혼자 만든 메타버스
리브아일랜드 개발 이야기

강해성
e-mail: koh1260@naver.com

픽셀 감성 메타버스 서비스

livisland.com

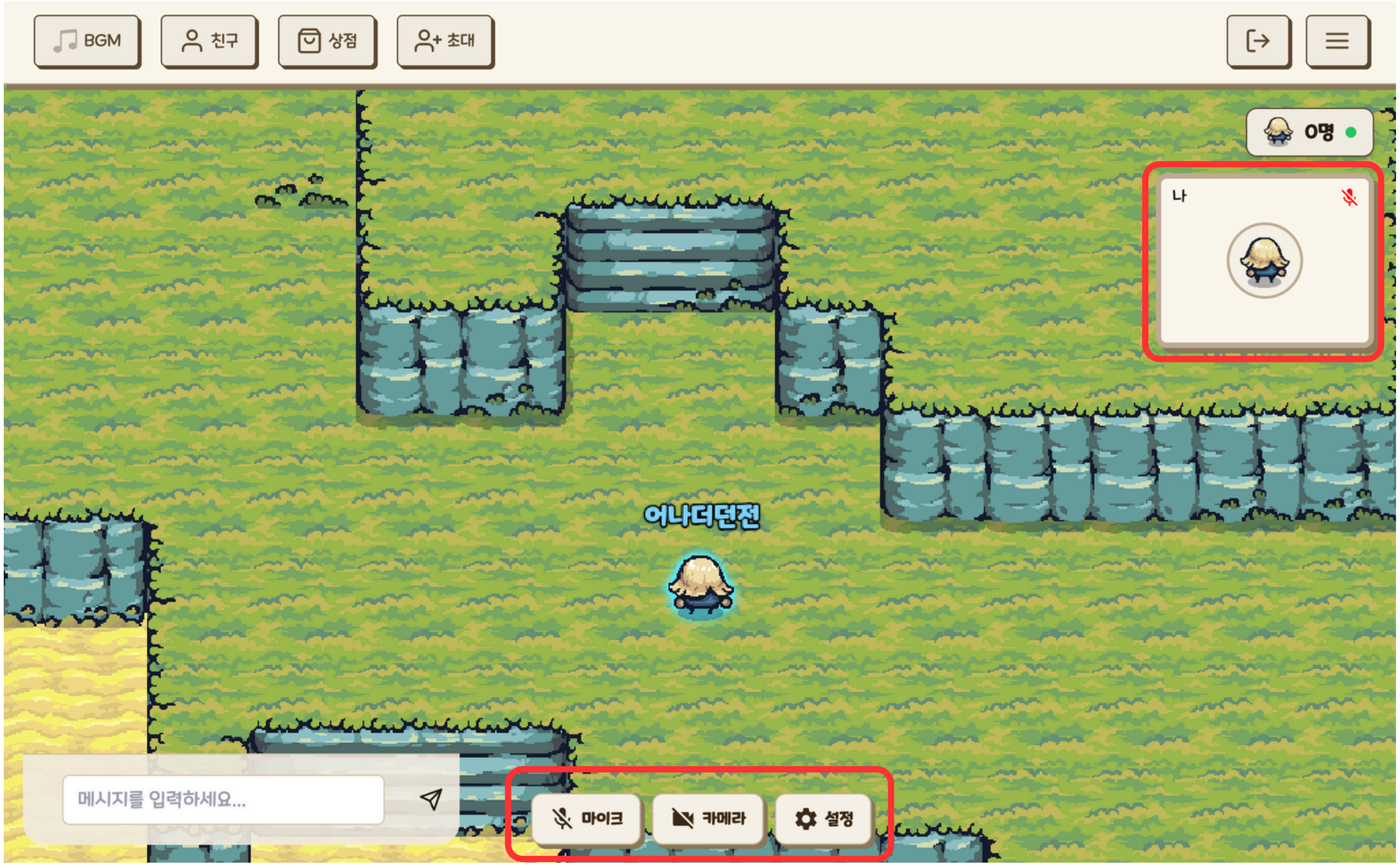
주요 기능

- 무인도
- 비밀섬

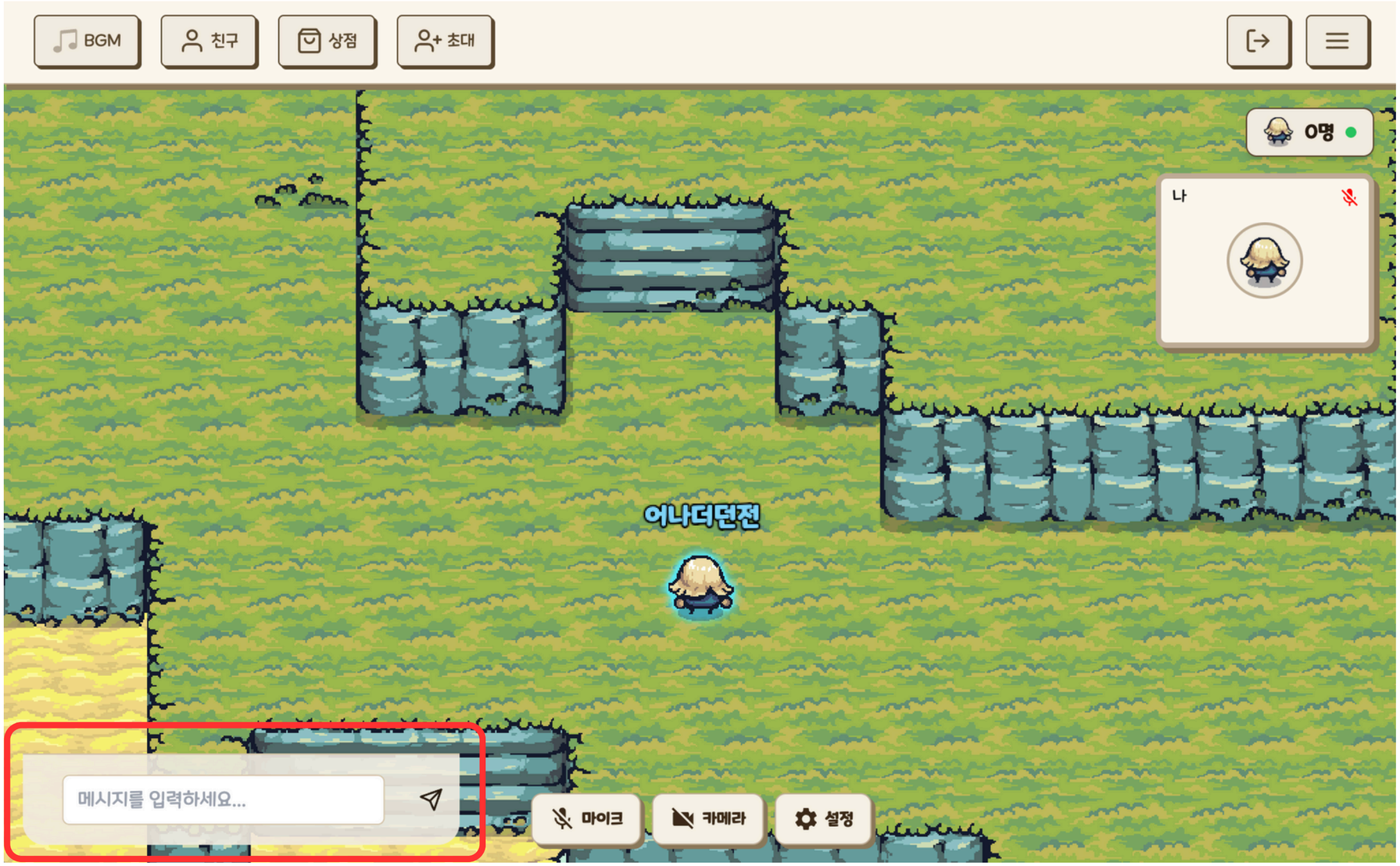
- 텍스트 채팅
- 음성 채팅
- 화상 채팅

- 상점
