Postmortem Report

김성원, 정수영, 주정민, 임성균, 유승민

2025.04.04 금

Fig. 1.1 게임 INFEST 개요

INFEST 개발이 시작된 첫 날이다. 사전에 김성원의 주도 하에 유승민, 주정민이 개발 직군으로 팀에 합류했고 그 후 좀비 FPS 장르에 관심이 있던 임성균, 정수영이 개발 직군으로 추가 합류했다. 약 09:00시 모두가 한 자리에 모인 후 통성명을 시작했다. 서로 간에 좀비 FPS 장르에 대한 관심과 열정을 간단하게 확인 후 기획 팀장 김성원이 게임에 대한 간단한 브리핑을 시작했다. 약 두 시간에 걸쳐 11:00시까지 기획 브리핑을 진행하며 의견을 주고받았다. 그 후 개발 팀장 유승민의 주도 아래 INFEST 개발에 필요할 것으로 예상되는 기술 브리핑을 진행했고, 동시에 팀의 개발 방향성에 대해 토의하고 정립했다. 약 두 시간을 진행했고, 13:00시에 마무리하고 1시간의 점심 및 휴식 시간을 보냈다. 14:00 부터는 INFEST 개발의 핵심 기능인 Photon Fusion2에 대한 R&D를 진행하기로 했다. Photon Fusion2의 공식 기술 문서를 훑어보며 기본 기술들을 확인하고, 간단한 예제를 각자 만들어보며 기술을 익혔다. 동시에 기획자 김성원은 플레이어/몬스터/아이템 데이터 테이블을 작성하며 R&D 후를 대비했다. 그렇게 21:00까지 R&D를 진행한 후 다음 주에 만나 서로가 습득한 내용을 공유하기로 하며 하루를 마무리했다.

2025.04.07 월

09:00시에서 13:00시까지 개발자들은 저번 주 진행했던 R&D의 결과를 공유하고 토의하며 Photon Fusion2의 개념을 개발자 모두가 이해하게 되었다. 동시에 INFEST에서 이 기술이 어떻게 적용될지에 대한 시나리오를 작성해보며 각자의 지식을 보태 최종적으로 우리가 사용할 기술을

정리하고 어떻게 사용할지에 대한 방향성을 정립했다. 기획자 김성원도 이 토의에 참여했으며 기술적인 한계를 확인해 기획을 재조정했다.

기획서	직군	세부 작업 목록	작업일	총 작업일
구조설계	기획	데이터 테이블 제작 및 기획서 정리	1	4
	개발	Fusion2 R&D 및 구조 설계	3	
전투	기획	UI 기획서 및 상점 및 전투 시스템 기획서 정리	3	6
		데이터 테이블 (캐릭터, 몬스터, 무기, 아이템)	1	
		전투 공식 테이블 작성	2	
	개발	플레이어 조작 (이동, 사격)	3	12
		무기 및 아이템 구현	3	
		무기 변경	1	
		몬스터 AI (감지, 추적, 공격)	2	
		몬스터 랜덤 스폰	1	
		공격 판정 적용	1	
		데미지 공식 적용	1	
상점	기획	UI 아트 리소스 정리 문서	1	6
	개발	상점 UI	2	
		구입 / 판매	1	
		어빌리티 구매	1	
		랜덤 스폰	1	
			총 일수	28

Fig. 2.1 MVP 개발 스펙

14:00시부터 18:00시까지는 오전에 부족했던 R&D 및 토의를 진행하는 동시에 앞으로의 로드맵을 작성/검토/공유했다. 이 과정에서 Diagram / Loadmap / Software Management Docs / Notion 등 개발에 필요한 도구들을 작성했다. 19:00시부터 21:00까지는 이전까지 작성한 자료를 토대로 업무를 분담하고 로드맵을 구체화시켰다

2025.04.08 (화) - 2025.04.11 (금)

09:00부터 21:00까지 각자 담당한 작업을 진행했다. 기획자 김성원은 데이터 테이블을 작성하고 전체적인 UI/UX 기획 문서를 작성했다.

Fig. 3.1 매칭 UI 기획

기획자 김성원은 사용자의 시선 움직임을 최우선으로 고려해 Fig. 3.1과 같은 초기 매칭 UI를 기획했다.

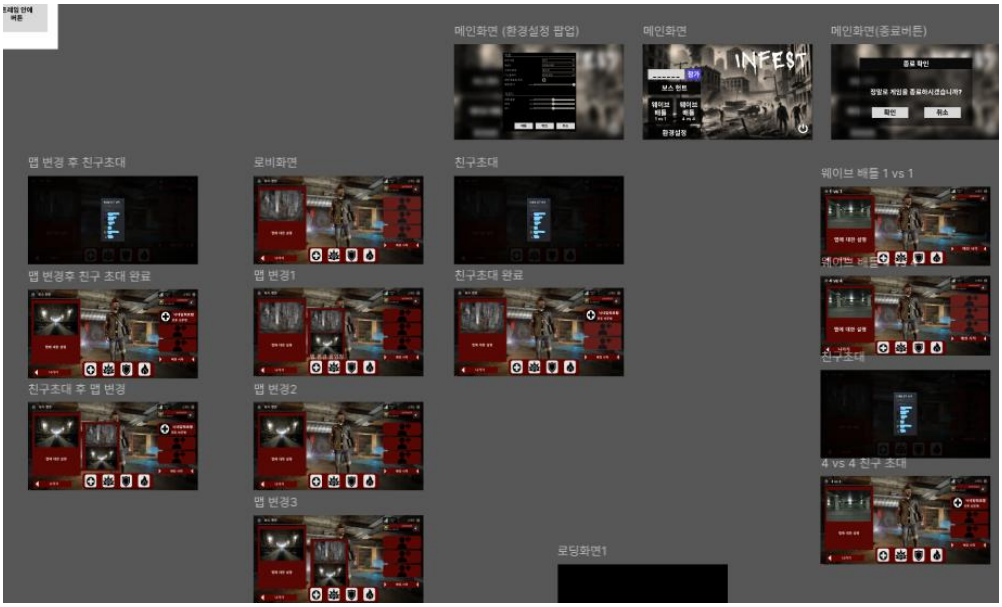


Fig. 3.2 Figma UI Wireframe

기획자 김성원은 Figma를 활용해 게임 접속부터 매칭까지의 UI 흐름을 보기 좋게 작성했다. 덕분에 개발진은 직관적으로 UI를 이해할 수 있었고 서로 간에 소통하는데 오해가 생길 일도 없었다.

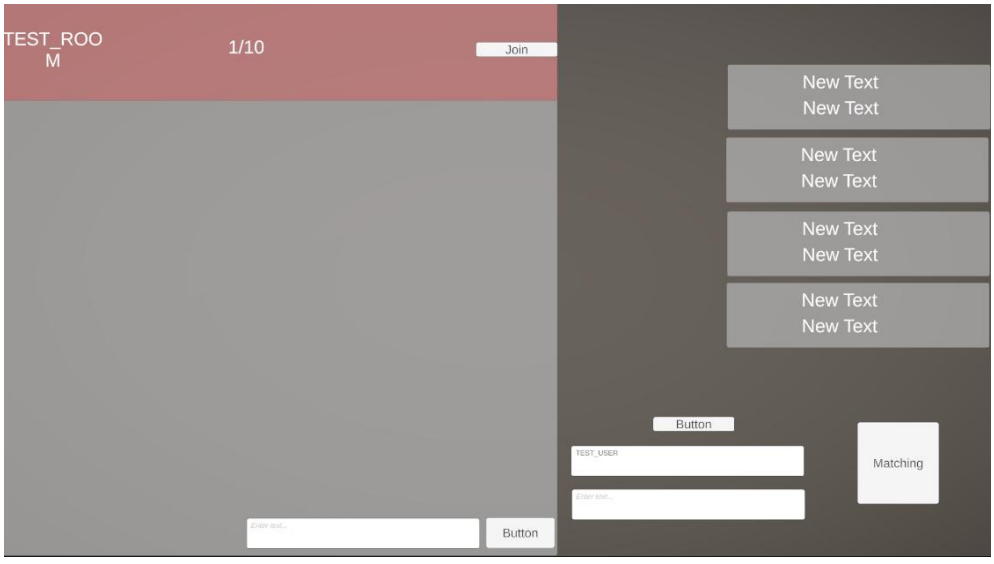


Fig. 3.3 Matching System Development

매칭 개발자 유승민은 임의로 UI를 배치하고 기능 테스트를 진행했다. 우선 Session Browser 형식으로 기능을 구현하고 정상 작동하는 것을 확인했다. 그러나, Fusion2에서 성능 때문에 Session Browser를 권장하지 않고 굳이 사용해야 할 이유도 없기 때문에 자동 매칭 기능으로 다시 개발하

기로 결정했다.

Fig. 3.4 Player

개발자 임성균, 주정민, 정수영은 플레이어 기능을 움직임, 총알, 총으로 나눠 각자 개발을 진행했다. 임성균은 플레이어의 움직임을 FSM을 통해 제어하는 구조를 선택했다. 그리고 플레이어 조작에 관련한 입력을 InputAction으로 처리하고, 이를 NetworkStruct에 넣은 후 네트워크 동기화를 진행하는 방식을 시도했다. 그 외에 1인칭 카메라로 보는 자신의 캐릭터와, 다른 플레이어가 보는 자신의 캐릭터의 모습과 애니메이션을 구분 지었다. 플레이어 모델은 Fusion2에서 제공하는 기본 예제에서 가져왔으며, 추후 출시 전에 바꿀 예정이다.

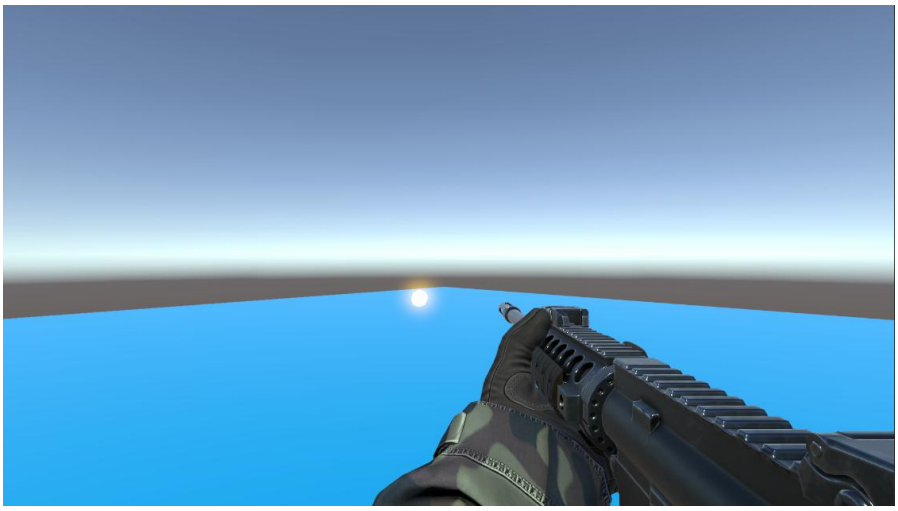


Fig. 3.5 Weapon

개발자 정수영은 총에 관련된 로직을 개발했다. 반동, 집탄율, 연사 속도, 탄창 등 총기에 필요한 변수들을 정의하고 관리해준다. 총알 발사에 관련된 과정을 메소드로 정의했으며, 추후 플레이어 쪽에서 메소드를 호출하는 것만으로 총을 발사할 수 있게끔 스크립트를 작성했다. 개발자 주정민은 총알의 발사부터 피격까지의 로직을 작성했다. Fusion2에서 제공하는 LagCompensation을 활용해 RayCast를 쏴 Hitbox라는 특수한 콜라이더에 적중하는지를 판단해 피격 계산을 처리한다. 피격 판정 계산과 플레이어에게 보이는 총알 이펙트를 분리했으며, 네트워크 동기화까지 잘 되는 것을 확인했다.

