졸업프로젝트 최종보고서

네트워크 게임 앱 제작

네트워크를 이용한 협동 슈팅 게임

학 번: H201503020

학 과:컴퓨터공학과

제작자:노 동 균

목 차

1. 개발 개요
2. 계획 수립
3. 자료 조사
4. 요구사항 분석
5. 프로그램 기반 기술
6. 프로그램 설계
7. 후기
8. 개발 개요
   1. **개발 동기**

• 평소 게임개발에 관심이 많았고, 관련 직종으로 취업을 희망하고 있음

• 네트워크를 활용한 모바일게임의 대부분은 RPG 위주의 성향이 강하여 새로운 장르에

도전해보고 싶었음.

* 1. **목표**

• 네트워크를 사용하는 안드로이드 협동 슈팅게임을 제작하여 실제 서비스 하는 것을 목표로함.

1. 계획 수립
   1. **게임 소개**

|  |  |
| --- | --- |
| 게임 이름 | 시그널 : 평행전쟁 |
| 장르 | 네트워크 기반 협동 슈팅 |
| 컨셉 | SF 슈팅 우주전쟁 |
| 목표 | 스테이지를 진행하여 최종 보스 처치 |
| 특징 | 각 유저는 다른 차원에서 게임을 진행  한정된 방법으로 차원을 넘어 서포트수행  Ex1) 상호차원이동  : 서로의 전투지역을 맞바꾼다.  Ex2) 협력요청 : 차원이동  : 한쪽의 유저가 다른 유저의 전투지역에 참가,  한정된 시간동안 서포트가 가능  PTT 보이스를 통해 서로 통신하여 도움을 주고받을 수있음 |

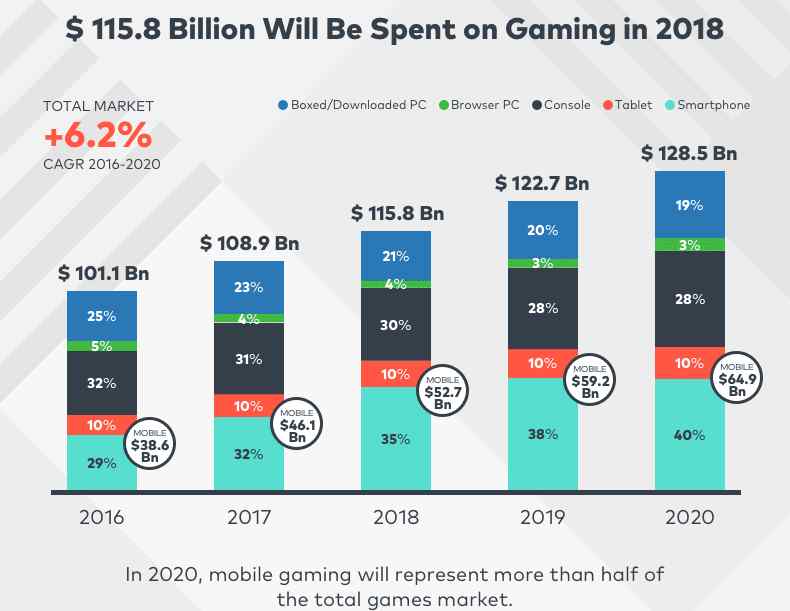
* 1. 개발 환경

|  |  |
| --- | --- |
| 개발플랫폼 | 안드로이드 |
| OS | Windows 10 Pro |
| 게임 개발도구  코드 편집기 | Unity 2018.2.11f1 Personal  MonoDevelop 5.9.6 GNU Public License |
| 사용 언어 | C# |

* 1. **개발 일정**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 9월초 | 9월말 | 10월초 | 10월말 | 11월초 | 11월말 | 12월초 |
| 개발 플랫폼 선정  및 사전 자료조사 |  |  |  |  |  |  |  |
| 게임아이디어 선정  및 구체화 |  |  |  |  |  |  |  |
| 게임 그래픽/ 사운드 선정  및 구매 |  |  |  |  |  |  |  |
| 유니티 네트워크 프로그래밍에 대한 이해 및 학습 |  |  |  |  |  |  |  |
| 게임 UI 설계 및 구현/수정 |  |  |  |  |  |  |  |
| 중간 보고서 작성 및 제출 |  |  |  |  |  |  |  |
| 게임 플레이 설계 및 구현 |  |  |  |  |  |  |  |
| 테스트 및 서비스 |  |  |  |  |  |  |  |
| 최종보고서 작성 및 제출 |  |  |  |  |  |  |  |