- 꾸미기 아이템

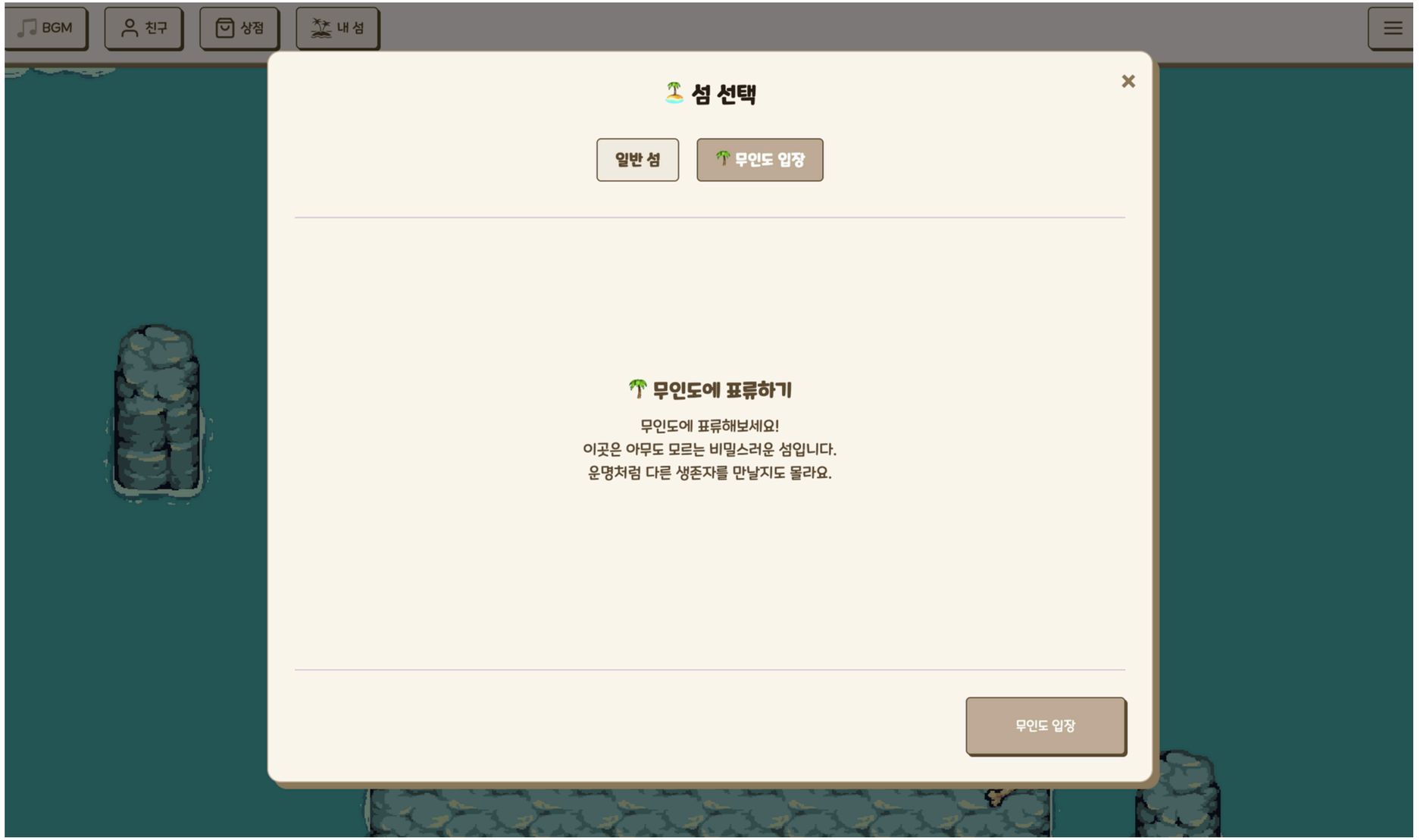
주요 기능



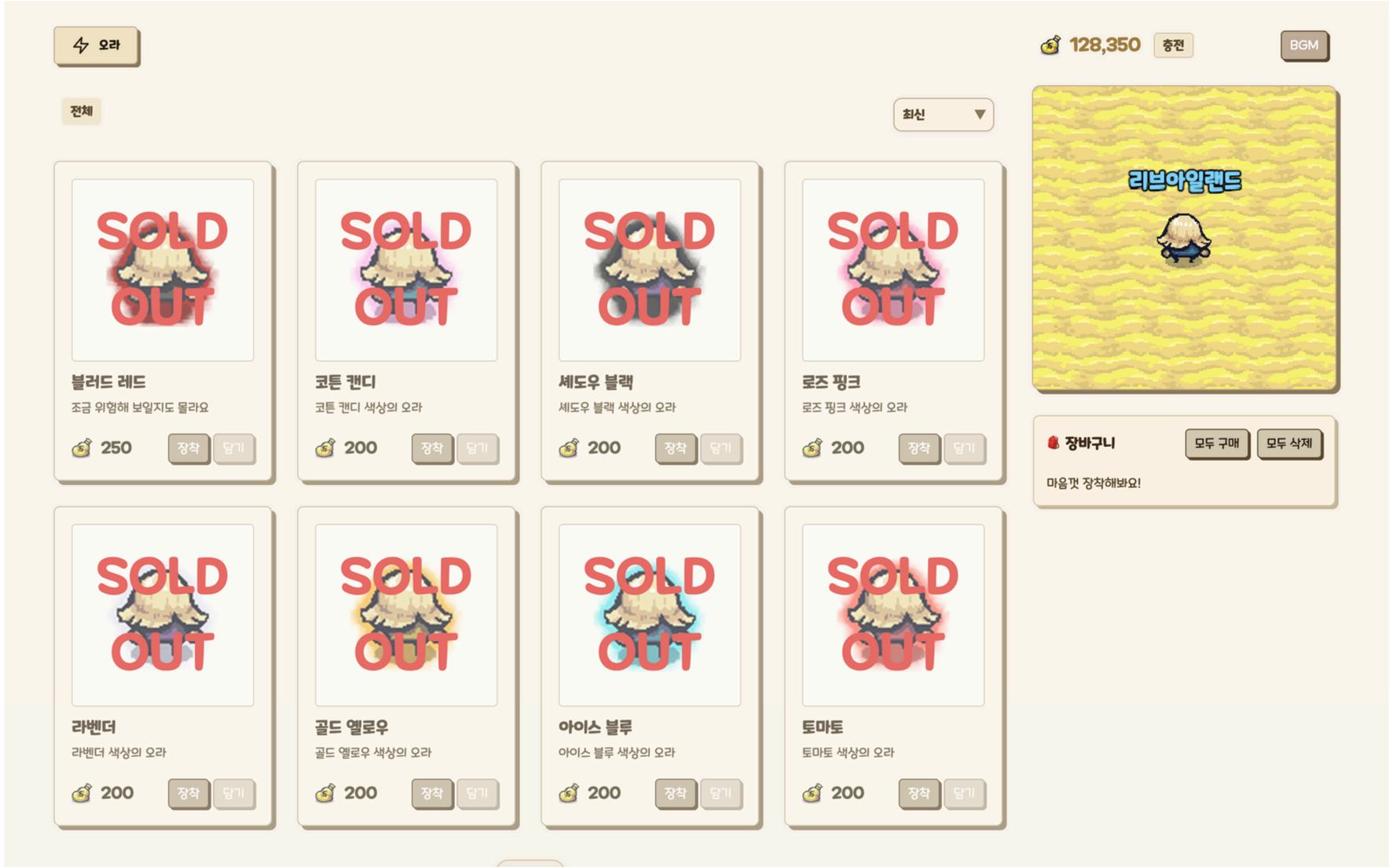
주요 기능



주요 기능


BGM
친구
상점
내 섬
섬 선택
일반 섬
무인도 입장
무인도에 표류하기
무인도에 표류해보세요!
이곳은 아무도 모르는 비밀스러운 섬입니다.
운명처럼 다른 생존자를 만날지도 몰라요.
무인도 입장