2025.04.14 (월) – 2025.04.18 (금)

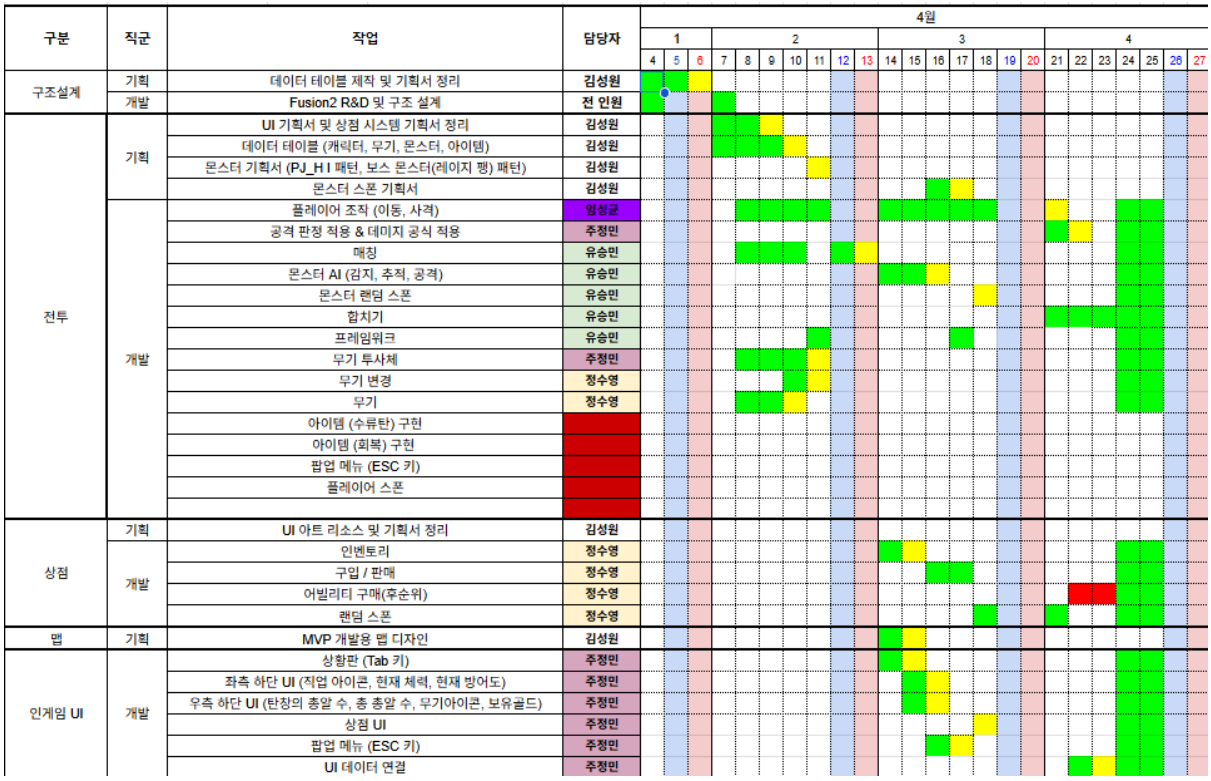


Fig. 4.1 상세 일정 보완

09:00시부터 10:30시까지 기획자 김성원의 주도 하에 팀의 상세한 일정을 확립했다. 지난 주에 작업을 진행하며 팀의 전체적인 개발 흐름이 파악하기 어렵고 앞으로의 계획이 불투명하다는 사실을 느껴 진행한 작업이다. 플레이어 하나의 기능을 쪼개 각자 개발하는 방식에 한계를 느껴 큰 분류에서 기능을 나눠 각자 담당하기로 했다. 그렇게 개발자 유승민은 몬스터, 개발자 임성균은 플레이어, 개발자 주정민은 UI, 개발자 정수영은 인벤토리를 담당하기로 결정했다.

10:30시부터 개발자 유승민은 따로 R&D를 진행하며 습득했던 지식을 팀원 모두에게 공유했다. 팀원 모두는 내용을 토대로 서로 의견을 주고받았고, 이 과정에서 서로 모르는 부분을 집어주고, 어떤 내용이 중요한지 다시 한번 짚어보며 팀의 전체적인 기술 이해도가 향상되었다.

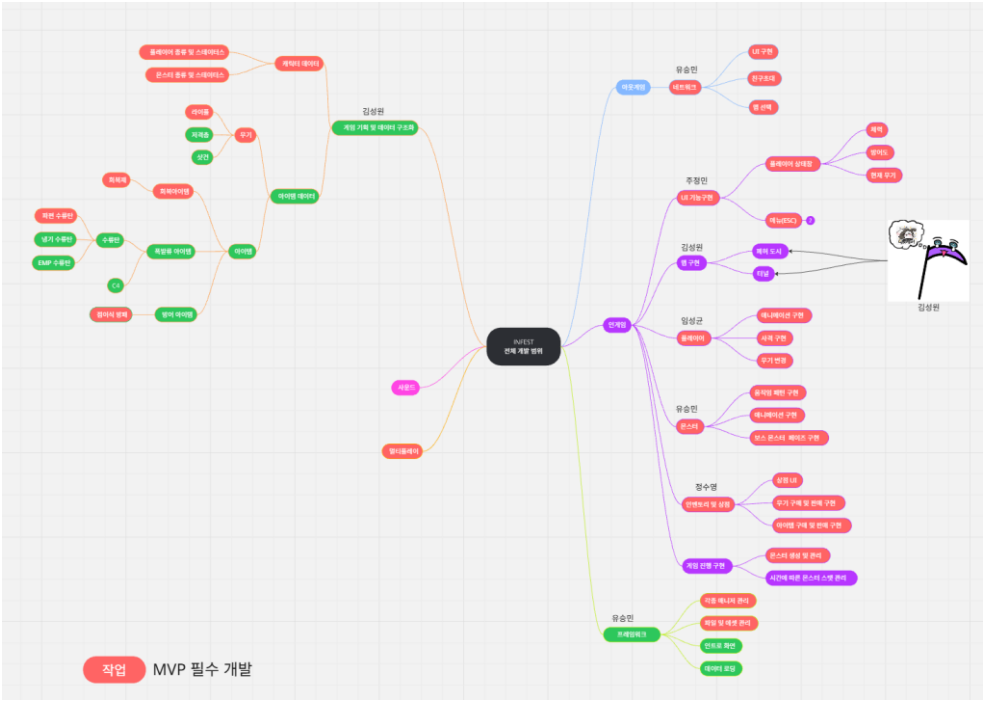


Fig. 4.2 Miro를 활용한 기능 분류

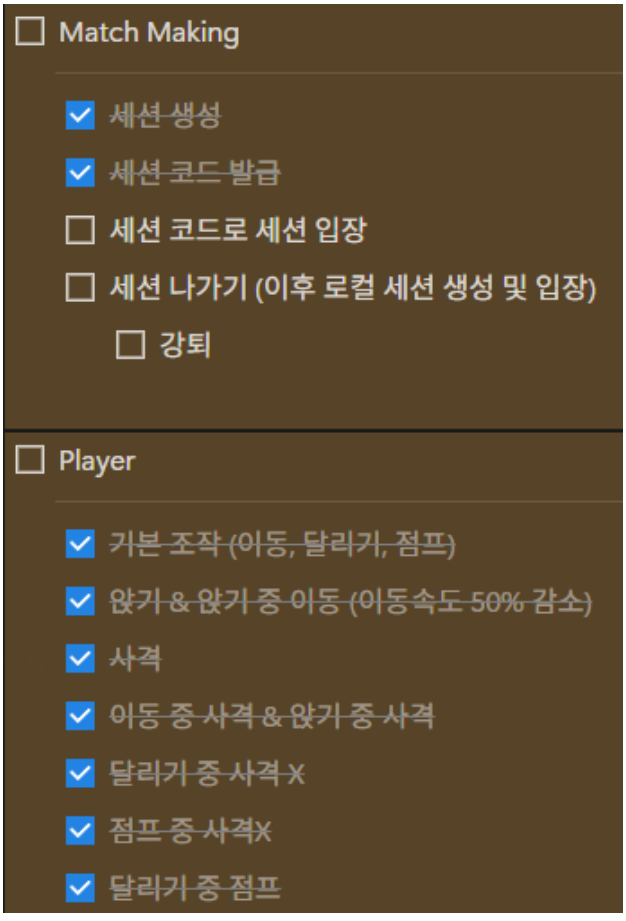


Fig. 4.3 ToDoList

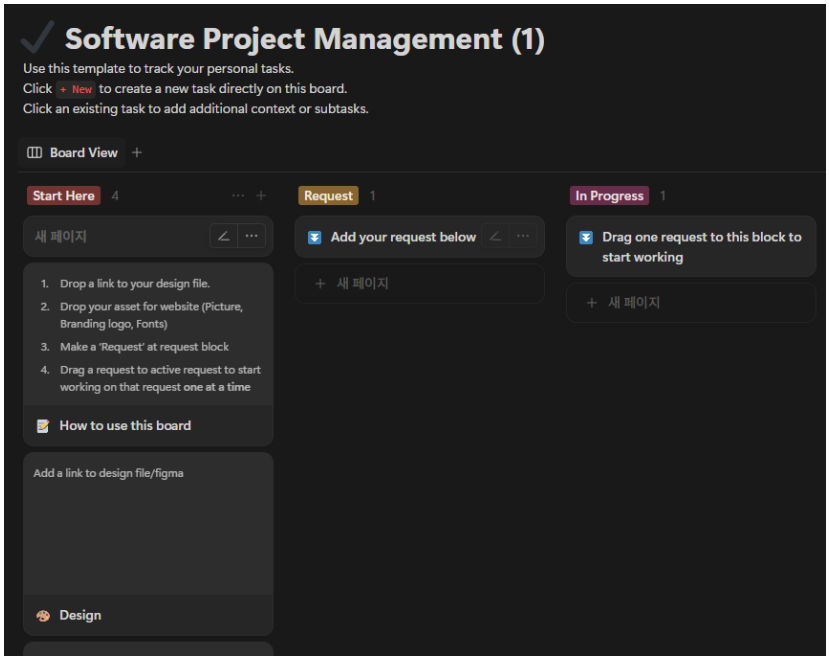


Fig. 4.4. Software Project Management

상세 일정을 확립하는 과정에서 다양한 개발 방법론이 논의되었다. 개발을 진행하며 그 중 팀에 가장 적합한 것을 찾아내기로 했다.

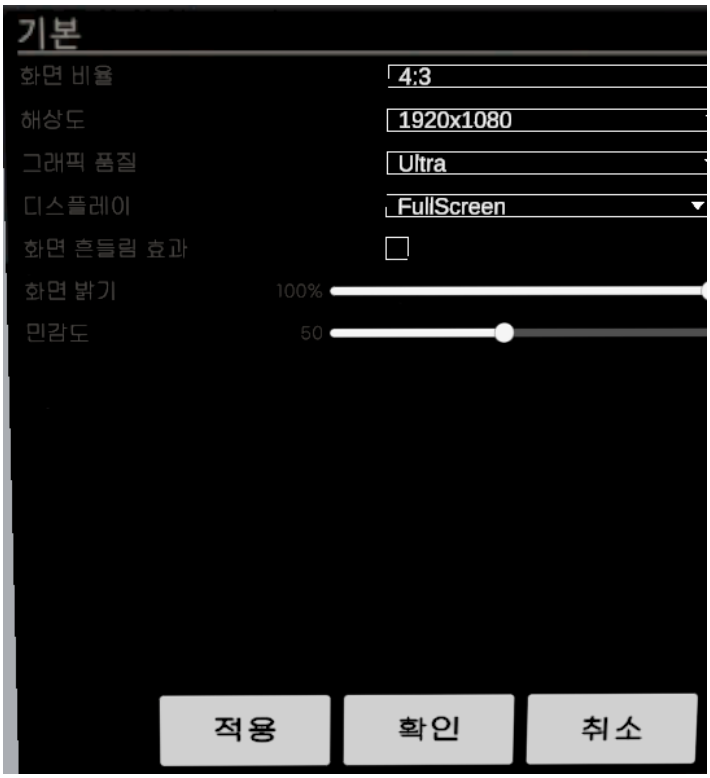


Fig. 4.5. UI

UI 개발자 주정민은 MVC 패턴의 UI 구조를 시도했다. 동시에 UI를 프리팹으로 저장 후,

UIManager를 통해 동적으로 로드하고 숨기는 관리 체계를 성립했다. UI를 보여주고 숨기는 것에 애니메이터를 활용해 보기 좋은 연출까지 삽입하였다. 네트워크 동기화가 필요한 점수판과 같은 경우 RPC를 통해 정보를 제공하고 받도록 하였다.

