모바일 게임 개발자를 위한 유니티 컴포넌트 개발

이현수, 조경호, 신도현, 진영택
한밭대학교 컴퓨터공학과
leehs9145@gmail.com, dostmoth@gmail.com
shindohyun00@gmail.com, ytjin@hanbat.ac.kr

Unity Components Development for Mobile Game Developers

Hyun-Su Lee, Gyoung-Ho Jo, Do-Hyun Shin, Young-Taek Jin
Dept. of Computer Engineering, HanBat University

요 약

멀티플레이 게임을 개발하기 위해서는 게임 서버의 구축이 필수 요소이며, 이러한 멀티 플레이 게임은 데이터 송수신이 빨라야 하므로 두 디바이스 간의 직접적인 통신을 위한 홀 펀칭, 릴레이 서버와 같은 기술이 요구된다. 그러나 멀티플레이 게임의 구현을 위해서는 일반적인 서버보다 경제적, 시간적인 자원이 더 많이 소요되기 때문에 대규모 개발 조직과 달리 소규모 개발자들은 어려움을 겪는다. 이러한 문제점을 해결하기 위해 본 연구에서는 멀티 플레이 게임 서버를 자체 제작하고 이를 이용할 수 있는 Unity Asset 형태의 컴포넌트를 개발하여 개발자들이 싱글 플레이 게임을 멀티플레이 게임으로 쉽게 바꿀 수 있도록 하였다.

1. 서론

현재 게임 시장의 현황을 분석한 결과 다른 유저와 함께 즐길 수 있는 멀티플레이 게임이 유저들 사이에서 큰 인기를 얻고 있으며 이런 멀티플레이 게임들의 특징을 살펴봤을 때 대부분 잘 알려진 대규모 게임회사들의 제품이다[1]. 반면 Google Play Store 게임 순위에서 100~200위 정도의 하위권을 차지하고 있는 게임들은 개인 개발자 또는 중소기업이 개발한 제품이고 멀티플레이를 지원하지 않는 것으로 보이고 있으며 이런 싱글 게임 제품들이 게임 시장에 많은 비중을 차지하고 있는 것으로 나타나고 있다. 이러한 결과는 게임회사의 경제적인 부분과 인력의 제한 때문이다.

멀티플레이 게임을 개발하기 위해서는 게임 서버의 구축이 필수 요소이다. 하지만 (그림 1)에서 제시한 바와 같이 멀티플레이 게임은 데이터 송수신이 빨라야 하므로 두 디바이스 간의 직접적인 통신이 요구된다. 아울러 일반적인 서버보다 경제적, 시간적인 자원이 더 많이 소요되는 홀 펀칭이나 릴레이 서버 같은 기술이 요구된다. 이에 따라 스타트 업이나 일인 개발자 또는 소규모 개발자들은 대규모 회사와 달리 멀티플레이 게임 구현에 많은 어려움을 겪는다.

이와 같은 이유로 인해 스타트 업이나 일인 개발자가 게임 시장에서 많은 비중을 차지하고 있음에도 불구하고 두각을 드러내지 못하고 있다.

(그림 1) 멀티플레이 게임의 필요성

따라서 본 연구에서는 위에서 언급한 문제점을 인지하고, 게임 개발자가 홀 펀칭이나 릴레이 서버와 같은 기술에 대한 상세한 내용을 고려하지 않고서도 어떤 종류의 싱글 게임이든지 서버를 쉽게 이용할 수 있게 하는 것을 목표로 하고 있다.

모바일 게임을 개발하기 위한 여러 가지 개발 도구가 존재하는데, 그중 Unity[2]는 모바일 게임 개발자들에게 가장 많이 사용되는 개발 도구 중의 하나이기 때문에 본 연구에서는 Unity를 이용하여 개발되는 게임을 대상으로 하였다.