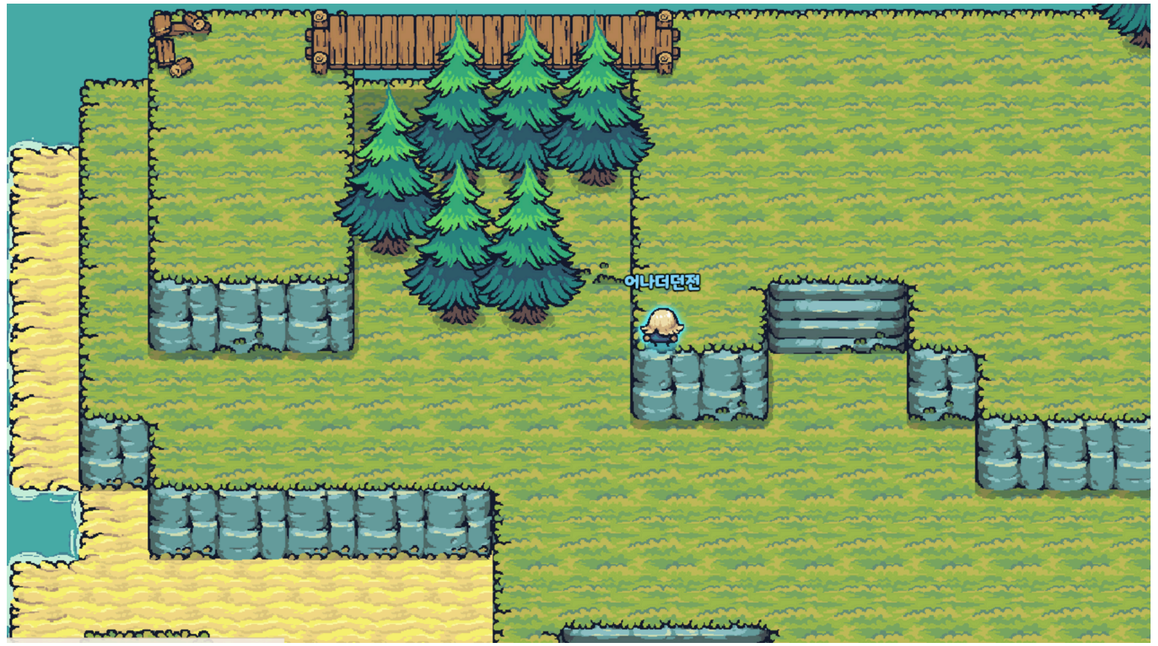
주요 기능


오라
128,350
충전
BGM
전체
최신
SOLD OUT
블러드 레드
조금 위험해 보일지도 몰라요
250
장착
담기
코튼 캔디
코튼 캔디 색상의 오라
200
장착
담기
셰도우 블랙
셰도우 블랙 색상의 오라
200
장착
담기
로즈 핑크
로즈 핑크 색상의 오라
200
장착
담기
라벤더
라벤더 색상의 오라
200
장착
담기
골드 옐로우
골드 옐로우 색상의 오라
200
장착
담기
아이스 블루
아이스 블루 색상의 오라
200
장착
담기
토마토
토마토 색상의 오라
200
장착
담기
리브아일랜드
장바구니
모두 구매
모두 삭제
마음껏 장착해봐요!

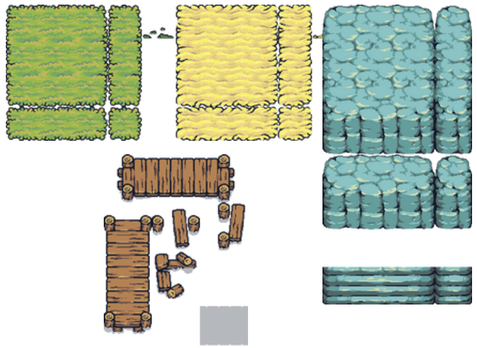
너무나 많았던 작업..