인벤토리 개발자 정수영은 먼저 기획 테이블에 근거한 데이터 구조를 설계했다. 기획 테이블에서 값을 가져와 저장하는 Data class, 이를 가져와 동적 데이터를 추가한 instance까지의 데이터 구조를 설계했다. 이를 이용해 주무기, 보조무기, 소비아이템, 골드 등의 정보를 저장하고 관리하는 인벤토리 class를 작성하고 추가로 상점을 구현해 구입/판매 기능까지 만들었다.

플레이어 개발자 임성균은 저번 주에 하던 플레이어 FSM 구조를 계속해서 개발했다.

Fig. 4.6 Monster PJ_HI

몬스터 개발자 유승민은 플레이어와 마찬가지로 FSM으로 몬스터를 제어하는 구조를 설계했다. 몬스터의 움직임을 제어하기 위해 NavMesh를 사용했고 네트워크에 동기화를 테스트했다. NetworkTransform이라는 컴포넌트를 사용해 몬스터의 Position과 Rotation을 동기화했고, 덕분에 몬스터의 움직임이 네트워크 상에서 잘 동작했다. 몬스터의 상태를 Phase와 Pattern이라는 큰 분류로 나눴다. 몬스터 모델과 애니메이션은 Mixamo에서 가져와 사용했다.

2025.04.21 (월) – 2025.04.25 (금)

팀은 다음 주의 MVP 발표를 위해 작업을 마무리하기 시작했다. 하루하루가 힘든 작업의 연속이었지만 팀원이 서로 친해지고 농담을 주고받으며 힘든 작업을 현명한 방식으로 극복하기 시작했다. 월요일 09:00부터 13:00 까지 회의를 진행하며 지금까지의 개발 진척 사항을 상세하게 체크했다. 기획자 김성원은 개발자들의 이야기를 들으며 기술적으로 어려운 부분이 있으면 기획을 고치는 방향으로 기획을 수정해 개발 난이도를 조정했다.

플레이어 개발자 임성균은 담당한 작업의 높은 난이도에 대해 난색을 표하며 팀원의 도움을 요구했다. 플레이어의 복잡한 구조를 Fusion2의 네트워크와 결합하는 것이 어렵다는 것을 브리핑해주었고, 어느정도 해결책을 가지고 있던 개발자 유승민이 플레이어 개발에 참여했다. 개발자 유승민은 Player가 가지고 있던 Input 로직을 밖으로 빼낸 후 Fusion2를 다리 삼아 플레이어와 연결하는 구조로 재설계했다. 이는 Fusion2가 인풋을 네트워크에 동기화 하는 구조에 근거한다. 그 후, 플레이어에서 여기저기 흩어져 있던 Input 처리 로직을 한 곳으로 모아 중앙 관리 체계를 확립했다. 약 3일에 걸친 리팩토링은 성공적이었고, 네트워크 동기화까지 잘 됨을 확인했다. 개발자 임성균은 자신의 한계를 확인하고 인정하며 자신이 맡을 수 있는 다른 업무를 찾았다.

기획자 김성원은 MVP 버전에서 사용될 맵의 레벨 디자인을 설계했고, 개발자 정수영은 이를 적용했다. 기획자 김성원은 그동안 구현했던 기능인 플레이어 / 몬스터 / 웨이브 / 상점 / 멀티를 전부 선보일 수 있는 방향으로 맵을 설계했다. 개발자 정수영은 레벨 디자인에 높은 흥미를 보이며 파밍같은 추가적인 재미요소를 구현했다.

2025.04.28 (월) – 2025.05.03 (금)

Fig. 6.1 MVP 개발 내용

기획자 김성원은 MVP 배포에 앞서 개발 내용을 정리했다. 청취자들을 위해 직관적인 시각 자료를 준비하기 위해 노력했다. 팀을 코치해주던 튜터로부터 컨텐츠 개발이 부족하다는 평가를 받았

으나, 네트워크 기술이 적용되었다는 점에서 납득될 수 있었다. 뒤이어 게임 개발을 위해 비교적 긴 개발 기간을 확보해야 한다는 피드백을 받았다. 개발진은 MVP 발표에 앞서 부족한 컨텐츠 개발과 버그 수정에 집중했다.

Fig. 6.2 네트워크 분석

발표를 진행하던 시점에 팀 분위기는 낙관적이지 않았다. 원인 불명의 튕김 현상과 온갖 버그가 개발자들로 하여금 실력의 한계를 느끼게 있었다. 또한 WebGL 빌드에 대한 한계 또한 알게 되었다. 팀이 사용하던 네트워크인 호스트/클라이언트는 웹에서 작동하지 않는다고 한다. Fusion2에서 웹에서 작동하게끔 하는 옵션을 제공하긴 하지만, 불안정하므로 가급적 쓰지 말라고 한다. 따라서 웹에는 싱글 모드만 등록하기로 계획을 잡았다.

발표가 마무리된 후에는, 개발팀장 유승민을 필두로 Fusion2 API에 대한 전반적인 리뷰를 진행했다. 유승민은 그동안 공부했던 Fusion2의 기본 개념과 사용법을 팀원들에게 공유했다. MVP 발표가 끝나고 쉬는 느낌으로 리뷰가 진행되었고, 팀원 모두가 만족했던 시간이었다.

2025.05.07 (수) – 2025.05.09 (금)

개발자 유승민은 MVP 때 있었던 네트워크 이슈의 원인을 분석하는데 하루를 온전히 할애했다. 네트워크가 불안정하면 게임의 기반 자체가 흔들리기에 유승민은 해내야만 한다는 압박감 속에서 R&D를 진행했다. 네트워크 상태 로그를 실시간으로 확인할 수 있는 Debug 씬을 따로 제작한 후 여러 상황을 시뮬레이션 했다.

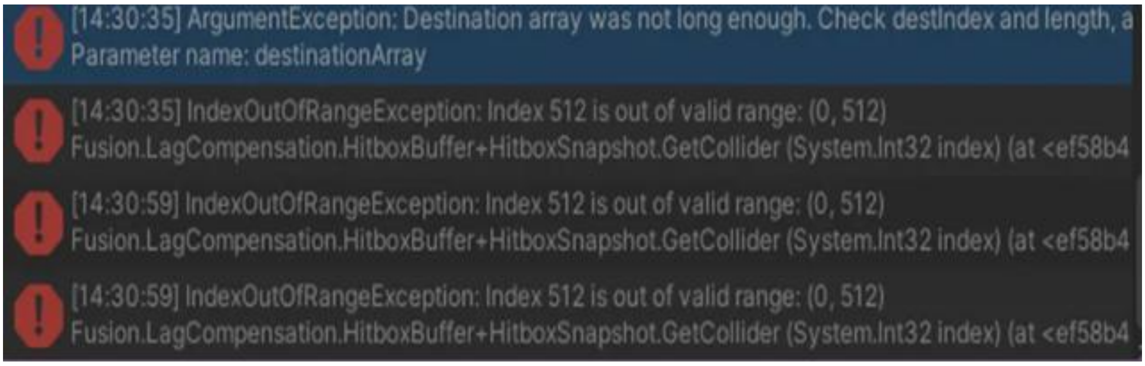
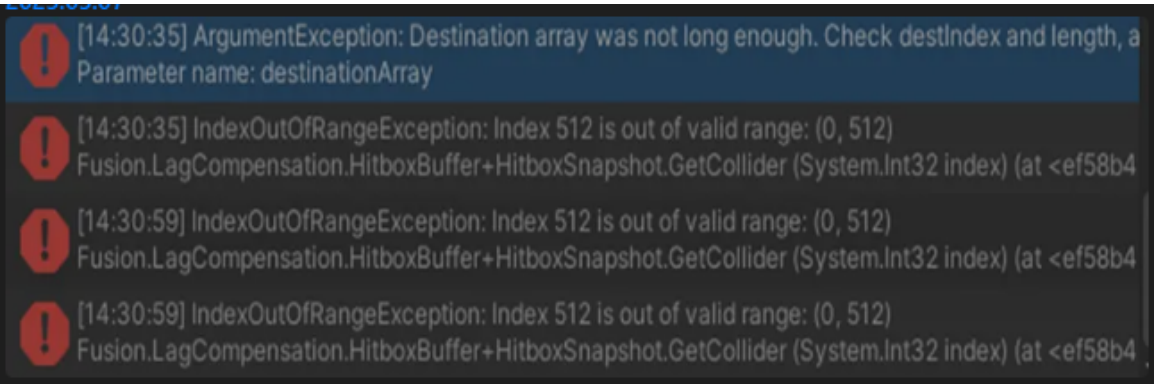


Fig. 7.1 IndexOutOfRange

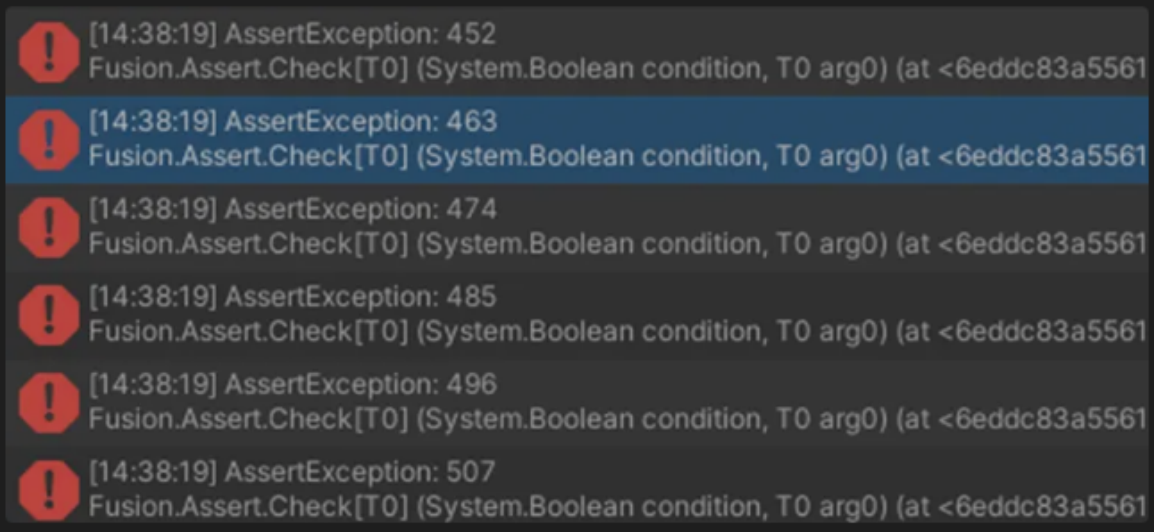
개발자 유승민이 여러 시뮬레이션을 돌려보며 확인한 결과, 많은 몬스터를 한 번에 네트워크 상에 소환하면 위 사진과 같은 IndexOutOfRange가 발생함을 확인했다. Log 내용으로 봤을 때 LagCompensation.Hitbox 관련 오류로 추측되었고, 관련 자료를 찾아보며 해결책을 강구했다.