1. 자료조사
   1. **게임 시장 동향**
      1. 글로벌 시장

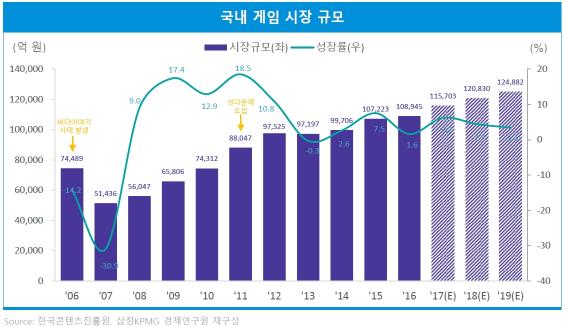


출처 : https://techstartups.com/2018/05/03/global-mobile-gaming-revenue-expected-reach-64-billion-2020-50-total-games-market/

• 해당 표는 게임의 글로벌 시장에서의 현황이다. 다름과 같이 16년 기준 1011억달러(18.10.28기준 115조 5천억원)의 시장규모를 가지고 있었으며, 17년 1089억달러(18.10.28기준 124조 4천억원)로 성장했음을 보여주고 있다. 이후 더욱더 성장할 것으로 예측하고 있으며 평균 성장률 5프로 안팎으로 성장할 것으로 보인다.

• 그중 모바일의 점유율 상승이 돋보이는데 16년 29%에서 20년 최대 40%라는 점유율 상승을 예측하고 있다.

* + 1. 국내 시장

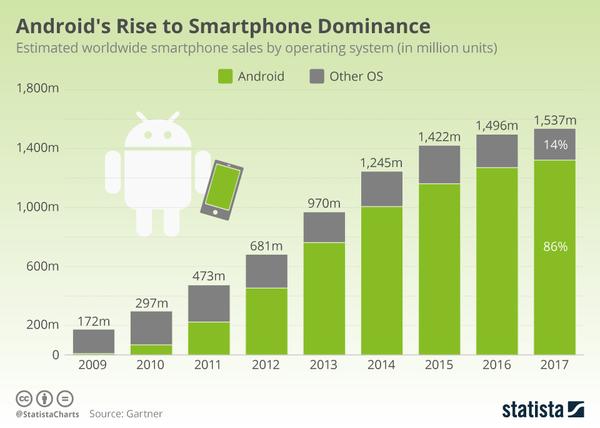


출처 : http://news.mtn.co.kr/v/2018100118045044421

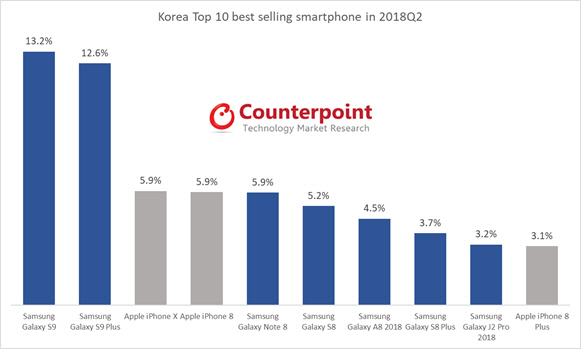
• 다양한 이유로 게임시장은 여러 굴곡을 겪었으나 08년 아이폰의 발매를 시작으로 폭발적인 성장을 기록, 11년까지 현 평균 10%이상의 성장률을 보였다. 셧다운제 도입으로 인해 성장률이 급격하게 떨어졌지만 평균 3%의 성장률을 보이고 있는 안정세로 돌아섰다.

• 16년기준 10조원 규모의 시장으로 19년에 12조 규모의 시장으로 성장할 것으로 예측하고 있다.