- 기획
- UI 디자인
- 맵 제작
- 상품 제작
- 사업자 등록
- 홍보
- 결제 구현
- 서버
- 디비
- 인프라

개발 외의 작업들



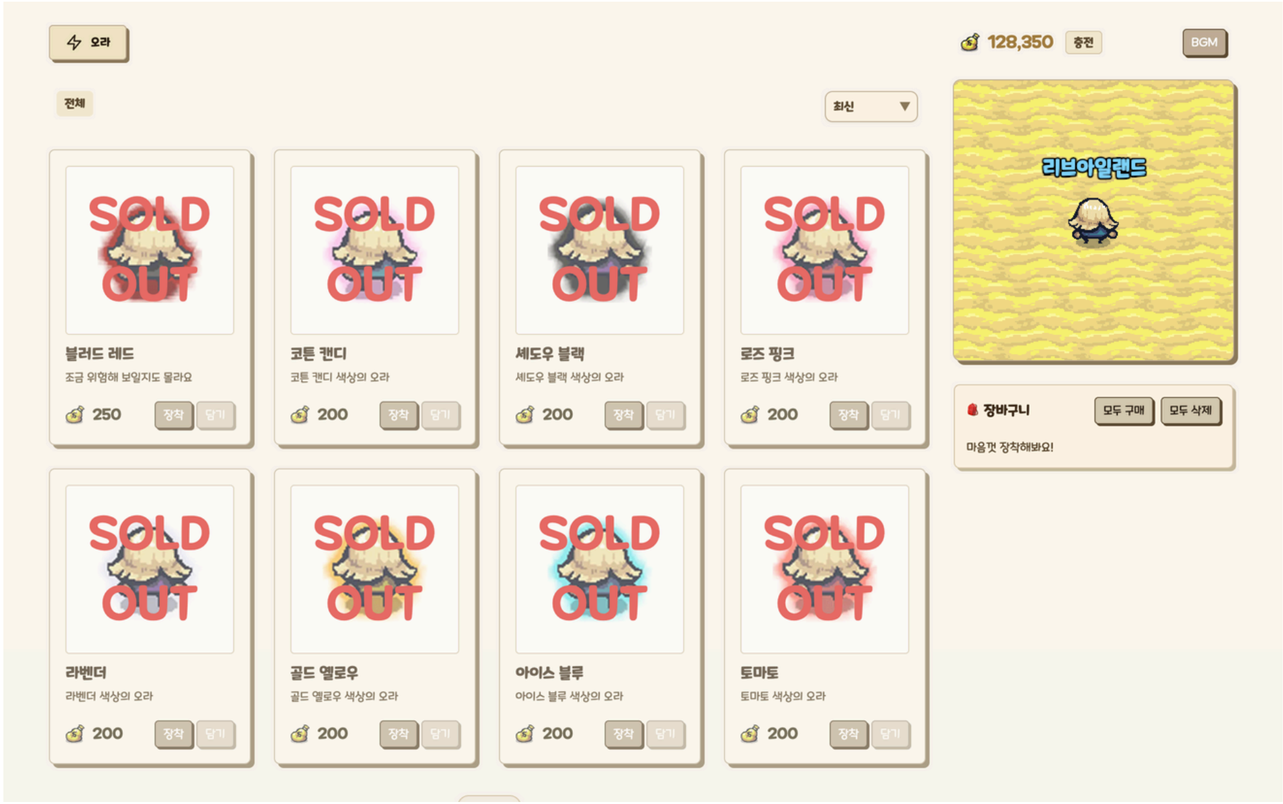
맵 만들기



개발 외의 작업들

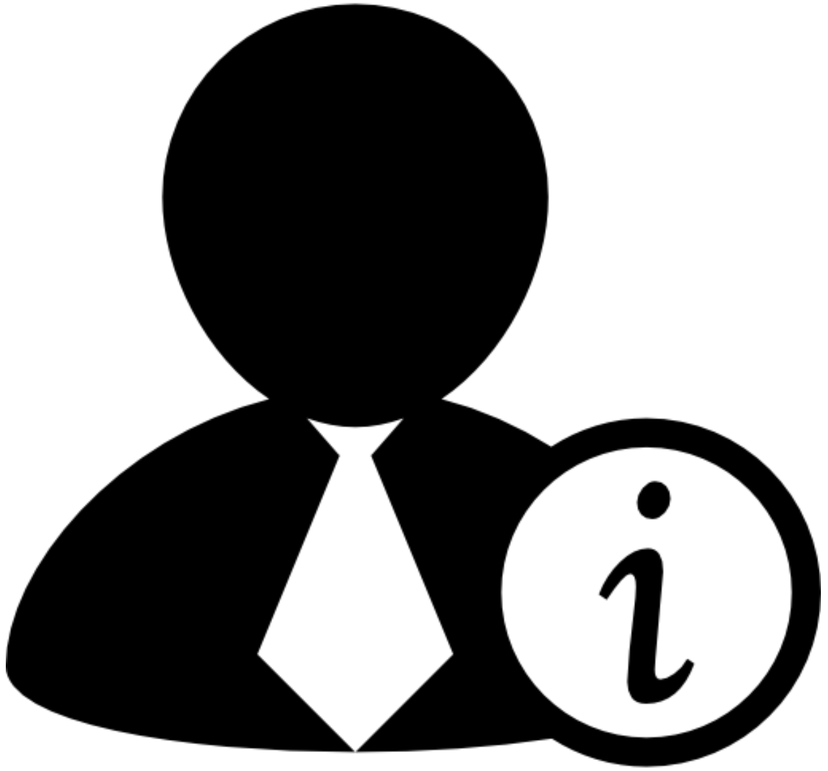
도트도 찍고

개발 외의 작업들



디자인도 하고

개발 외의 작업들



사업자 등록

개발 외의 작업들

광고 영상 제작

개발 외의 작업들

PG사 계약

메타버스 서비스 주요 특징

게임적 요소

캐릭터·애니메이션·월드·충돌

음성 화상 채팅

실시간

게임적 요소

JS로 구현된 HTML5 게임 엔진!