Fig. 7.2 OutOfMemoryException

기획자 김성원은 Main Stage에서 사용될 맵을 배치했다. 네트워크 캐릭터가 NavMesh와 접촉했을 때 끼임 현상이 발생하는 걸 방지하기 위해 맵의 모든 오브젝트를 BoxCollider로 바꾸며 개발진들의 업무 부담을 줄여주었다. 동시에 NavMesh를 직접 Bakle하고 다시 수정하는 등의 기획자가 할 수 있는 개발 업무를 도맡는 열정을 보여 주었다.



좀비 300마리 스폰, 4번째 플레이어가 접속하면 발생하는 오류, 아니면 에디터라서 발생?

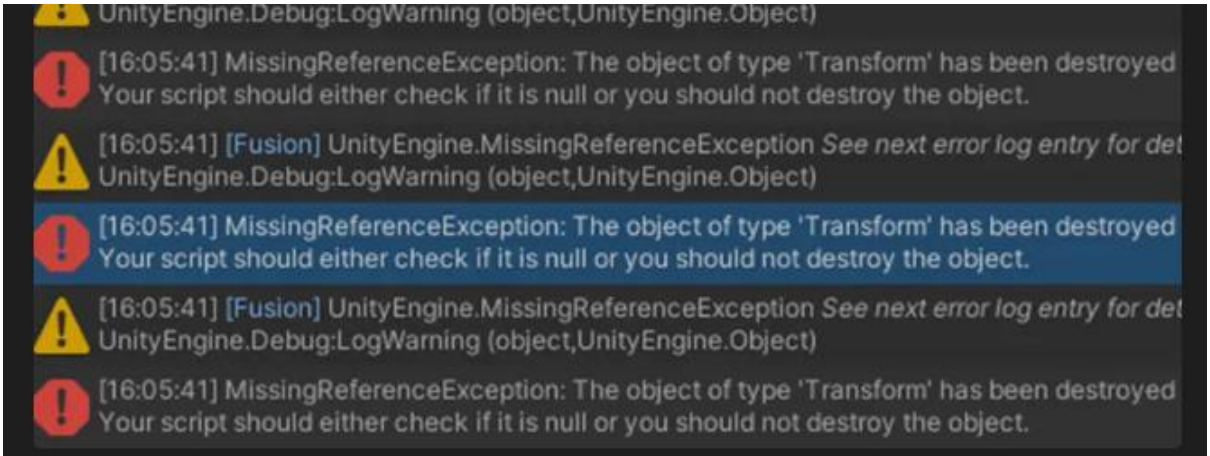


권한 없는 클라이언트에서는 히트박스가 필요 없고, 비활성화해서 퍼포먼스를 아끼는 게 맞습니다.

실제로 Fusion 공식 문서에서도 **"서버나 권한 클라이언트 외에는 히트박스 비활성화"** 를 추천하고 있습니다.

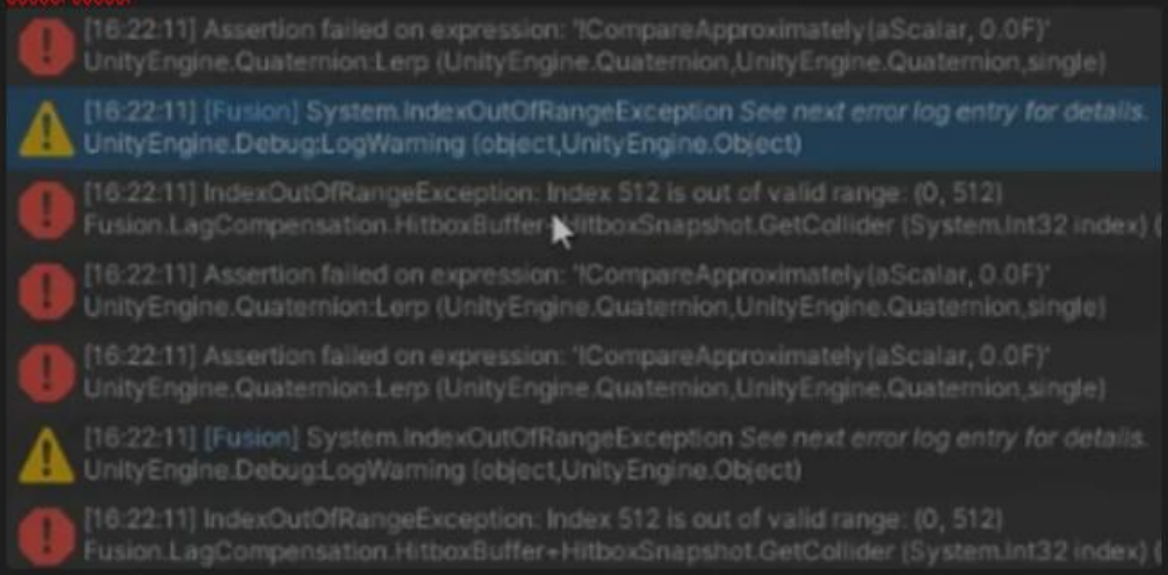
첫 클라이언트 550개의 히트박스를 잘 가져오는 듯한 모습? 제한이 512가 아니야...?

한 번에 소환이 문제인가? 300마리 한 번에 소환 > 문제 생김 > 갑자기 60마리 밖에 스폰 안 되고 나서 문제 없음 > 순차적 생성 > 몇 백 마리도 문제 없음



팀원 분이 클라이언트로 접속하려 하니까 팅굼

좀비 스폰 자체는 큰 문제가 없는 듯 보임 > NetworkTransform 써도 되긴 할 듯



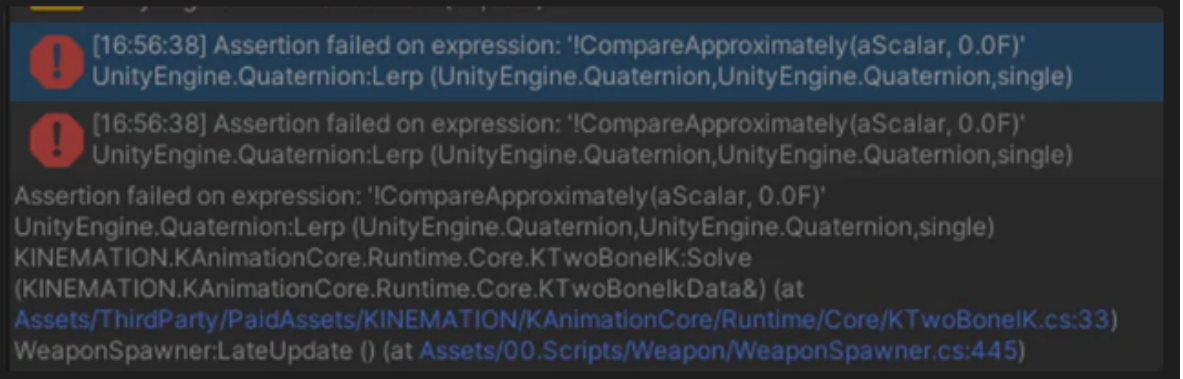
좀비가 기존에 많이 소환되어 경우, 클라이언트가 접속했을 때 발생하는 에러다. Fusion2에서 한 번에 많은 양의 Hitbox를 처리하는 과정에서 생기는 오류가 확실한 것으로 보인다. 순차적인 생성으로 해결할 수 있는 문제지만, 고려해야 할 사항이 하나 존재한다. 게임 플레이 도중에 클라이언트가 접속한 경우, 상황에 따라 제한 이상의 히트박스를 처리하려 시도할 수도 있다. 이 경우에 대한 방어 코드가 필요해 보이며 당장에 생각나는 방안은 다음 두 가지이다.

1. 클라이언트 측에서 히트박스를 비활성화한다.

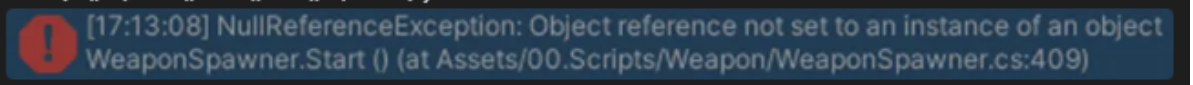
2. 클라이언트가 난입할 수 있는 상황을 제한한다.

전자의 경우 최적화 부문에서도 이점을 챙길 수 있는 듯 보인다. 만약, 클라이언트 측에서 LagCompenstaion.RayCast를 사용할 필요가 없다면 HitBox를 비활성화 해도 무방하다. 하지만, 이 경우 네트워크 최적화 방안 중 하나인 "클라이언트 RayCast 계산"을 배제해야 한다. 후자의 경우가 무난하지만, 난입 타이밍이 유저가 원하는 데로 되지 않는다는 문제점이 있다. 심지어 클라이어너트 측에서 그 상황을 제한하는데 고려할 요소가 많은 것으로 보인다. 일단, 클라이언트 측에서 RayCast 연산을 하는 방법을 고려해본 후 생각해보자.

하나 더 생각났다. 그냥 좀비 소환 제한을 걸어버리자.



히트 박스를 제거하고 한 2백 마리 스폰한 후 클라이언트로 접속했을 때 생기는 버그. INFEST 개발록-20250509100643070.webp챗 지피티가 제시하는 원인과 해결법. 타당해 보인다. 그럼 이 오류가 왜 발생하는 걸까? 그 오류의 원인을 정확히 어떻게 확인할 수 있지? 너무 많은 연산을 하면서 생기는 문제 같다만. 방어 코드에서 0과 1을 포함한 값을 return 하도록 해줬더니 해결되었다. 그리고 다음의 에러가 새로 생겨난다. (+ 1은 올바른 숫자다. 다른 쪽에서 문제를 해결해야 한다.)



오래도록 우리를 괴롭히는 에러다. 유니티 생명주기와 Fusion2 생명주기 사이의 순서 때문에 발생하는 문제로 파악된다. 네트워크 객체에서 유니티 생명주기를 사용하지 않는 방향으로 리팩토링을 진행해야 한다. 자세한 내용은 더욱 알아봐야 한다.

```
if (_weapons[i].key == Player.local.characterInfoInstance.data.StartAuxiliaryWeapon)
```

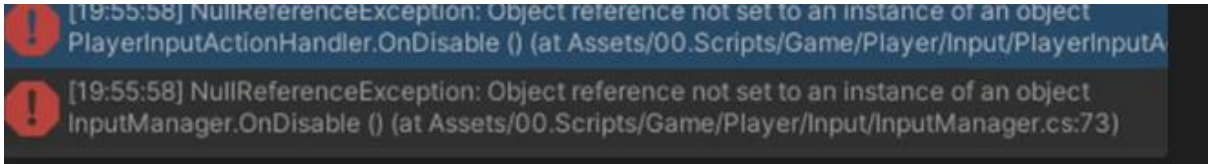
코드를 살펴보니 null 오류가 난다면 그건 Player 쪽일 수밖에 없다. WeaponSpawner는 Player가 들고 있으니까 그냥 Ref 변수를 따로 할당해주자. 아니 그럼 WeaponSpawner는 생겼는데 Player는 안 생긴 거라고? 이런 오류가 생길 수도 있구나. Player.local은 static이니까 컴파일 타임 때 stack에 저장되는 거 아닌가? 그럼 이쪽 null 오류는 아닌 걸까? 다른 Start에서 초기화를 진행하는 class가 있는 것으로 의심되는 상황이다.