* 1. **개발 환경**
     1. 안드로이드
        + 시장 현황



출저: <http://news.egloos.com/4161047>



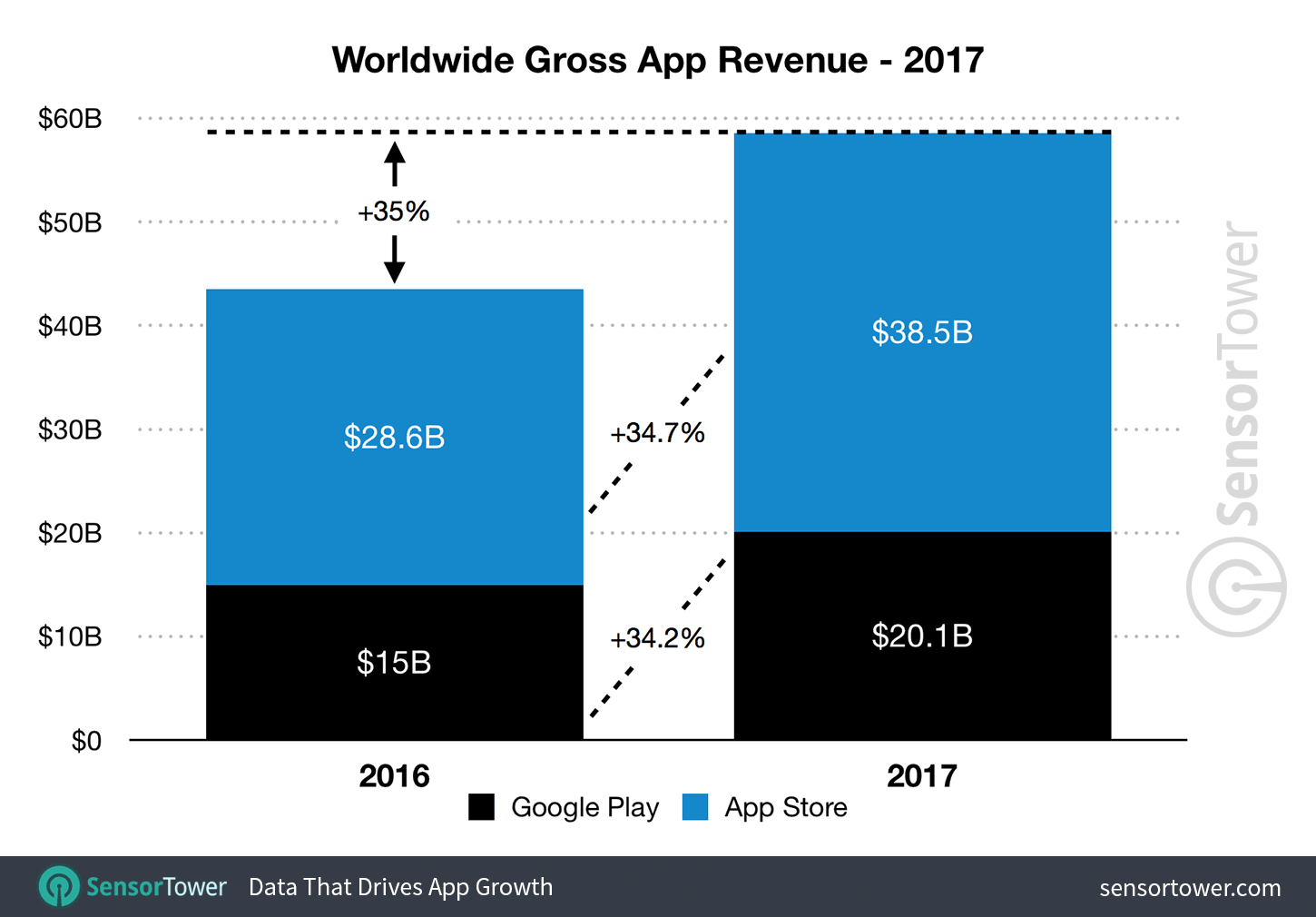
출처 : <http://www.sedaily.com/NewsView/1S4GTRNHCN>

• 08년 아이폰의 출시 이후 10년도에 본격 도입된 안드로이드는 위의 표와 같이 급속도로 발전해오고 있으며 17년기준 총 153억7천만대 중 86%의 점유율을 기록하고 있다

• 국내의 점유율의 경우 18년도 스마트폰 판매율을 비교해보면 삼성전자가 전체 48%의 점유율을 가지고 있음 판매율. 기준이라 정확하게 측정할 수는 없지만 관련 자료가 미흡하여 판매자료로 대체했다

• 한가지 아쉬운 부분은 안드로이드(구글플레이), IOS(앱스토어) 의 전세계 매출액을 비교하면 IOS의 매출액이 큼을 알 수 있다.

• 그러나 애플앱 개발의 한계는 애플 자사 컴퓨터를 통해서만 IOS 어플이 제작가능하며 자사 컴퓨터의 비용이 일반 PC에 비해 가격이 높게 측정되어있어 개발을 입문하고자 하는 초보나 초기 어플로 제작하기에는 무리가 따른다.



출처 : <http://dpg.danawa.com/news/view?boardSeq=60&listSeq=3524250>

* + 1. 유니티 (출처 : https://namu.wiki/w/유니티 (게임%20엔진)#rfn-9)



• 국내 양대 게임엔진 중의 하나로 저사양/소규모 게임 개발에 적합한 게임엔진이다. 2005년 6월에 처음 발표되어 스마트폰이 대중화 되기 전까지는 PC/MAC 기반의 플랫폼만 지원 하였으나, 스마트폰 대중화 이후 IOS, Android 같은 모바일 플랫폼, 다양한 콘솔기기(PS3,4, XBOX등) 플랫폼을 지원하게 되면서 폭발적으로 성장한 게임엔진 도구이다.