게임적 요소

Unity

커다란 유니티

강력하지만 작고 귀여운 Phaser

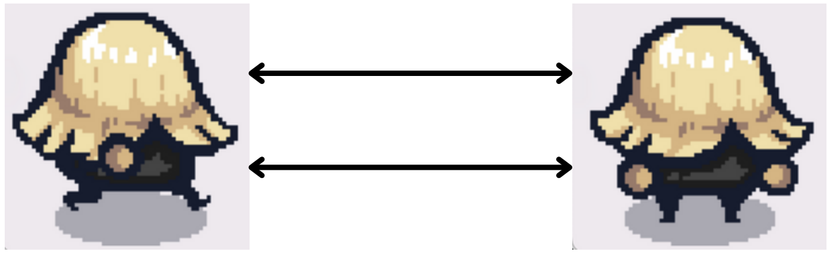
게임적 요소

컴포넌트 패턴

말풍선
파티클
장비
이름 라벨
네트워크

이름 라벨

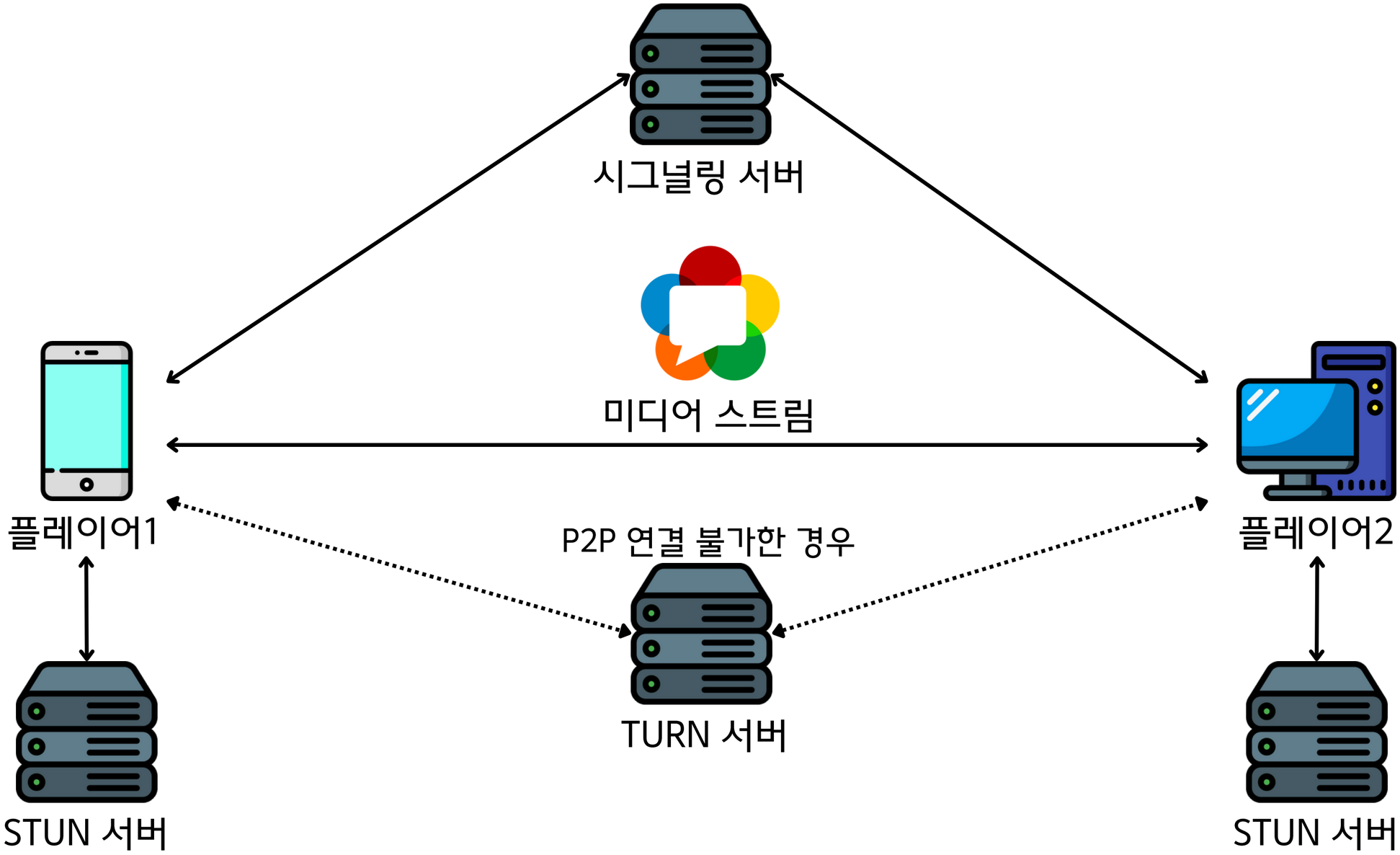
FSM



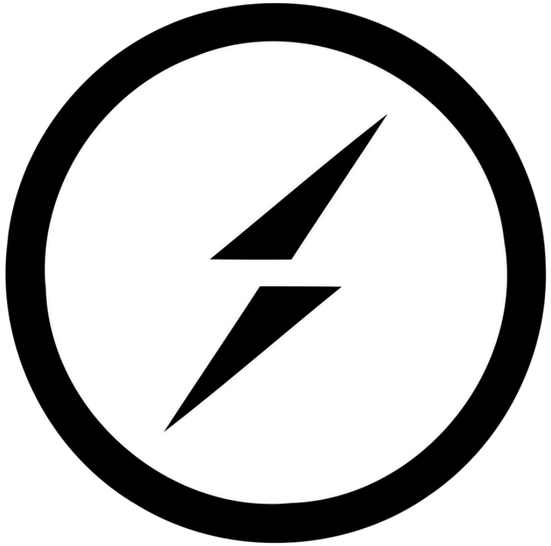
음성 화상 채팅

WebRTC

음성 화상 채팅


시그널링 서버
미디어 스트림
플레이어1
플레이어2
P2P 연결 불가한 경우
TURN 서버
STUN 서버
STUN 서버

실시간



SocketIO

플레이어 좌표 동기화

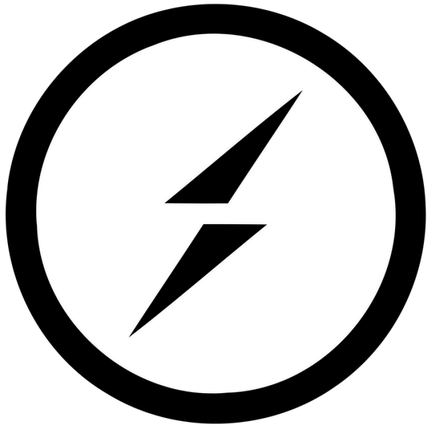
공격

TS로 대통합!

Next.js

Phaser.js

Nest.js



Socket.IO

TS 최고!