일단 player를 inspector에서 ref로 가져오니까 실행이 잘 된다. 다만, 몬스터가 전부 스폰 되지 않는 문제가 발생했다. 그리고 hitbox 관련 오류는 안 뜬다.

종합하자면, 멀티가 튕기는 현상의 원인은 많은 수의 객체를 한 번에 스폰하는 점에 있다. 이로 인해 발생했던 이유는 1. Hitbox buffer 초과, 2. 많은 연산으로 인한 Querenton.Lerp의 t 값에 1이 들어가서 문제 생김. > 근데 1은 들어가야 되는 숫자인데. 3. 많은 연산으로 인한 Player.local static 변수가 null이 된다. 위 모든 문제를 해결했더라도 client 측에서 한 번에 많은 네트워크 객체를 스폰하면서 결손이 생긴다.

플레이어 움직임을 자연스럽게 하는 법은, FixedUpdateNetwork에서 GetInput으로만 조건을 거는 것. 하지만 불안정해서 다음의 문제점이 있다. 1. 뚝뚝 끊김 2. 감도가 갑자기 늘어남. 솔직히, 어떻게 완벽하게 마무리할지 잘 모르겠다. 커스텀 TRSP를 작성해야 싶기도 하고. 이건 한 번에 된다기보다는 지속적으로 지켜봐야 할 듯 하다.

매칭 > 씬 전환 > Scene 정보를 넣고 runner.StartGame > inputAction이 비활성화 되는 문제는 다음에 있다.



모종의 이유로 Disable이 호출된다... 아마 새로이 세션을 만드는 과정에서 껐다 켰다 하면서 뭐시기 한 거 같은데 정확한 원인 찾기엔 여유가 없으니 파쓰.

플레이어 동기화를 하는 방법을 새로이 생각해냈다. HasInputAuthority로 로컬 쪽에서 값 계산, Network 변수에 그걸 넣어주고, 움직임 적용은 모든 클라이언트에서 적용 되도록 하는 것.

```
public override void FixedUpdateNetwork()
{
    base.FixedUpdateNetwork();

    if (HasInputAuthority)
    {
        if (isMenu) return;

        if (statHandler.CurrentHealth <= 0) return;

        if (GetInput(out NetworkInputData data))
        {
            Vector2 mouseDelta = data.lookDelta;

            float mouseX = (yRotation + mouseDelta.x) * _sensitivity * Time.deltaTime;
            float mouseY = mouseDelta.y * _sensitivity * Time.deltaTime;

            // 좌우 회전 (플레이어)
            _parentTransform.Rotate(Vector3.up * mouseX);

            // 상하 회전
            //if (_cameraHolder.rotation.eulerAngles.x > 80f)
            //    return;
            //else if (_cameraHolder.rotation.eulerAngles.x < -80f)
            //    return;
            //else
            //  _cameraHolder.Rotate(Vector3.right * -mouseY);

            // 상하 회전 (카메라 홀더만)
            xRotation -= mouseY; // 위로 이동하면 음수, 아래로 이동하면 양수
            xRotation = Mathf.Clamp(xRotation, -80f, 80f);

            //_cameraHolder.localEulerAngles = new Vector3(xRotation, 0f, 0f); // X축 회전만 적용
            _cameraHolder.localRotation = Quaternion.Euler(xRotation, 0, 0);
            //_parentTransform.Rotate = Quaternion.Euler(0, mouseY, 0);
        }
    }
}
```

이 코드에서, 좌우 회전은 동기화가 되는데 상하 회전이 동기화 되지 않는다. 희안하네. 다시 한번 확인해보니까 클라이언트 측에서만 회전되고 동기화가 되지는 않는다. 회전 값 적용하는 코드를 if 문 바깥으로 꺼내야 한다. 상하회전은 왜 적용되지 않는지는 의문이다.

이 코드에서, 좌우 회전은 동기화가 되는데 상하 회전이 동기화 되지 않는다. 희안하네. 다시 한번 확인해보니까 클라이언트 측에서만 회전되고 동기화가 되지는 않는다. 회전 값 적용하는 코드를 if 문 바깥으로 꺼내야 한다. 상하회전은 왜 적용되지 않는지는 의문이다.

if문 바깥으로 회전 적용 코드를 뺐지만 여전히 서버에 동기화가 되진 않는 모습이다. 이유가 뭘까~~ Networked 변수를 클라이언트가 바꾼다고 해서 서버에 반영이 안 되는 걸까?

에디터에서 호스트를 실행해 확인한 결과 클라이언트에서 변경한 네트워크 변수 값은 네트워크에 반영되지 않는 듯하다. 문득, Render에서 카메라를 갱신해주면 되지 않을까 하는 생각이 든다. 일단 HasInputAuthority 조건문으로 로컬에서 입력을 네트워크에 알리는 방식은 안 되는 듯.

2025.05.09

- Player.local과 같은 코드는 유지 보수가 힘들어 보인다. 그래서 고치고 싶었는데, 콘텐츠 개발 기간이 1주일도 안 남은 시점이라 뭐부터 할지 모르겠다. 음... 일단 Player.local은 Store 쪽에서만 사용되고, 앞으로 더 쓸 일이 없지 않을까? 그럼 나중에 해도 되지 않을까? 그래서 콘텐츠 개발부터 먼저 하기로 결심했다.
- Late Join의 경우, 존재하는 모든 네트워크 객체들을 runner.spawn() 한다는 개념으로 이해된다. 이걸 커스텀 할 수 있는 방법은 없나? ObjectProvider 같은 걸 사용해야 할까?

Miscellaneous

This section contains settings that handle miscellaneous parameters regarding Fusion projects.

- `Enqueue Incomplete Synchronous Spawns`: if enabled, synchronous incomplete spawns (Addressables, etc.) get enqueued instead of throwing an exception. They will instead return `Fusion.NetworkSpawnStatus.Queued` and an attempt to spawn the object will occur on the next frame. Enabling this can be useful for transitioning from Fusion 1.x.

- NetworkProjectConfig의 Miscellaneous에 Enqueue Incomplete Synchronous Spawns 라는 옵션이 존재한다. 이걸 쓰면 될 거 같은데?
- NetworkProjectConfig에 Allow Client Server Modes In Web GL 옵션이 있다. 공식적으로 추천되지 않고,
- 튕기는 이유는 > Hitbox + 과도한 연산으로 인해 저희가 짜 놓은 코드 중 실행 순서가 꼬여서 방어 코드가 확실하지 않는 로직에서 에러가 뜨는 경우 > 다 방어하면 > 안 튕김 > 몇 개가 스폰이 안 됨 > Fusion2에서 옵션으로 제공해주네? > Hitbox 사이즈 제공해주네? > 과도한 연산은 막을 수 없다.
- 전 처리 후 실행해야 되는 로직이 있는데, 전 처리 전에 로직이 실행되버림
- ShutdownReason > 파일에 저장되는지는 모름 > 유니티 에디터 로그에 있을려나? > UI에 나오게끔
- 쉽지 않음 ㅎㅎ
- 저장 (json) 몬스터 위치, 상태, 플레이어 닉네임에 매칭되는 플레이어 골드

Lag Compensation

Enabling `Lag Compensation` allows hitbox detection to be more accurate to what players are seeing currently and very useful for fast-paced games such as first-person shooters. This is done by storing a set of snapshots of the game's hitboxes. These settings let users define how much information is stored within each snapshot, affecting the hitbox detection's accuracy.

> Lag Compensation is only available in `Server Mode` and `Host Mode`

- `Hitbox Buffer Length In Ms` : specifies the length buffer in which hitbox data will be stored in milliseconds; the longer the time, the longer hitbox snapshots will be stored before being overwritten by new ones.
- `Hitbox Default Capacity` : the default number of hitbox values stored per snapshot. Has a minimum of 16.
- `Cached Static Colliders Size` : the size of the cached static colliders (PhysX or Box2D) array of the default `Lag Compensation` Queries.

You can read more about Lag Compensation here.

- Hitbox 옵션도 있네... 야호...
- 각주를 달지 않은 자의 말로인가? MovementSpeed 변수를 네트워크 동기화 용으로 만들어 놓고, 그걸 까먹어서 다 지워버리다니. 더 화나는 점은 다 지우자 마자 본래 용도가 번뜩 떠올랐다는 것. Bash로 다시 되돌리자... 그럼 AIPathing.spped와 MovementSpeed

이 문서들에서는 `[Networked]` 변수의 사용과 최적화에 대한 자세한 정보를 제공합니다. 특히, 스냅샷 크기를 줄이기 위한 방법과 네트워크 성능을 향상시키는 전략에 대해 설명하고 있습니다.

스냅샷 크기 권장 사항

Fusion 2에서는 스냅샷 크기를 다음과 같이 관리하는 것이 권장됩니다:

- **32바이트 이하:** 이상적인 스냅샷 크기
- **64바이트 이상:** 경고 메시지가 발생할 수 있음
- **128바이트 이상:** 심각한 성능 문제가 발생할 수 있음
- **256바이트:** 하드 제한 (이 이상은 허용되지 않음)

각 `[Networked]` 변수의 타입에 따라 차지하는 바이 ” 가 다르므로, 변수의 수보다는 전체 스냅샷의 크기를 기준으로 최적화를 진행하는 것이 효과적입니다.

예시: 스냅샷 크기 계산

예를 들어, 다음과 같은 `[Networked]` 변수를 사용하는 경우를 고려해보겠습니다:

- `float` 2개: 4바이트 × 2 = 8바이트
- `Vector3` 1개: 12바이트
- `Quaternion` 1개: 16바이트
- `NetworkBool` 4개: 1바이트 × 4 = 4바이트

총합: 8 + 12 + 16 + 4 = **40바이트**

이 경우, 스냅샷 크기가 32바이트를 초과하므로 최적화를 고려해야 합니다. 예를 들어, `Quaternion`을 압축하거나 `NetworkBool`을 비트 필드로 결합하여 크기를 줄일 수 있습니다.

최적화 팁

- **불필요한 `[Networked]` 변수 제거:** 자주 변경되지 않거나 네트워크 동기화가 필요 없는 변수는 `[Networked]`로 선언하지 않는 것이 좋습니다.
- `[Accuracy]` **속성 활용:** 부동 소수점 변수의 정확도를 조정하여 데이터 크기를 줄일 수 있습니다.
- `[Capacity]` **속성 사용:** `NetworkArray`, `NetworkDictionary` 등의 컬렉션의 최대 크기를 제한하여 메모리 사용을 최적화할 수 있습니다.
- **커스텀** `INetworkStruct` **사용:** 복잡한 데이터를 효율적으로 전송하기 위해 커스텀 구조체를 정의할 수 있습니다.

- https://doc.photonengine.com/fusion/current/concepts-and-patterns/optimizations

Bandwidth Optimization

Optimizing bandwidth is important for a variety of reasons:

1. It reduces the costs of your application by reducing bandwidth costs.
2. It allows a larger percentage of your player base to play the game without encountering networking issues.
3. Fusion's eventual consistency transfer algorithm works best when the snapshot size of network packets is less than MTU (Maximum Transmission Unit ~1280 bytes).

- https://doc.photonengine.com/fusion/current/manual/data-transfer/networked-properties

Allowed Types

The following types are supported by Fusion and can be `[Networked]`:

Blittable Primitives			
byte	sbyte	short (Int16)	int (Int32)
ushort (UInt16)	uint (UInt32)	ulong (UInt64)	float (Single)
Blittable Unity Struct Types			
Vector2	Vector3	Vector4	Quaternion
Vector2Int	Vector3Int	BoundingSphere	Bounds
BoundsInt	RectInt	Color	Color32
System and User Defined Blittable Types			
Enums	System Types such as System.Guid	Structs	Other INetworkStructs

- 한 명의 플레이어가 움직일 때 패킷 변화량이 이렇게 크다. 권장량 맞추기 너무 빡빡한 걸?