• 2015년 GDC(Game Developers Conference)에서 unity4의 완전 무료화 선언 및 Unity5의 출시 및 조건부 무료화 선언으로 게임산업 및 엔진 발전에 기여한 현황이 있다.

• 유니티 라이선스 비용

- 연매출 20만불 이상 : 월 140000

- 연매출 20만불 이하 : 월 40000

- 연매출 1억 이하 : 무료

• 저렴한 라이선스 비용에 더불어 애셋스토어라는 게임 리소스 마켓과 개발자끼리 리소스를 사고 팔 수 있는 장터를 운영하고 있다. 이를 통하여 이미지나 사운드를 비용만 치르면 쉽게 구할 수 있게 되었으며 비용을 치루지 않아도 사용할 수 있는 무료 리소스들도 많이 존재해 소규모 개발, 인디게임 개발에 많은 도움이 되고 있다.

• 개발언어로 C#과 자바스크립트를 지원하고 있으며 자체 API 라이브러리를 구성하고 있다. 이를 통해서 직관적인 개발이 가능하게 되어 이전보다 간단하게 게임을 구성할 수 있다.

1. 요구사항 분석
   1. **시스템 부분**

|  |
| --- |
| 1. 모바일에서 호스팅하여 운영될 것  - 스마트폰 호스트로 운영, 서버역할을 하며 동시에 클라이언트로서도 동작하게끔 해야 함 |
| 2. 실시간으로 상호작용이 가능할 것  - 모바일 기기특성상 통신감도에 따라서 끊기는 경우나 강제종료 발생에 대비한 처리절차(동기화 관련) 구현  - 다양한 상황(차원이동 등)의 적절한 대처를 위하여 서로 각자의 상황을 저장하고 있어야 됨. |

* 1. **리소스 부분**

|  |
| --- |
| 1. 그래픽 Drawing 능력 부족 해결 필요 |
| 2. SF에 어울리는 배경 / 유저 / 몬스터 / UI 이미지 필요 |

* 1. 게임 구현 부분

|  |
| --- |
| 1. 직관적이고 인체공학적인 UI 설계(모바일)  - UI 및 화면간 이동에 대한 use case를 시각화 |
| 2. UI와 게임플레이어 간의 상호작용 적용 |
| 3. 몬스터 리스폰 수행 시스템 설계 |
| 4. 몬스터 및 보스 총알 패턴 디자인 |
| 5. 로비<-> 게임씬 상호 이동 시 데이터 안전 전달 및 시현 |
| 6. 호스트 구축 및 클라이언트 접속 구현, 데이터 공유 및 시현 |

1. 프로그램 기반 기술
   1. **시스템 부분**
      1. 모바일 호스팅

* 로컬 Network Manager를 이용

\* NetworkManager

* + 유니티에서 Network 프로그래밍의 중추로 게임상태 관리, 네트워크 정보관리, 유닛 스폰관리, 화면관리, 디버깅 정보 엑세스, 멀티 플레이어의 프로젝트의 상태 관리를 제어 할수 있음.
  + 이 클래스의 StartHost() 메소드를 통하여 서버이자 클라이언트로서 작동할 수 있음.
  + 모든 디바이스에서 동일하게 작동하며 크로스 플랫폼도 지원한다.
  + 모든 통신은 UDP로 진행이 된다,
    - UDP 통신
    1. 실시간 상호작용/ 네트워킹
* NetworkManager의 networkAddress에 host ip, networkPort에 host port 정보를 입력하고, StartClient() 메소드를 실행시키면 연결됨
* 모든 네트워크 통신은 NetworkManager를 통해 자동적으로 진행되게 설계 되어있다, NetworkBehavior클래스를 상속한 클래스가 필요하다, 여기에 움직임, 객체 생성, 데이터 싱크 등을 지원하는 명령어나 클래스를 통하여 정보들을 서버에 전달한다.
* 움직임 지원 : NetworkTransform 클래스
* 네트워크상 독립개체 선언 : NetworkIdentify 클래스
* 네트워크상 객체생성 : command,ClientRPC 명령어 및 spawn메소드
  + [command] 명령어

클라이언트에서 실행한 메소드 내용을 서버에서 수행시키게함.

* + [ClientRPC] 명령어

서버에서 실행한 메소드 내용을 클라이언트에서 수행함.

* + Spawn 메소드

클라이언트 시스템에 생성된 오브젝트를 타 클라이언트 시스템에 생성시키고,

싱크 시켜주는 오브젝트 (command)와 같이 써야함.