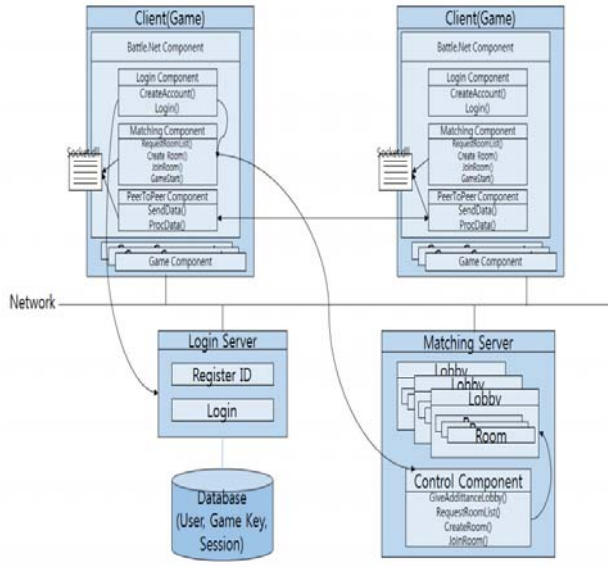
Unity라는 게임 개발 툴은 C#으로 이루어진 스크립트 언어와 싱글 스레드 루프 구조로 인해 누구든지 깊은 지식 없이도 게임을 디자인할 수 있다. 하지만 멀티플레이 게임을 위해 요구되는 통신 모듈의 개발은 어렵다. 그 이유는

스크립트 언어 레벨에서는 소켓을 사용할 수 없어서 dll로 만들어진 native code가 필요하고 싱글 스레드 구조를 극복하기 위해서 비동기적인 처리가 필요하기 때문이다. 본 연구에서는 이러한 불편한 점들을 개선하고 멀티플레이 게임을 편하게 만들기 위한 플러그인을 제작하고, 서버개발에 들어가는 비용을 줄이고자 플러그인과 연동되는 1:1 매칭 시스템 서버 제작에 초점을 맞추었다.

2. 시스템 설계

본 연구에서는 싱글 플레이 게임을 멀티 플레이 게임으로 쉽게 전환하는 데 필요한 서버를 자체 제작하고 이를 개발자들이 편하게 이용할 수 있도록 클라이언트 컴포넌트를 개발하였다. 서버 및 클라이언트 컴포넌트의 구조는 (그림 2)와 같다.

2.1 시스템 구조

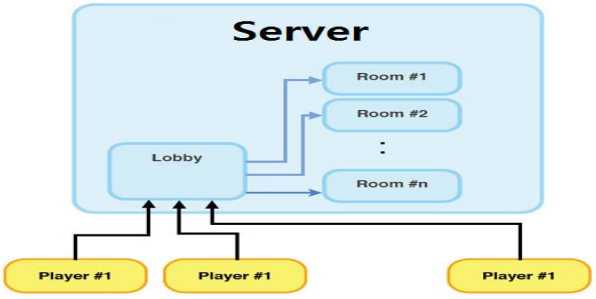


(그림 2) 멀티플레이 시스템 구성도

싱글 플레이 게임을 멀티플레이 게임을 전환하기 위해서는 사용자 관리와 두 디바이스 간의 통신이 필요하다. 이를 위해 본 연구에서는 두 개의 서버를 제작하였다. 하나의 서버는 Linux, PHP, MySQL, Apache를 이용하여 구축된 웹 서버를 통해 회원 가입과 로그인, 사용자 데이터를 저장하는 스토리지 역할을 하는 서버로서, HTTP 프로토콜을 사용하고 80번 포트로 접속 후 XML 데이터 포맷을 이용하여 통신을 수행하게 된다.

다른 서버는 Linux, C++, epoll 방식을 사용하는 사용자 매칭 서버이다. epoll방식[3]은 상태변화의 확인을 위한, 전체 파일 디스크립터를 대상으로 하는 반복문이 필요 없고, select 함수에 대응하는 epoll_wait 함수호출 시, 관찰대상의 정보를 매번 전달할 필요가 없으므로 멀티 스레드나 select 방식보다 효율적으로 요청을 처리할 수 있다.