필드에 존재하는 모든 좀비들 > 감지 범위 늘리기
Root motion을 사용ㅇ하면 믹사모 애니메이션에 맞게 오브젝트를 이동할 수 있는 모양이다. 실제로 이동하긴 하는데, 그럼 부모 객체는 어떡하라는 거야~
어찌어찌 되었다. Humanoid로 바꾸고, Avatar 생성하고, 이에 맞춰서 애니메이션 옵션 정해주고, Root를 RootTransform으로 바꿨던 거 다시 None으로 하니까 내가 찾던 옵션 뜨고 잘 된다.

2025.05.12 (월) – 2025.05.15 (금)

Scrum을 진행하며 게임 내 피격 사운드가 필요하다는 의견이 나왔다. 개발자 주정민은 이에 눈을 반짝이며 자신이 직접 피격 사운드를 녹음하겠다고 나섰다. 팀원들은 반쯤 기대하며 결과물을 기다렸고, 작업이 끝났을 때 놀라지 않을 수가 없었다. 생각보다 적절한 연기력에 플레이어가 실제로 좀비에게 맞는 듯한 느낌을 받았고, 어색하지 않고 중독성이 있어 그대로 사용하기로 했다.

2025.05.19 (월)

팀원 모두는 이번 주에 진행해야 할 유저테스트를 준비하기 위해 월요일부터 분주하게 작업을 진행했다. 자신이 개발한 게임을 곧 유저들에게 선보인다는 설렘 덕분에 팀원 모두의 사기는 최고점을 찍고 있었고, 어느 때보다 순조로운 진척도를 보여주었다. 기획자 김성원은 유저 테스트 배포에 앞서 Analytics, 설문지, 배포 멘트, 맵 디자인을 손보았다. 조금이라도 더 유의미한 데이터를 수집하는 것을 목표로, 유저의 시선에서 게임을 시뮬레이션 해보며 Analytics 데이터들을 가공하는 작업을 관철하였다. 개발자 유승민은 기획자 김성원의 설계한 Analytics 데이터 수집 코드를 게임 내에 적용해보고, 잘 적용되는지 확인하고, 필요하면 리팩토링 하는 과정을 반복했다. 상점 부분에서 무기의 Key 값이 아니라, 상점 내에서의 index 값이 넘어오는 문제가 자꾸 발생해서 약 세 번의 수정을 진행했다. 이 시점 몬스터 "Grita"가 원격에서 플레이어를 때리는 버그가 있었는데, 이 때문에 개발자를 가장 많이 죽인 몬스터로 선정되었다.

Fig. 8.1 몬스터 개발

일반 몬스터 WarZ, DeadCop과 엘리트 몬스터 Grita, Bowmeter 작업이 진행되었다. 이미 개발진들에게 애칭으로 불리며 장남, 장녀로 불리는 PJ_HI와 Stacker를 필두로 개발자 유승민이 리팩토링을 진행했고, 개발자 주정민은 엘리트 몬스터를 개발자 임성균은 일반 몬스터의 로직을 구현했다. 기획자 유승민은 기획 부분에서 몬스터들에게 부족한 부분을 짚어주었고, 개발자 주정민과 임성균은 개발 측면에서 몬스터한테 있었으면 하는 기능들을 검토했고, 개발자 유승민은 모든 의견을 반영해 몬스터 프레임워크를 수정했다.

2025.05.20 (화)

Main Stage인 Ruined City에서 플레이 테스트를 해봤다. Stats가 약 4만이 나와 팀을 코치해주던 튜터가 당황하였다. 이를 개선하기 위해 라이트 베이킹과 오큘러전 컬링을 진행하기로 하였다. 라이트 베이킹 중에는 유니티 작업이 힘들어지므로, 비교적 유니티를 사용할 일이 없는 기획자 김성원이 이 작업을 진행했다. 라이트 베이킹에 소요된 시간은 약 10시간이며, 김성원은 이 작업이 끝나는 새벽 세 시까지 기다렸다 마무리 작업까지 수행했다. 라이트 베이킹 결과물을 Commit하는데 데이터가 너무 크다는 문제점이 있었다. 기획자 김성원은 이미 새벽 작업에 지친 상태라 이에 스트레스를 많이 받고 있었다. 결국 데이터를 나눠서 commit하는 식으로 문제를 해결하였다.

무기 관련해서 유저 경험을 망칠 수 있는 치명적인 오류가 있다. 무기 애니메이션이 게임 로직에 완전히 녹아들지 않아서 총을 다 안 쐈는데 탄창이 비었다는 에니메이션이 나오거나, 애니메이션 자체가 이상하게 동작하는 문제가 있다. 개발자 정수영은 이에 탄식하며 밤 늦게까지 작업을 진행하였다. Weapon마다 존재하는 Setting 값을 조정하여 게임 기획에 맞게 값을 조정하였다. 그리고 기획자 김성원이 제작한 수류탄, RPG 미사일을 기존 애니메이션에 직접 넣고 수정하여 그럴싸한 연출을 뽑아내었다. 수류탄을 던지는 모션부터 폭발에 몬스터가 광범위로 죽는 모습에 팀원 모두가 감탄을 금치 못했다.

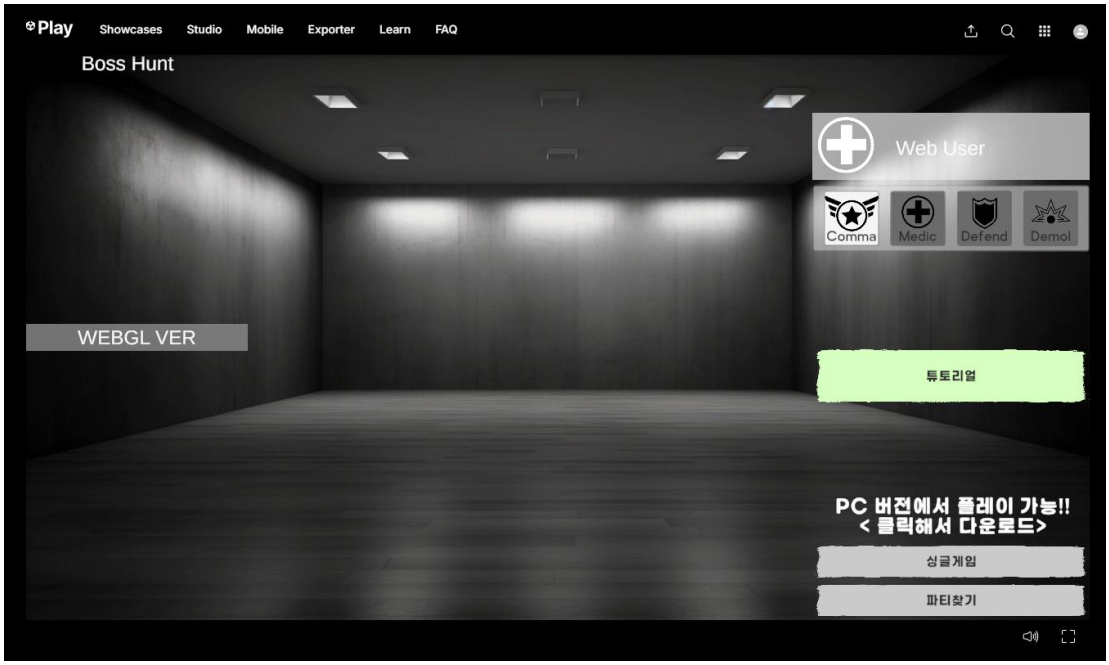


Fig. 8.2 WebGL 빌드

WebGL 빌드를 인터넷에 등록하는데 문제가 생겼다. WebGL의 꿈이 좌절되는 순간으로 기획팀장 김성원과 개발팀장 유승민은 슬프고도 막막한 기색을 감출 수가 없었다. 그래도 심기일전하여 평소의 작업 시간을 넘어선 끝에, 하나하나 해결책을 찾아내었다. 첫 번째 문제는 웹 빌드에서 텍스처가 안 보인다는 점이었다. 개발자들이 현재 가진 지식으로는 원인이 무엇인지 감조차 잡을 수 없었는데, 다행히 튜터가 쉐이더와 텍스처 문제임을 파악하고 해결책을 제시하였다. 쉐이더를 URP/Lit로 바꾸고 BaseMap을 WebGP 1025 DXTC1으로 바꿈으로써 해결하였다. 두 번째 문제는 웹에 올리기에는 파일 용량이 너무 크다는 것이었다. 이에 대해선 기획자 김성원이 MVP 때 사용했던 맵을 튜토리얼로 개조해 그것만 웹에 등록하는 해결책을 내놓았다. 이에 따라 개발자 정수영은 기본 조작, 몬스터 사냥, 상점 이용, 웨이브에 걸친 기본적인 튜토리얼 기능을 집어넣었다. Itch.io는 여전히 용량 제한에 걸렸으나, 다행히 Untiy Play에 등록하는데 성공했다. 개발자 유승민은 웹 빌드에 맞춰 UI를 수정하였고, 기획자 김성원이 만들어 놓은 설문조사/데스크탑빌드 다운로드 링크 접속 스크립트를 집어넣었다.

웹 빌드에서 몇몇 텍스처가 보이지 않는 문제를 해결하기 위해선 Shader를 URP/lit로 변경해야만 했다. 따라서 개발자 정수영, 임성균, 주정민은 Material의 Shader를 변경하는 작업에 투입하였다. 단순한 반복 작업이어서 비교적 여유가 생긴 개발자들은 서로 농담을 주고받으며 심심한 작업을 비교적 즐겁게 보냈다. 약 한 시간 정도의 반복 작업 끝에 모든 Material의 Shader를 변경하고 그에 맞는 Basemap을 할당하는 작업이 끝났다. 그러나, 이를 메인 프로젝트로 옮겨오면서 작업 내용이 반영되지 않는 문제가 발생하고 만다. 이에 주정민은 변경한 Material과 Texture를 Package로 만든 후 메인 프로젝트로 옮기는 방식을 사용해 문제를 해결했다.

2025.05.21 (수)

호스트의 움직임은 부드러운 반면, 클라이언트의 움직임은 그렇지 못한다는 치명적인 단점은 여전했다. 개발자 유승민은 했던 모든 시도들을 뒤로하고, Fusion2에서 제공하는 SimpleKCC Addon을 발굴해 적용하였고, 결국 클라이언트 측의 움직임을 부드럽게 하는데 성공하였다. 그 시간이 약 01:00시였으며, 몸에 가득한 피곤함에도 엄청난 희열을 느꼈다.

몇 일 간의 한계를 넘어선 작업 끝에 드디어 05.21 18:30에 유저테스트 배포를 시작했다. 배포 후 팀원 모두는 긴 휴식 시간을 보냈고, 저녁 때 다시 모여 다른 팀이 만든 게임들을 즐겼다. 그 사이에 몇 가지 피드백이 들어왔다. 개발자 유승민은 그 피드백을 보고 한숨을 내쉬었는데, 그 내용 중 하나가 멀티가 되지 않는다는 것이었기 때문이었다. 때문에 개발자 유승민은 이미 한계인 몸을 이끌고 23:00까지 핫픽스를 진행했다.