사용법 :

[command]

public void method(){NetworkServer.Spawn()}

* + 데이터 싱크 : [SyncVar] 명령어와 hook 속성

[SyncVar] 속성 : 데이터 변동을 서버에서만 실행하게하는 것

hook. = SyncVar에 의해 데이터 변동이 감지되면 모든 클라이언트에서 작동할

콜백메소드를 지정하는 것

* 데이터는 자동으로 공유되지만 변동사항에 대해 체력바 같은 외부 객체에는 적용이 되지 않으므로, 콜백 메소드를 지정하여 데이터를 직접 시현하는 외부 UI 등을 변경함.

사용법 :

[SyncVar(hook =”실행메소드”)]

public int a;

1. 프로그램 설계
   1. **게임 플레이 요약**
2. 게임플레이는 비행슈팅게임과 비슷함.
3. 최대 3명의 유저는 서로에게 영향을 줄 수 없는 독립된 필드위에 홀로 소환
4. 각 유저는 자동으로 발사되는 총알과 스킬(미사일 2개, 일반스킬 1개) 3개,

그리고 **차원 서포트 기능을** 이용하여 적 기체를 파괴

1. 모든 유저가 파괴한 적 기체가 일정이상이 되면 모든 유저에게 보스가 등장.
2. 모든 유저에게서 등장하는 보스는 동일한 보스로 체력을 공유한다. 보스의 체력이 모두 소진되면 게임은 종료됨.

다음의 모습은 두 유저의 플레이의

모습을 간략화 하여 표시한 화면이다.

UI에 대한 설명은 다음장에 있다.

각각 독립된 공간에서 싸우게 되며

차원 서포트라는 기능을 이용하여 서로를 도울수 있다.

다음 모습은 두 유저가 위험도 포인트를 채워

보스를 소환한 모습이다.

보스는 위험도 포인트를 채우는 순간

모든 유저에게서 동시 등장.

보스는 여러가지 패턴을 가지며

보스패턴의 수행은 유저간 독립적으로 운영된다.

* 1. **Scene**
* Unity는 Scene이라는 단위로 게임 구현부를 나눌 수 있음.
* Main, Connect, Game Scene 나누어 구현
* Main : 시작화면(다른 기능 없음)
* Connect : Host/Client 선택, Host 서버 Open,

Client 네트워크 연결, 대기실 기능을 포함함.

* Game : 실제 게임을 진행하는 Scene

* 1. **UI**
     1. Main Scene
* 게임의 시작화면으로써의 기능만

수행, Start 클릭시 Connect Scene으로 이동함.

* + 1. Connect Scene
* Host/Client Select UI
  + Host에 접속할 수 있는 위치이자,

대기실의 기능 수행

* + Host 버튼: Host UI로 변경(Host 접속)
  + Connection 버튼:

Connect UI 로 변경(Client 접속)

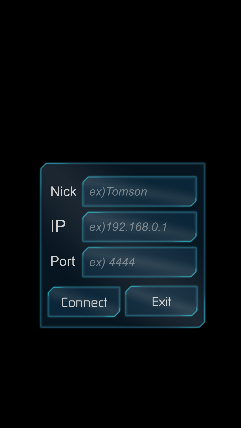
* + Exit 버튼 : 메인화면으로 이동.



* Host UI
  + Nick : 닉네임 입력
  + Port : Hosting할 포트번호
  + Create 버튼

: Server Open 및 WaitRoom

이동( Server이자 Client)

* + Exit 버튼 : Host/Client Select UI이동
* Connect UI(Client UI)
  + Nick : 닉네임 입력
  + IP : Host 서버의 IP 입력
  + Port : Host 서버의 포트 입력
  + Connect 버튼: IP/Port을 이용하여 Host 접속
  + Exit 버튼 : Host/Client Select UI이동
* WaitRoom
  + Server :IP/Port 정보를 표시함.
  + Example : 유저의 이름을 표시함.
  + NotReady : 준비상태를 보여줌
  + Ready 버튼 :
    - 클라이언트라면 Host에 준비상태 전송
    - 서버라면 Start로 표기되며 모든

클라이언트가 Ready상태가 되면 작동됨.

* + Exit 버튼 : Host/Client Select UI로 이동함.
* Game Scene
  + 1 : User 상태 및 차원 서포트 수행시

User 선택 창

* + 2 : Pause 버튼으로 Pause 창을 킴

Pause 를 눌러도 게임은 지속 진행.

* + 3 : 체력바
  + 4 : 특수 스킬 및 고유기체 스킬(구현예정)
  + 5 : 특수 미사일1
  + 6 : 특수 미사일2
  + 7 : 유저

* 1. **게임 구현 부분(Class 계략 설명)**
     1. **NetworkLobbyManagerExtend**

Game의 전반적인 관리와 네트워킹을 담당하는 클래스

Button의 기능 중 네트워크에 연관된 동작을 수행 시 이곳에서 수행됨.