동시성

최대 인원: 3

현재 인원: 2

동시성

동시성

1. 섬의 현재 인원 조회
2. 최대 인원과 비교
    a. 가득 찬 경우 예외
3. 입장

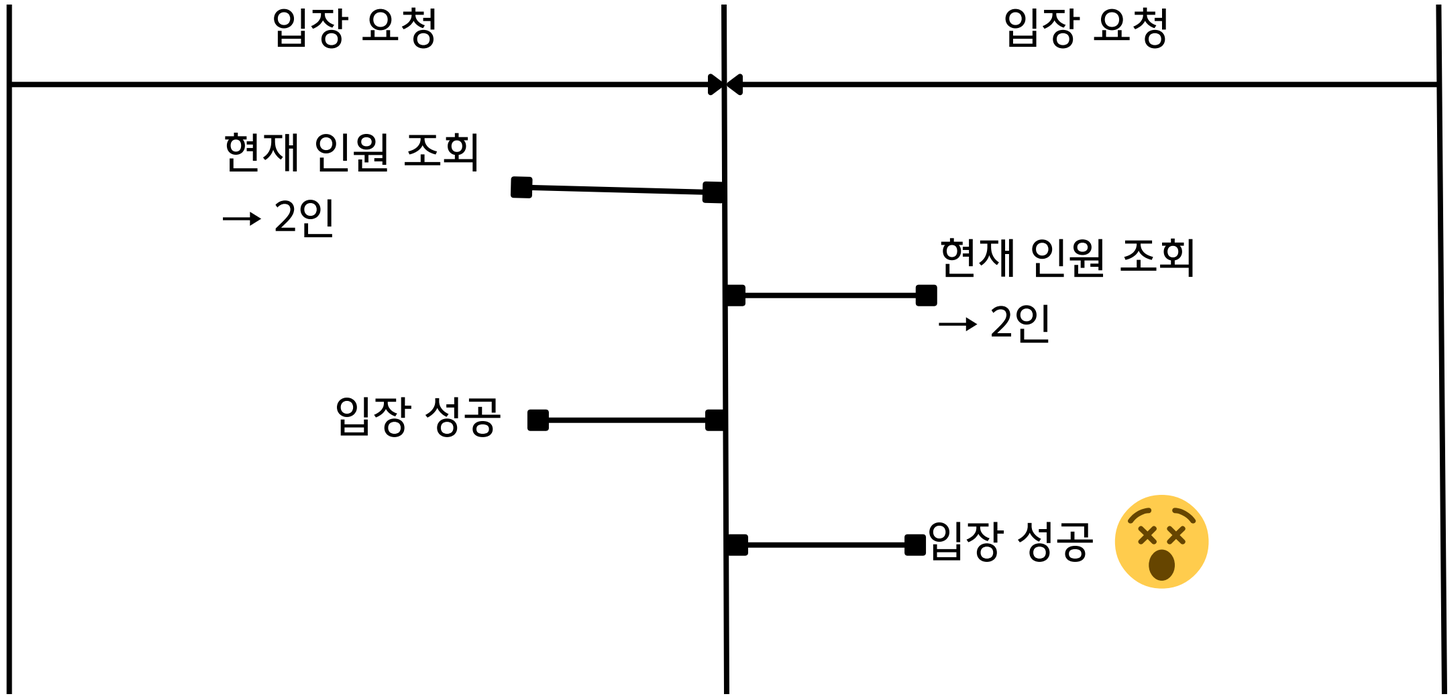
동시성

1. 섬의 현재 인원 조회
2. 최대 인원과 비교
    a. 가득 찬 경우 예외
3. 입장

Redis 비동기 I/O

동시성


들어갈래~
나도 들어갈래~
최대 인원: 3
현재 인원: 2


입장 요청
입장 요청
현재 인원 조회
→ 2인
현재 인원 조회
→ 2인
입장 성공
입장 성공

동시성

비관적 락

동시성 – Redis Lock

스핀 방식	Pub Sub
계속 재시도	대기

동시성 – Redis Lock (스핀)

```
private async acquireWithRetry(
        key: string,
        ttl: number,
        maxRetries: number,
        retryDelay: number,
    ): Promise<boolean> {
        let attempt = 0;

        while (attempt ⩽ maxRetries) {
            const lockAcquired = await this.redis.acquireLock(key, ttl);
            if (lockAcquired) return true;

            if (attempt === maxRetries) break;

            await this.delay(retryDelay);
            attempt++;
        }

        throw new DomainException(
            DomainExceptionType.LOCK_ACQUIRED_FAILED,
            HttpStatus.CONFLICT,
            LOCK_ACQUIRED_FAILED_MESSAGE(key),
        );
    }
```

동시성 – Redis Lock (pub sub)

공정성에 대한 고민

내가 먼저야

내가 먼저거든?

동시성 – Redis Lock (pub sub)

지킨다면 Queue가 필요하다

동시성 – Redis Lock (pub sub)

관리 포인트 및 리소스 증가 성능 저하

줄 똑바로 서시고 가실 분은 가세요~

동시성 - Redis Lock (pub sub)

내 서비스에서는 공정성이 중요할까

동시성 – Redis Lock (pub sub)

- 적은 참여 인원
- 참여 대기가 아닌 성공 실패
- 사용자들은 알 수 없음

동시성 – Redis Lock (pub sub)

공정성 < 성능 및 리소스 관리

동시성 – Redis Lock (pub sub)

```
export class RedisTransactionManager implements OnModuleDestroy {
