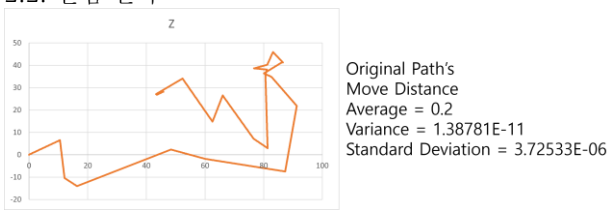
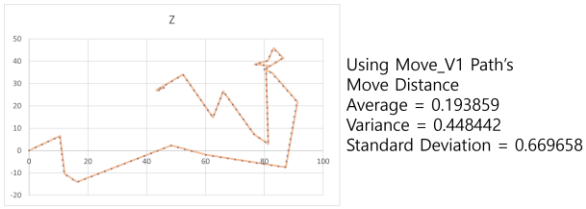
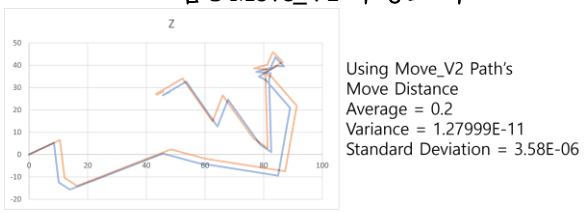


그림 2 플레이어의 원본 경로



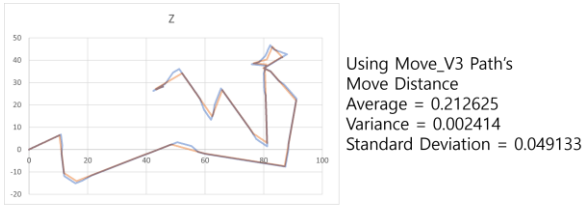
Average Distance with Original = 1.005526

그림 3 Move_V1 의 경로 비교



Average Distance with Original = 3.031399

그림 4 Move_V2 의 경로 비교



Average Distance with Original = 0.713291

그림 5 Move_V3 의 경로 비교

- 동기화 기법에 따른 이동 경로 분석 결과

그림 3, 4, 5 는 동시 접속자가 50 명인 게임에서 클라이언트마다 1 초에 4 번의 위치 정보 패킷을 전송하였을 때, 임의의 클라이언트(Player 1)의 이동 경로 분석 결과이다. 그래프 우측의 통계 결과는 각 프레임에서 이전 위치와의 거리를 통계로 사용하였고, 그래프 하단의 통계 결과는 그림 2 의 원본 경로(Original Path)와 비교하여 평균 거리를 나타내고 있다.

실험 결과 원래의 이동 경로와의 평균 거리는 Move_V1 과 Move_V3 는 각각 1.005526, 0.713291 로 크게 차이가 없었지만 Move_V2 는 3.031399 로 큰 차이를 보였다. 위치를 보정하는 기능이 원래의 이동 경로와의 거리에 영향을 크게 미치는 것으로 확인되었다.

이전 프레임과의 평균 거리는 모두 약 0.2 로 비슷했지만 표준 편차에서 Move_V2 와 Move_V3 는 각각 3.58E-06, 0.049133 으로 매우 작았지만 Move_V1 에서 0.669658 로 상대적으로 높게 나왔다.

Ⅲ. 결 론

본논문에서는 몇 가지 이동 동기화 기법을 정의하고, 50 명의 클라이언트가 접속한 환경에서의 각 동기화 기법의 이동 경로를 원본 이동 경로와 비교하여 동기화 성능을 분석해보았다. 원래의 이동 경로와의 평균 거리는 Move_V1 과 Move_V3 가 비슷하게 작았으며 Move_V2 는 상대적으로 크게 나왔다. 이동 경로의 평균 간격은 모두 비슷했으나, 표준편차에서 Move_V2 가 매우 작았고, Move_V3 도 작았지만 Move_V1 이 상대적으로 크게 나왔다. 원래의 이동 경로를 잘 따라가는 것뿐만이 아니라 부드럽게 이동해야 하는 이동 동기화 특성상 해당 표준편차는 작게 나와야 하므로 Move_V1 은 이동 동기화 방법으로써 적절하지 않다. 만약 V1 의 방법을 사용하려면 게임의 프레임 수만큼 이동 패킷을 전송해야 하는데, 이는 매우 높은 전송량을 요구하므로 비효율적이다.

따라서, Dead Reckoning 을 통해 다른 클라이언트의 부드러운 움직임을 상대적으로 적은 패킷 전송량으로 구현할 수 있다는 것을 알 수 있다. 이는 접속한 클라이언트 수가 많은 네트워크 게임 환경에서 네트워크 통신량을 크게 줄일 수 있을 것으로 예상된다.

ACKNOWLEDGMENT

본 연구는 2024 년 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학사업의 연구결과로 수행되었음(2022-0-01077)

참 고 문 헌

[1] J. Aronson. “Dead Reckoning: Latency Hiding for Networked Games” 1997, (https://www.gamedeveloper.com/programming/dead-reckoning-latency-hiding-for-networked-games).

[2] C. Haag. “Targeting - A variation of dead reckoning” 2001, (https://www.gamedev.net/reference/articles/article1370.asp).