NetworkLobbyManager에서 네트워크상 발생할 수 있는 콜백메소드를 지원함

* + 1. **CommonButtonScript**

단순한 기능만 필요한 버튼들의 기능 구현

Main에서의 Start,Exit 버튼 여러 씬에서의 Exit버튼 구현(단순 Scene 이동)

* + 1. **ConnectSceneButton**

ConnectScene(게임 접속 씬)에서의 버튼 동작 구현

게임 접속씬의 모든 UI를 관리하는 개체 및 대기방에 접속할 때

네트워크의 정보를 NetworkLobbyManagerExtend에 전달하여 서버에 접속

Host(서버와 클라이언트를 동시에 수행)로 접속도 가능

* + 1. **InGameButtonScript**

게임 수행 씬에서 간단한 Button 동작을 담당하는 개체

Ex) Pause 버튼 클릭후, 나가고자 할 때

* + 1. **ParentButton**

게임 수행 씬에서 게임플레이에 필요한 동작을 수행할 때 필요한 버튼의

최상위 개체, 게임씬에서 유저와 관련된 일을 수행할 시 Parent 버튼을 상속

* + 1. **SkillButton**

Parent 버튼을 상속한 개체, Gameplay 유저에 있는 skill을 선택하여

Skill을 사용하게 해주는 버튼 개체

* + 1. **UserUIButton**

타 유저의 정보(체력정보)와 이름과 동기화됨.

버튼을 클릭함으로서 타 지역에 있는 유저를 도울 수 있게됨(미구현)

* + 1. **UserUIButtonMon**

UserUIButton이 눌렸는지를 검사하는 객체, UserUIButton 3개중 1개만

눌려 있을 수 있도록 한다.

* + 1. **outBoundary**

화면을 빠져나가는 게임객체들을 제거시켜주는 개체

* + 1. **TableSetter**

로비에서 게임 수행 씬에 접속시 필드 초기화를 담당하는 개체

TableSetter를 통해 outboundary와 Spawner가 생성됨.

보스의 등장과, 게임의 종료 여부를 관리하는 개체이기도 함.

* + 1. **Spawner**

몬스터들을 일정주기로 생성 시켜 주는 개체.

TableSetter에 의해 생성되며, 위험도 게이지(체력바옆 빨간게이지)가

일정이상 차오르면 TableSetter에 의해 다시 제거됨.(보스 소환을 위함)

* + 1. **LobbyPlayer**

Waitroom 상에서 Player의 정보를 보여주며, 동시에 LobbyPlayer의 상태를 관리하는 객체

* + 1. **LobbyPlayerList**

LobbyPlayer의 접속 및 로비상의 유저를 보관하는 객체

* + 1. **MoveObject**

게임 수행씬에서 움직이는 모든 객체의 최상위 객체, Enemy와 Player는 MoveObject를 상속하여 구현을 수행한다.

HP 정보, 발사할 탄, 발사속도, 탄 속도, 탄 생성위치 등을 관리하며

HP 변동, 탄 생성 및 발사를 수행한다.

* + 1. **GamePlayer**

MoveObject를 상속한 개체 타 유저객체, 본인의 유저객체를 초기화하고 UI에 대응시켜 정보를 표시한다.

Game 상에서 직접 표현될 게임 오브젝트, 게임 수행씬 UI를 통해 조작이 가능

NetworkIdentity, NetworkTransform을 포함하여 독립개체 선언 및 위치정보를 네트워크에 공유

그 외 [command] 명령을 통하여 weapon 오브젝트 생성시, 공유

* + 1. **Weapon**

총알, 미사일등의 상위객체, 움직임속도, 데미지, 등을 관리하는 개체

충돌 시 데미지 및 탄 삭제 등의 역할도 수행

* + 1. **EnemyWeapon**

Weapon을 상속하고, 내부 내용 중 Player에 데미지를 입히도록 변경한 개체

* + 1. **Bossbumb**

Boss 패턴 중 동그란 총알에 대한 내용을 구현한 개체,

생성 이후 일정시간이 지나면 파괴되며 파편을 뿌리는 개체

* + 1. **EnemyObject**

모든 적 기체의 최상위 개체 유저를 향하는 탄을 발사하는 내용,

파괴되었을때 위험도 게이지 변동을 수행하는 개체

* + 1. **Slayer**

총을 3발씩 발사하는 적 기체의 내용을 담은 개체, 3발이 발사 순간 유저

방향을 향하도록 하는 기능이 구현되어있음.