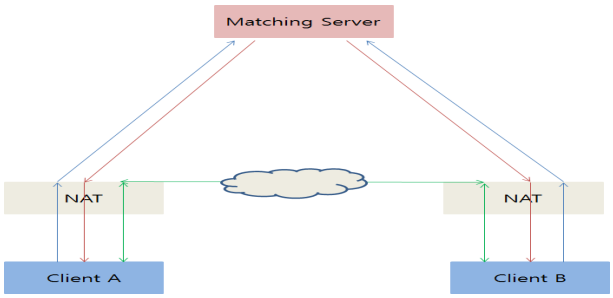
2.2. 매칭 서버 구조 및 기능



(그림3) 매칭 서버의 설계

본 연구에서는 한 서버를 Battle.Net 형식(로비 방식)으로 어떤 게임이라도 활용할 수 있는 서버를 구축하고 그와 연동되는 컴포넌트를 제작하였다. Battle.Net, 로비 방식이란 호스트가 되는 사용자가 방을 생성하고 다른 사용자가 그 방에 입장하게 되면 서버는 두 사용자를 연결해주고 서버는 두 사용자가 연결을 유지하고 있는지만 확인하고 어떤 데이터를 주고받는지 관여를 하지 않게 된다.

이 상황에서 두 사용자의 네트워크 정보를 알아 온 후에 상대방에게 서로의 네트워크 정보를 알려줌으로써 두 사용자를 연결해주어야 하는데 이러한 기술을 홀 펀칭이라고 한다[4].



(그림 4) 홀 펀칭 과정

두 클라이언트가 모두 NAT 뒤에 있다고 하더라도 잘 알려진 랑데뷰 서버를 통해서 직접적인 P2P UDP 세션을 구성케 하는 기법으로 홀 펀칭의 동작 과정은 아래와 같고, PeerToPeer Component에서 구현하였다.

1) 처음에 서버에 접속한 클라이언트 A는 어떻게 다른 클라이언트 B에게 데이터를 보내야 할지 모른다. 따라서 클라이언트 A는 UDP 세션을 구성하기 위해 매칭 게임 서버에게 도움을 요청한다.

2) 매칭 게임 서버는 클라이언트 A에게 클라이언트 B의 public과 private 종점(IP 주소, UDP 포트)을 담은 메세지로 답한다. 그와 동시에, 게임 서버는 클라이언트 B와 연결된 UDP 세션을 이용해서 클라이언트 B에게 클라이언트 A의 public과 private 종점을 담은 연결 요청 메세지를 보낸다. 이 메세지들이 수신되고 나면, 클라이언트 A와 클라이언트 B는 서로의 public과 private 종점들을 알게 된다.

3) 클라이언트 A가 클라이언트 B의 public과 private 종점들을 매칭 게임 서버로 부터 받으면, 클라이언트 A는

UDP 패킷을 두 종점 모두에게 보내기 시작한다. 어떤 종점이든 첫 번째로 클라이언트 B로 부터 유효한 답변을 끌어낸 종점과 고정되어 연결된다.
이와 비슷하게, 매칭 게임 서버로 부터 전달 된 연결 요청 메시지에 있는 클라이언트 A의 public과 private 종점들을 클라이언트 B가 받게 되면, 클라이언트 B는 클라이언트 A의 각각의 종점들에 UDP 패킷을 보내기 시작하며 첫 번째로 동작하는 종점과 고정되어 연결된다. 이 메시지들이 비동기적이라는 조건으로 순서와 타이밍은 중요하지 않다.