    private subscribedChannels = new Set<string>();
    private waitingContexts = new Map<string, Set<() => void>>();

    constructor(private readonly redis: RedisClientService) {
        // 메시지 이벤트 등록
        this.redis.subscriber.on(
            'message',
            (channel: string, message: string) => {
                this.handleChannelMessage(channel, message);
            },
        );
    }

    // 다른 코드들 . . .
}
```

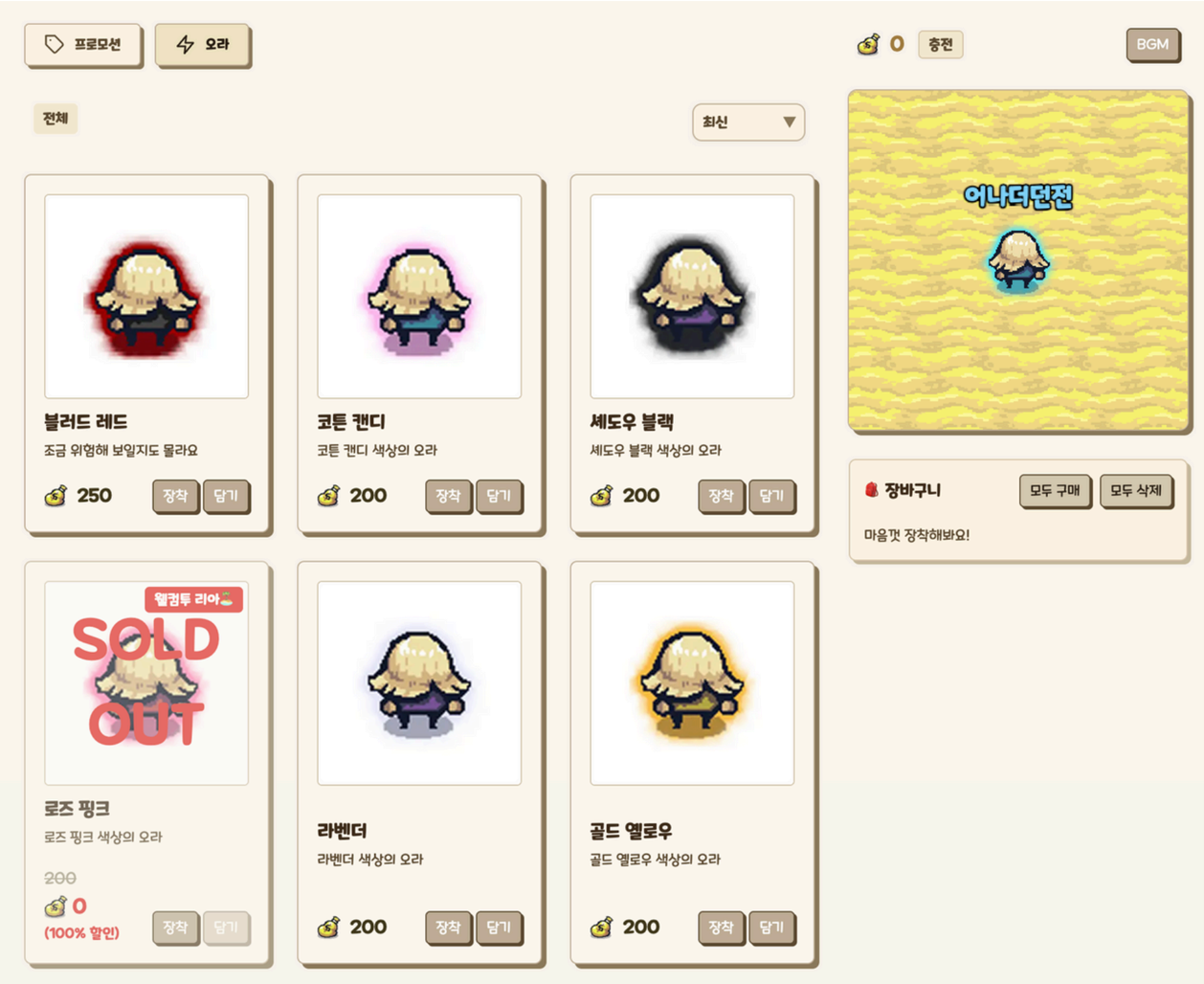
동시성 – Redis Lock (pub sub)

```
private async acquireLockWithPubSub(
        lockKey: string,
        ttl: number,
        maxRetries: number,
        timeout: number,
    ): Promise<void> {
        const channel = `${lockKey}:release`;

        // 채널 구독 (한 번만)
        await this.ensureChannelSubscription(channel);

        let attempt = 0;

        while (attempt <= maxRetries) {
            // 락 획득 시도
            const lockId = await this.redis.acquireLock(lockKey, ttl);
            if (lockId) {
                return; // 락 획득 성공
            }

            // 마지막 시도라면 더 이상 대기하지 않음
            if (attempt === maxRetries) {
                break;
            }

            // 다음 시도를 위해 해제 신호 대기
            await this.waitForRelease(channel, timeout);
            attempt++;
        }

        throw new DomainException(
            DomainExceptionType.LOCK_ACQUIRED_FAILED,
            HttpStatus.CONFLICT,
            LOCK_ACQUIRED_FAILED_MESSAGE(lockKey),
        );
    }
```