* + 1. **BossInfo**

TableSetter에 의해 생성되며, 보스들의 체력공유 및 보스 생성 및제거를

위한 개체 보스의 체력이 줄면 자동적으로 모든 보스에 동기화된다.

* + 1. **BossPrefab**

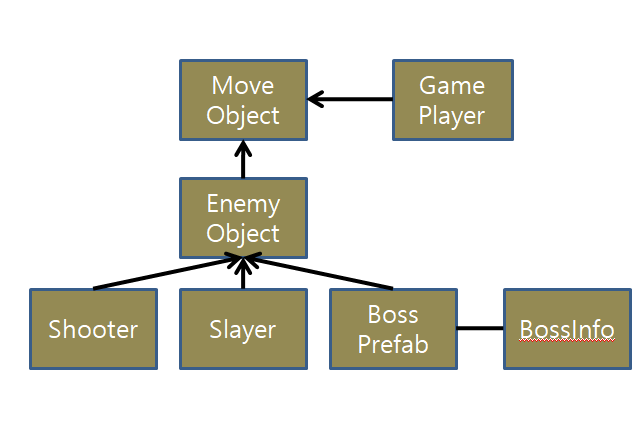
게임상에서의 보스 모습을 나타내는 개체 게임상에서 보여지는 패턴 및 이동

데미지 BossInfo에 전달 등을 담당하는 개체

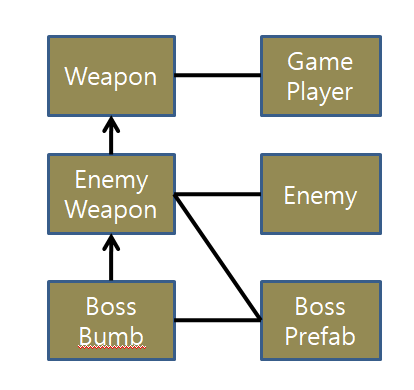
* + 1. **Skill**

스킬에 대한 내용을 담는 객체, 스킬 타입, 데미지, 지속시간, 발사될 객체, 특수기능을 포함하는 개체(미구현)

* 1. 게임 구현 부분(Class 상관관계도)
     1. 움직이는 객체(보스, 적기체, 플레이어) 관계

게임필드 위의 능동적인 개체(플레이어 적기체)는 모두 Move Object 상속

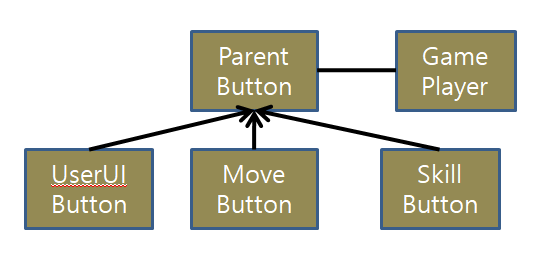
적 기체는 EnemyObject를 상속

* + 1. 무기 객체 관계

Boss Bumb은 EnemyWeapon을 EnemyWeapon을 각각 상속하고 있으며

GamePlayer는 weapon을 Enemy는 EnemyWeapon 을 사용하며

Boss는 BossBumb와 EnemyWeapon을 사용

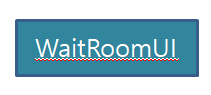
* + 1. Button 객체 관계

UserUI, Move, Skill 버튼은 모두 ParentButton을 상속

ParentButton에는 GamePlayer정보가 들어가 있으며 게임 수행 씬에

진입할 때 모든 ParentButton을 상속하는 버튼에

GamePlayer 개체를 전달

* + 1. 로비 진입

1. NetworkLobbyManagerExtend 에서 Client 접속 인식

2. Client접속을 허가하고, Client에서는 접속을 위한 사전준비

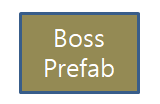
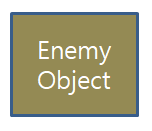
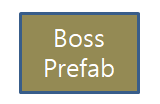
3. 준비가되면 LobbyPlayer의 OnClientEnterLobby() 메소드가 수행4. 자동적으로 LobbyPlayerList에 LobbyPlayer가 생성되어 추가됨