2025.05.22 (목)

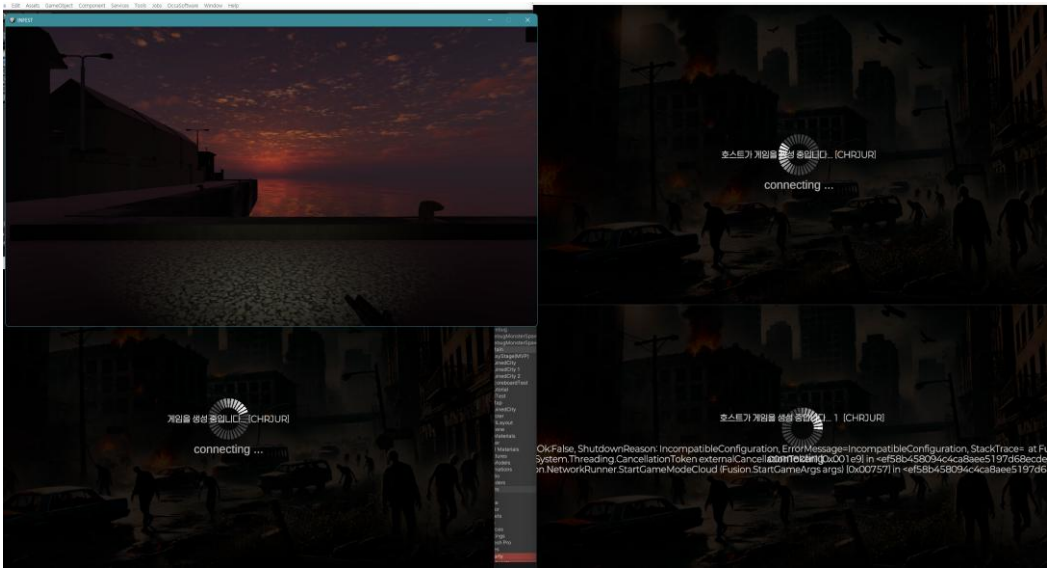


Fig. 8.3. 불안정한 권한 부여

다음 날 개발자 유승민은 이전까지 너무 달린 탓에 일이 손에 잡히지 않았다. 그래서 적당히 시간을 때울 심산으로 매칭 리팩토링을 진행했다. 매칭에서 가장 문제가 되는 부분은, Shared로 매칭하고 Host로 전환하면서 무결성을 보장하기 어렵다는 것이었다. 전환하면서 MasterClient가 나가버리고, 그 권한이 다른 Client에게 넘어가 또 다른 MasterClient가 발생하는 식이었다. 네트워크 상에서 이를 통제하는 확실한 방법이 떠오르지 않아 곤란했는데, 개발자 유승민은 RPC를 사용해 이를 해결했다.

2025.05.23 (금)

기획팀장 김성원, 개발팀장 유승민을 주도 하에 팀원 모두와 게임에 대한 정체성을 확립해 나아

가는 회의를 시작하였다. 기획적으로 오류였던 부분, 유저들이 콘텐츠에서 느꼈었으면 했던 기획적인 부분들이 주 회의 내용이었으며, 서로 놓쳤을 부분들을 확인해 나아가는 과정이었다.

기획자 김성원이 초기에 이 게임을 기획할 때, 기존 웨이브 형식의 좀비 게임과는 다르게 플레이어가 좀비 웨이브를 처치하고, 처치한 골드를 통해서 상점을 이용하여 더 좋은 무기로 교체하며 성장하여, 보스 좀비와 언제든지 전투를 진행할 수 있다는 것을 차별점으로 두어 기획을 진행하였다. 또한, 넓은 맵을 활용해 상점이 랜덤적으로 생성되면서, 플레이어로 하여금 효율적인 자원의 관리 중요성을 강조함과 동시에, 약간의 스트레스가 작용하기를 원했다.

시스템 전반적인 부분들이 랜덤적으로 이루어진 것들이 많았고, 통제 불가능한 변수가 많다는 것을 깨닫게 되었다. 그리고 플레이어로 하여금 "이 게임의 목표가 뭐지?", "내가 이 게임을 왜 플레이해야 하는 거지?" 라는 의문점을 자아낼 수 있었다. 상점도 랜덤, 추가로 기획한 미스터리 박스도 랜덤, 웨이브를 발생시키는 고장난 생체 경보기는 플레이어가 직접 확인 불가능하도록 기획하여, 플레이어가 가만히 있으면 좀비가 스폰되지 않기 때문에 위와 같은 의문점을 더 가지게 되는 요소가 되었다.

또한 웨이브를 울릴 수 있는 수단이 고장 난 생체 경보기 밖에 없는 상황에서 더욱 더 게임의 정체성이 흐려졌고, 데이터와 회의를 통해 이 망가진 시스템을 다시 다지기로 결정하였다.

배포전에 적용한 Unity Analytics를 통해서 대부분의 유저들이 2분 채 되지 않는 시간에 게임을 나가는 현상이 발행하였다. 해당 데이터와 잡은 좀비의 수를 바탕으로 유저들이 인게임에서 할 것이 없어서 많이 빠져나간 것으로 판단을 하였다.

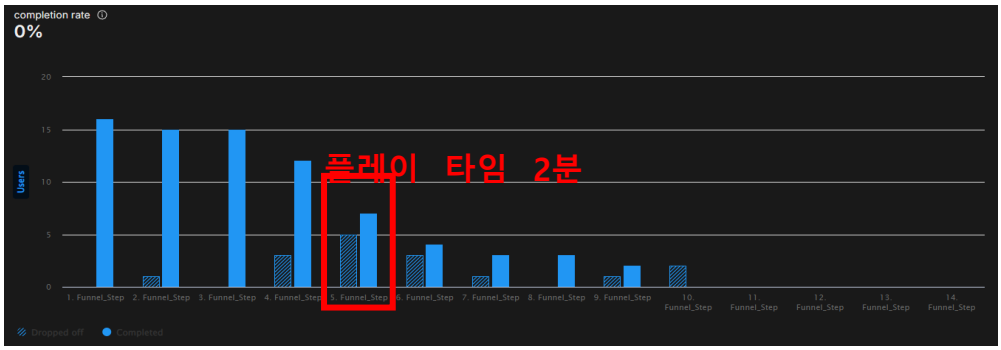
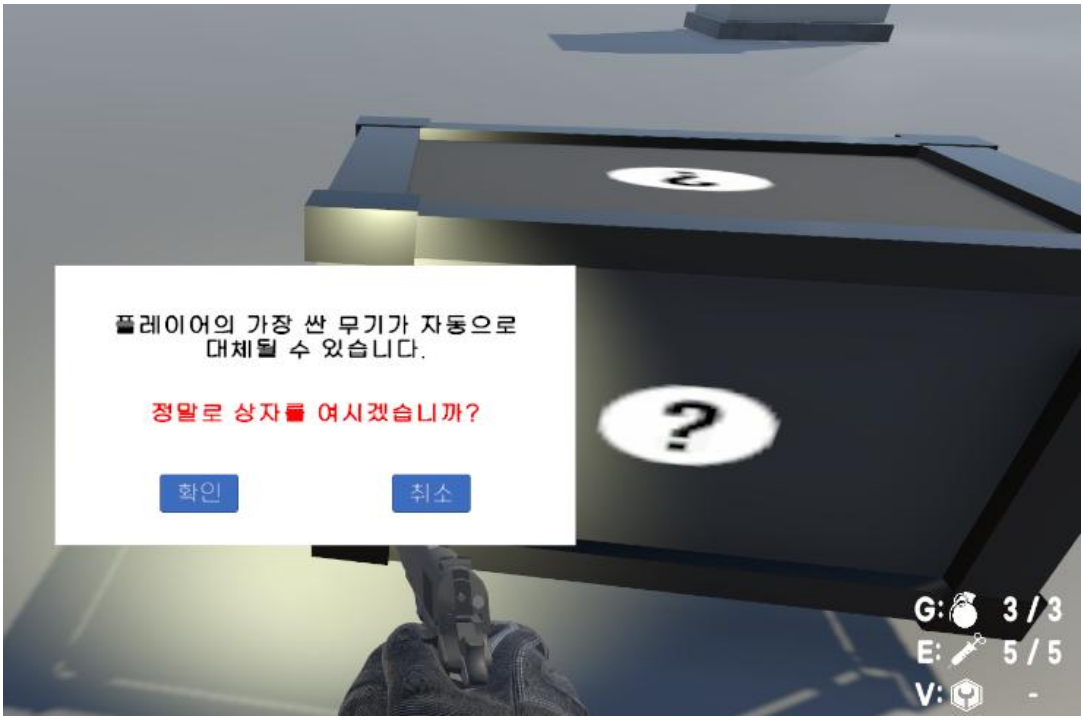


그림 1: Funnel (Window 기준)

위와 같은 상황과 불안정한 시스템을 다듬는 과정을 회의를 통해 진행하였고, 랜덤이었던 상점의 위치를 확정으로 바꾸어 나가고, 미스터리 박스는 던전 내에서의 웨이브 발생 트리거 역할로 바꿔 줌으로써 리스크와 리턴을 플레이어가 선택할 수 있도록 시스템을 재설계하였다. 또한, 이 게임의 핵심 콘텐츠 중 하나인 보스 헌트 콘텐츠의 전투 시스템을 처음부터 플레이어에게 보스를 만나게 하여, 이 게임의 주 목표를 각인시켜주는 시스템을 채용하였고, 1~3번째 전투까지는 보스를 시간대 별로 도망치게 하여, 유저가 보스를 추적한다는 느낌을 살려 기존 INFEST의 정체성을 흐리지 않게 하였다. 전투가 지속되면서 보스의 패턴을 추가하여 보스가 점점 강해지고 있는 것

을 느끼게 하였다. 인 게임 중에서 웨이브를 발생시키거나 몬스터 추가 스폰을 하는 '그리타(몬스터)'의 역할을 "몬스터 추가 스폰"만으로 줄여 경보기와 그리타의 역할을 확연하게 분리하여 기획의 모호성을 바로잡았다.



미스터리 박스

기획자 김성원은 플레이어들에게 재미를 더해주기 위해 미스터리 박스라는 뽑기 컨텐츠를 기획했다. 돈을 지불해 랜덤으로 무기를 얻을 수 있으며, 그 중에는 상점에서 판매하지 않는 RPG, 기관총과 같은 히든 무기가 포함되어 있다. 개발을 담당한 건 개발자 정수영으로, 기획자 김성원이 Blender 툴을 사용해 제작한 상자에다가 물음표 스프라이트를 붙여 오브젝트를 만들었다. 그 후, 상점과 상호작용하는 로직과 동일한 방식으로 미스터리 방식 상호작용 로직을 구현했다. 로직을 테스트하는

2025.05.26 (월)

Fig. 9.1 유저 피드백

유저 테스트 배포 후 17개의 유저 피드백이 수집되었다. 팀원들이 미처 발견하지 못한 이슈들과 개선점들을 알 수 있었고, 이를 토대로 작업을 계획하고 진행했다. 더 나아가, 유저들이 우리 게임을 하고 평가한다는 사실 그 자체에 팀원들은 격양되었다. 가장 많은 피드백이 크로스헤어의 부제였다. 이는 사실 기획자 김성원의 의도한 것이었는데, 생각보다 많은 피드백이 들어오자 김성원은 당황하며 자신의 기획을 검토하고 수정하였다.