2.3 클라이언트 컴포넌트 구성 및 기능

클라이언트 컴포넌트는 싱글 플레이 게임을 멀티 플레이 게임으로 바꾸고 싶어 하는 개발자가 본 연구를 통해 개발된 서버를 쉽게 이용하기 위해 Unity Asset 형태로 제공되는 컴포넌트이다. 다음에서 각 컴포넌트에 대한 기능과 특징을 제시한다.

- LobbyChatting Component
  - 배틀넷 로비에서 유저들 간 채팅 메시지 송수신 처리

- MatchingGame Component
  게임 방 생성, 입장, 리스트 요청, 게임 시작
  - GiveAttendanceLobby
    로비는 게임당 하나의 로비가 생성되며 그 해당 게임의 키 값으로 식별.
  - RequestRoomList
    해당 로비에 생성된 방들의 정보를 요청 및 갱신.
  - CreateRoom
    게임 로비에 방을 생성하는 기능이며 방을 생성한 사용자는 호스트로서 바로 생성된 방으로 이동.
  - JoinRoom
    해당 방으로 접속.

- LoginBattleNet Component
  회원가입, 로그인(성공 시 Matching Server에서 서버 IP, Session 반환) 기능 수행
  - CreateAccount
    http 통신을 통한 회원가입. 사용자는 이름, 이메일, 비밀번호의 정보를 배틀넷 서버의 데이터베이스에 회원으로 등록.
  - Login
    http 통신을 통해 배틀넷 서버의 데이터베이스에 등록된 유저인지 판단하고 맞을 경우 세션 키 값을 반환.

- Socket.dll Component
  비동기적 송수신, 데이터 포맷(XML), 큐 기능 제공.

- PeerToPeer Component
  홀 펀칭을 이용한 데이터 송수신(성공시 ACK 전송).

이들 컴포넌트의 개발 언어는 C++, C#이며 통합 개발환경은 Unity와 Visual Studio 2015, 서버 환경 Linux(CentOS7), 데이터베이스는 MariaDB를 사용하였다.
개발에 사용된 스마트 폰의 기종은 삼성 갤럭시 노트4이다.

3. 사례

(그림 5) 테스트 게임화면

본 연구를 통해 구축한 서버와 컴포넌트를 테스트하기 위해 횡 스크롤 러닝 게임을 개발하였다.(그림 5)
개발된 게임은 한 캐릭터를 2명의 유저가 같이 컨트롤하는 게임으로 한명은 점프를, 다른 한명은 공격을 통해 장애물을 파괴하는 역할을 맡아 서로 협동하여 달리는 게임이다. 개발된 게임에 LoginBattleNet Component를 이용하여 사용자를 등록시키고, MatchingGame Component를 사용하여 두 사용자를 1:1로 매칭시켰다. 그 후 PeerToPeer Component를 사용하여 게임에 필요한 데이터를 두 디바이스가 직접 주고받을 수 있었다. 그 결과 싱글플레이 게임으로 개발된 게임을 개발한 컴포넌트를 이용하여 쉽게 멀티플레이 게임으로 변경할 수 있었다.

4. 결론

스마트폰의 사용량이 늘어가면서 모바일 게임 시장은 계속해서 커지고 있다. 이에 따라 많은 스타트업과 일인 개발자들이 모바일 게임 시장에 뛰어들고 있는 상황에서 본 연구에서 개발한 Unity 컴포넌트들은 멀티플레이 게임을 개발하는데 경제적, 시간적 자원이 절약될 것이며, 멀티 플레이 게임을 개발하는데 편리해짐에 따라 싱글 플레이 게임이 위주인 모바일게임 시장이 멀티플레이 위주로 활성화될 것이다.

5.참고문헌

[1] Google Play Store.
https://play.google.com/store/apps/category/GAME
[2] Unity Engine, http://Unity.com/
[3] https://en.wikipedia.org/wiki/Epoll
[4] Bryan Ford et al., Peer-to-Peer Communication Across Network Address Translators, 2005 USENIX Annual Technical Conference, 2005.