5. LobbyPlayerList에서 waitroom(로비) UI에 접근할 텍스트 필드개체정보를 LobbyPlayer에 전달

6. 여러 번의 네트워크 데이터 싱크를 통해 WaitRoomUI에 유저정보 시현



* + 1. 게임 진입

1. Lobby의 모든 유저가 준비되면 자동적으로 InGameScene으로 전환
2. 이때 LobbyPlayer가 유지되도록 설정/씬 전환시 자동삭제되나 DontDestroyOnLoad 옵션을 통해 씬 전환시 유지
3. NetworkLobbyManagerExtend의 OnLobbyServerSceneLoadedForPlayer를 통해 LobbyPlayer와 GamePlayer의 데이터 공유
4. 실행 후 GamePlayer는 InGameUI에 접근하여 ParentButton내 player정보 산입(버튼 누를 시 유저를 기준으로 실행되도록)
5. TableSetter는 씬 전환이 완료된 이후, Spawner(적 기체 생성)와 outBoundary를 각 유저의 위치에 생성
   * 1. 보스 진입
6. GamePlayer가 EnemyObject를 파괴
7. EnemyObject는 파괴 되며 TableSetter에 위험도게이지 추가
8. 일정이상 차오르면 TableSetter에서 Spawner를 파괴하고 Warning Message를 시현,
9. 메시지 시현 후 BossInfo를 생성
10. BossInfo는 각 유저 위치에 BossPrefab를 생성하고 BossPrefab(실제 게임객체)에 BossInfo 내용을 전달(체력공유를 위해)
11. 보스가 서서히 나타나며 보스 총알패턴을 수행.
    * 1. 게임 종료
12. BossPrefab에서 입은 피해는 BossInfo로 전달 및 네트워크 동기화
13. BossInfo에서 보스체력이 전부 소진했을 때 TableSetter에 종료 시퀀스 WinOperation()을 수행시킴
14. Win 화면이 시현되며 일정 시간 후 NetworkLobbyManagerExtend에 창 변환 요청
15. NetworkLobbyManagerExtend에서 자동적으로 모든 Client에 창변환 수행
16. 주의점 : 화면이 전환되기도 전에 LobbyPlayer의 LobbyPlayerList접근(게임수행씬에서는 없음)하므로 별도의 처리 필요
17. 별도의 코루틴(스레드와 비슷하나 데이터 동기에 안전)을 수행하여 LobbyPlayerList가 인식 될 때 까지 0.5초간격으로 확인하여 오류가 발생하지 않도록함.
    * 1. 게임 종료(유저 패배)
18. GamePlayer는 패배시 TableSetter에 패배정보 전달
19. TableSetter에서는 접속한 유저만큼의 패배정보가 전달되면 NetworkLobbyManagerExtend에 로비창 변환 요청
20. 로비창으로 전환 및 WaitroomUI 동기화
21. 후기

3개월 만에 그 동안 해보지도 않았던 실시간 네트워킹 게임을 한다는 것은 정말 어려웠습니다. 실제로 웹 서버, 홈페이지 구축과 같이 자동적으로 네트워킹을 수행해주고 실시간으로 신경 쓸 필요가 없던 것들을 여러 번 진행해왔지만 지금처럼 머리 아프지는 않았습니다. 특히나 테스트를 수행함에 있어서 두 번 이상 진행해야 했던 것, 호스트에서는 잘 수행되다가도 클라이언트를 접속시키면 잘 안 됬던 것, 그 중에서도 제일 괴롭혔던 것은 로비 씬에서 게임 수행 씬으로 넘어갈 때 진행이 안 됬 던 것이었습니다. 국내 자료는 물론 구글을 통하여 외국자료도 찾아보았었지만 쉽사리 나오지 않았고, 지속적인 검색을 통해서 찾아냈던 해법은 LobbyPlayer를 게임 씬으로 이동시켜야 한다는 것이었습니다. Unity 홈피의 매뉴얼을 주로 하여 구현 중 이었는데, 매뉴얼에는 그런 내용이 전혀 없었고, 거의 몇 주를 애먹었었던 것 같습니다.

이번을 기회로 서버와 네트워크 실시간 통신을 위해서 수많은 노력이 필요하다는 것을 체감하게 되었고, 유니티 API를 사용함에도 이렇게 어려운데, 실제 Low 방면의 통신을 진행하려면 더 열심히 공부해야 겠다는 생각을 하였습니다.

실력이 부족하여 구현하고자 하였던 모든 것을 구현하지는 못하였고, 실제 게임이라기 보다는 다소 기능 구현에 초점을 맞춰 진행하게 되어 다소 밑밑한 작업이 되었던 것이 아쉬웠던 점이었습니다.

이 프로젝트를 졸업과제로만 끝낼 것이 아니라 조금 더 보완하고 수정하여 실제 안드로이드 마켓에 개시하는 것까지 이번 방학을 보내야겠습니다.

막판에 여러모로 신경 써주시던 교수님께 감사드리며. 부족한 작품이지만 설사 졸업과제 패스가 되지 않더라도, 발판 삼아 더욱더 열심히 하도록 하겠습니다.