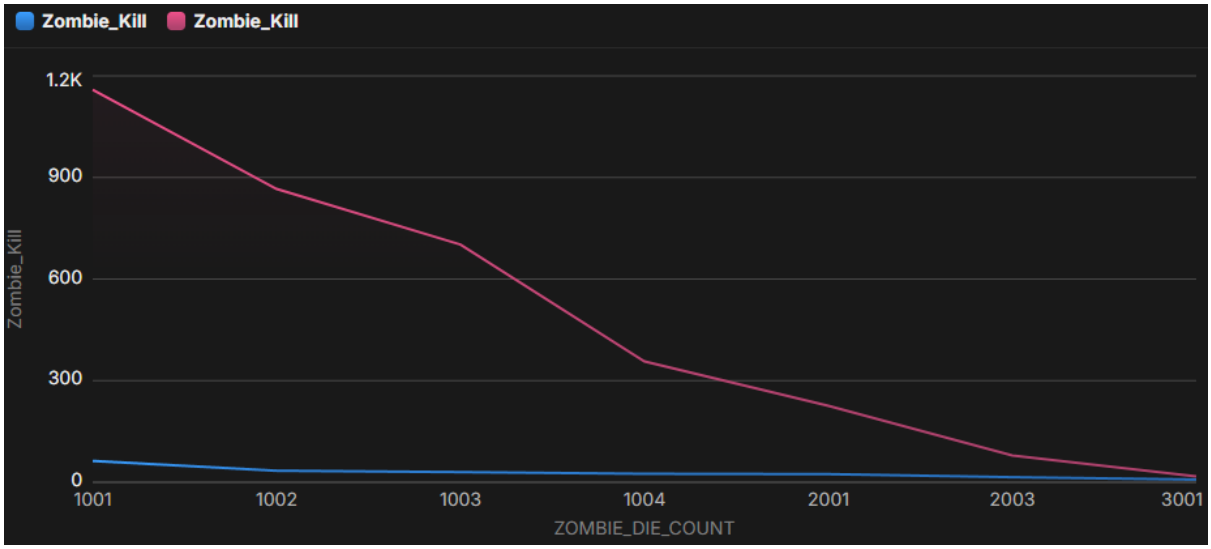


Fig. 9.2 Analytics 좀비 킬 수

게임 INFEST의 몬스터인 PJ_HI는 유저 테스트 배포 후 63명의 유저에 의해 1,159 마리가 사살되었다. 몬스터 Key 값이 높을수록 웨이브에서 소환되는 비율이 낮아지는데, 그래프에 잘 반영된 모습이다. 3001번 몬스터는 보스 몬스터 RageFang인데, 잡으라고 만든 몬스터가 아님에도 17 마리가 잡혔다.

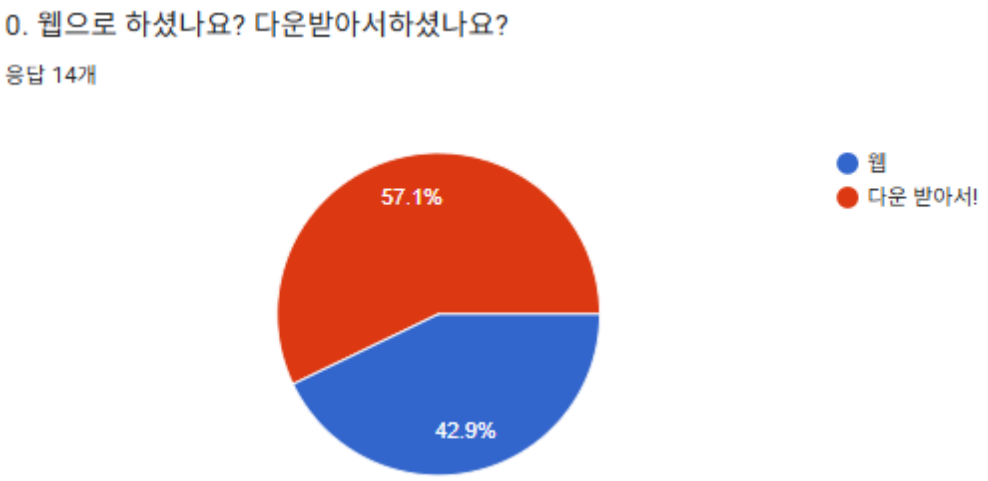


Fig. 9.3. 웹/다운 비율

웹 빌드를 강행한 것은 성공적인 판단이었다. 실제로 많은 유저들이 웹으로 플레이 했고, 이에 흥미를 느낀 몇몇 유저들이 다운 받아서 플레이 해주었다. 하지만 Unity Play에서 마우스를 꾹 누르는 입력이 먹히지 않는 탓인지 조작감이 불편하다는 얘기가 많이 들어왔다. 이를 해결하기 위해 Itch.io, Netifly, Github.page를 사용해보려 했으나 용량과 금액 문제로 포기했다.

2025.05.27 (화)

Fig. 9.3. Ragdoll

다소 심심하다고 느껴지는 몬스터 사냥을 개선하기 위해 개발자 유승민은 Ragdoll 기능을 추가했다. 이 과정에서 Ragdoll과 Animator가 겹쳐져서 Fig. 9.3과 같은 우스꽝스러운 상황이 연출되었고, 덕분에 팀 모두는 5분 동안 웃음을 참을 수 없었다. Ragdoll 기능을 추가하는데 있어 중요한 것은 동기화였다. 몬스터의 죽음은 이미 NetworkBool로 동기화가 되었지만, 수류탄과 같은 폭발에 의해 날아가는 Ragdoll을 구현하는 것은 별도의 RPC가 필요한 상황이었다. 개발자 유승민은 RPC를 최대한 안 쓰는 방향으로 기능을 구현하고 싶었다. RayCast와 데미지 적용 로직을 모든 클라이언트에서 처리하는 방식으로 Ragdoll을 컨트롤 하고 싶었지만, 시간의 한계를 마주하고 보류하게 된다.

2025.05.28 (수)

스마일게이트 멤버식 17기에 지원하기 위해 기획자 김성원과 개발자 유승민은 09:00부터 18:00까지 관련 서류 작성에 몰두했다. 서류를 작성하면서 각 항목에 대한 스마일게이트의 의도를 유추하고 그에 대한 대답을 팀끼리 토의한 후 작성하는 방식이었다. 아래는 몇 가지 답변을 가져온 것이다.

1. 가장 도움이 필요한 두 가지로 '그래픽 개선'과 팀 외적으로 교류할 수 있는 '공동체에 '가 있습니다. 최근 진행한 유저 피드백에서 그래픽 지적을 많이 볼 수 있었습니다. 피드백에 근거해 사용 중이던 에셋을 살펴봤는데, 실제로 완성도가 심하게 떨어진다는 점이 확인되었습니다. 심지어 1인칭 FPS 게임이라 상,하,좌,우 모든 면에서 조금이라도 결함이 있으면 티가 나기 때문에 그래픽 개선안을 꾀하게 되었습니다. 그러나 그래픽 문제를 디자이너도 없는 팀에서 자체적으로 해결할 순 없었습니다. 때문에 고퀄리티 에셋을 구매할 수 있는 자본이나, 그래픽 디자이너를 만날 수 있는 인력망이 필요합니다. 또한, 프로젝트가 진행될 수록 팀

이 보유하고 있는 지식과 경험의 한계가 드러나는 중입니다. 처음에는 게임 이슈들을 잘 찾아내고 해결할 수 있었으나 시간이 지날수록 네트워크 최적화, 그래픽 레더링 같은 어려운 이슈들이 놓치고 있는 추세입니다. 기획 부분에서도 똑같은 문제점을 겪고 있기

2. 개발 속도를 떨어뜨리던 기억에 남는 장애물을 키워드로 표현하자면 '소통'과 '편향'이라 할 수 있겠습니다. 소통은 비교적 흔한 문제 원인으로 작은 문제가 소통 부족에 의해 큰 문제로 번지는 일을 가리키고, 편향은 개발자들끼리 좋다고 편향되어 생각하던 것이 배포된 후 문제점으로 지적되며 시스템을 다시 설계하는 경우를 가리킵니다. 한 번은 3주 동안 개발되던 플레이어 로직이 결과물을 보니 엉성하고 네트워크 동기화도 안 되던 일이 있었습니다. 우선 팀원 대부분이 비전공자였다는 점, 이로 인해 스스로 작업 진행도를 평가할 수 없다는 점, 거기에 3주 동안 팀원이 서로 작업물을 공유하지 않았기 때문에 생겼던 문제였습니다. 이 일이 있던 후로는 매일 아침 스크럼을 진행하며 각자의 진행사항을 공유하고, 주에 한 번 코드 리뷰 시간을 보내며 서로 부족한 점을 보완했고 덕분에 다시는 이런 일이 발생하지 않았습니다. '편향'과 관련된 문제가 발생한 것은 유저 피드백을 수집한 직후였습니다. 개발진들이 좋다고 생각했던 시스템과 기획들이 사실은 유저들로 하여금 오히려 불편함을 야기한다는 것을 깨달았습니다. 유저의 관점에서 게임을 관철하지 않았다는 것이 문제의 원인으로 지목되었고, 최대한 많은 팀 외적인 피드백을 수집하여 기획을 수정하고 시스템을 재설계하여 해결하고 있습니다.

약 18:30경에 스파르타 부트캠프에서 최종 프로젝트 전시회장을 열었는데, 그 중 딱 하나 샘플로 전시된 프로젝트가 우리 조 프로젝트였다. 가장 먼저 브로셔를 제출한 조를 올린 것인지, 아니면 가장 잘 만든 조를 올린 것인지는 모른다. 팀원들은 전자일 것이라고 추측 중이다.

2025.05.29 (목)

09:00, 평소처럼 팀원 모두가 한 자리에 모였다. 최종 발표까지 영업일로 이틀 남은 시점에서 약간 어수선한 분위기가 형성되었다. 스마일 게이트 멤버십 17기 선정이 불확실한 상황에서 팀원은 각자의 미래를 꿈꾸었다. 개발자 임성균은 모 회사에 지원하여 면접을 앞두고 있다. 개발자 정수영은 스마일 게이트 멤버십 17기에 선정된다 한들 적극적인 작업이 힘들 것임을 넌지시 언급했다. 기획자 김성원은 개발자들의 의지에 따르겠다고 하였다. 또한 개발자 유승민은 멤버십에 선정되어도 구직 활동을 병행하겠다 했으며, 개발자 주정민은 별 다른 얘기가 없었다. 09:30, 팀원들이 서로 소소한 잡답을 나눈 후 개발자 유승민의 주도 하에 스크럼을 진행했다. 유저 테스트 배포 이후 스크럼 진행 능력, 업무 퍼포먼스가 떨어졌던 개발자 유승민은 컨디션을 회복한 듯 이전과 같은 모습을 보였다. 스크럼 결과 금주 금, 차주 월요일 두 번에 걸쳐 최종 빌드를 진행하기로 하였다. 그 때까지 개발자 유승민은 몬스터 리팩토링과 레벨 디자인을 하고, 개발자 임성균은 플레이어 추가 동작을 구현하기로 했다. 개발자 주정민은 전체적인 사운드와 UI를 점검하고, 개발자

정수영은 최적화 작업을 진행하며 작업을 마무리하기로 했다.

개발자 정수영은 Occlusion Culling을 이용해 3만에 가까이 나오던 Batch를 만 이하로 낮추는데 성공했다. 하지만 여전히 목표 batch인 1500에는 한참 멀었기 때문에 계속해서 R&D를 진행하기로 했다. 개발자 유승민은 수정된 기획서에 맞춰 보스 로직을 다시 작성했다. 플레이어와 전투를 시작하고

개발자 정수영은 오쿨러전 컬링, 가시거리 설정, 수류탄 유지 애니메이션, 몬스터 헤드샷이 안 맏는 이유가 히트박스 radius를 밝혀내며 연구소장이라는 별명을 얻었다.

개발자 유승민은 보스 헌트 바뀐 기획서에 맞게 보스가 도망가는 로직을 작성했다. 보스가 도망갈 때, 영화 헐크의 점프처럼 그 지역으로 날라간다. 그 후 방패를 설치했을 때 몬스터가 우회하게 만들었고 길이 없을시 방패를 때리게 만들었다. 부활했을 때 다시 몬스터의 타격이 될 수 있도록 Resapw시 Collider를 껐다 켜서 OnTriggerEnter를 다시 작동시켰다. UI, 사운드, 사용자 경험, 부활 로직

기획자 김성원의 브로셔에 각자 팀원이 맡은 역할을 적어달라 요청하였다. 개발자들은 각자 생각할 수 있는 모든 농담거리를 생각해내며 즐거운 시간을 보냈다. 다만 이에 대한 책임이 있던 기획자 김성원은