동시성 – Redis Lock (pub sub)

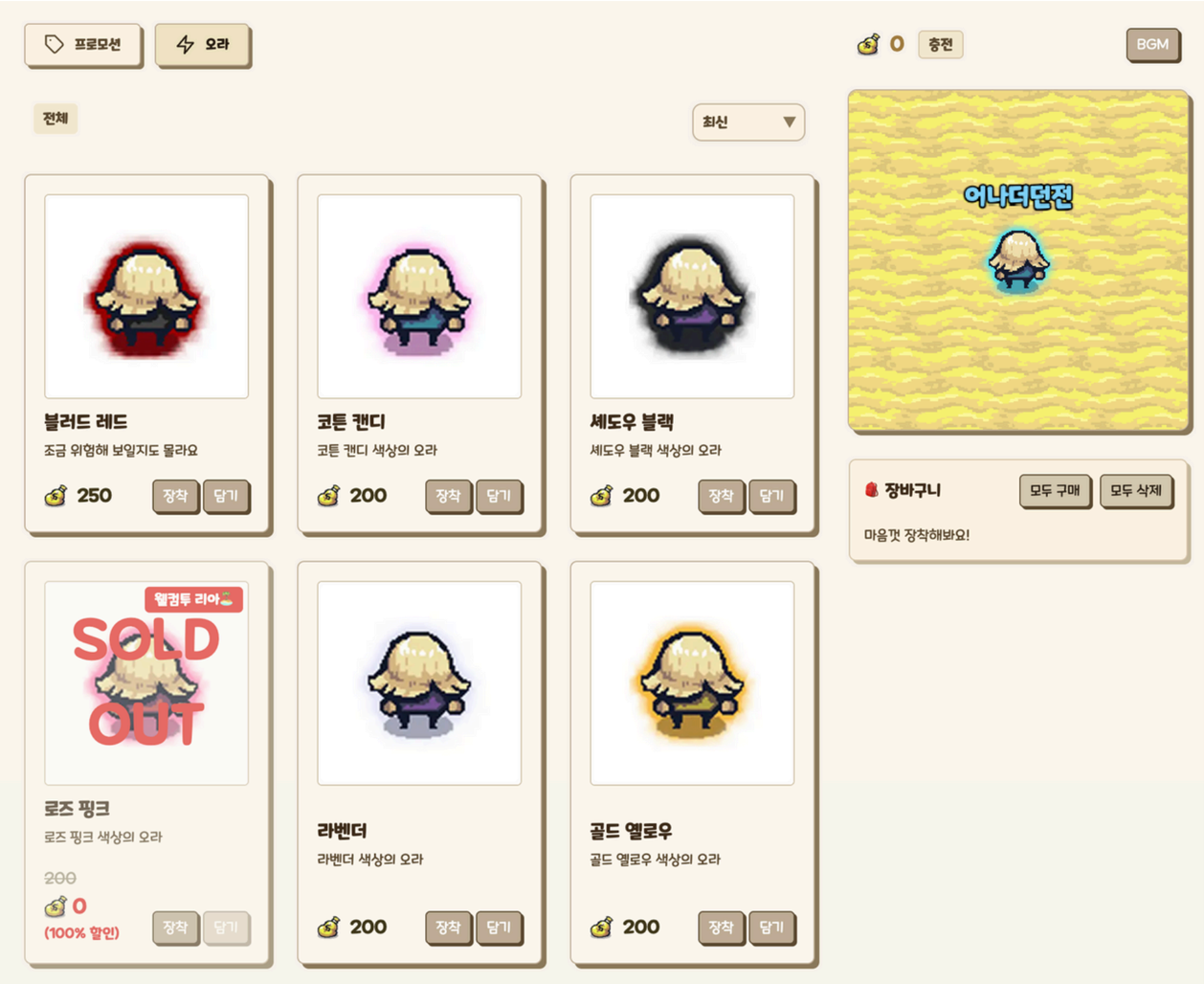
```
private async waitForRelease(
        channel: string,
        timeout: number,
    ): Promise<void> {
        return new Promise((resolve, reject) => {
            const wrappedResolver = () => {
                clearTimeout(timeoutId);
                resolve();
            };

            const timeoutId = setTimeout(() => {
                // 타임아웃 시 대기 목록에서 제거
                const waitingSet = this.waitingContexts.get(channel);
                if (waitingSet) {
                    waitingSet.delete(wrappedResolver);
                }

                reject(
                    new DomainException(
                        DomainExceptionType.LOCK_ACQUIRED_FAILED,
                        HttpStatus.REQUEST_TIMEOUT,
                        `Lock wait timeout for channel: ${channel}`,

),
                    );
            }, timeout);

            // 이 컨텍스트를 대기 목록에 추가
            const waitingSet = this.waitingContexts.get(channel);
            if (waitingSet) {
                waitingSet.add(wrappedResolver);
            }
        });
    }
```

동시성



적은 골드로 전부 사버려야지 크크

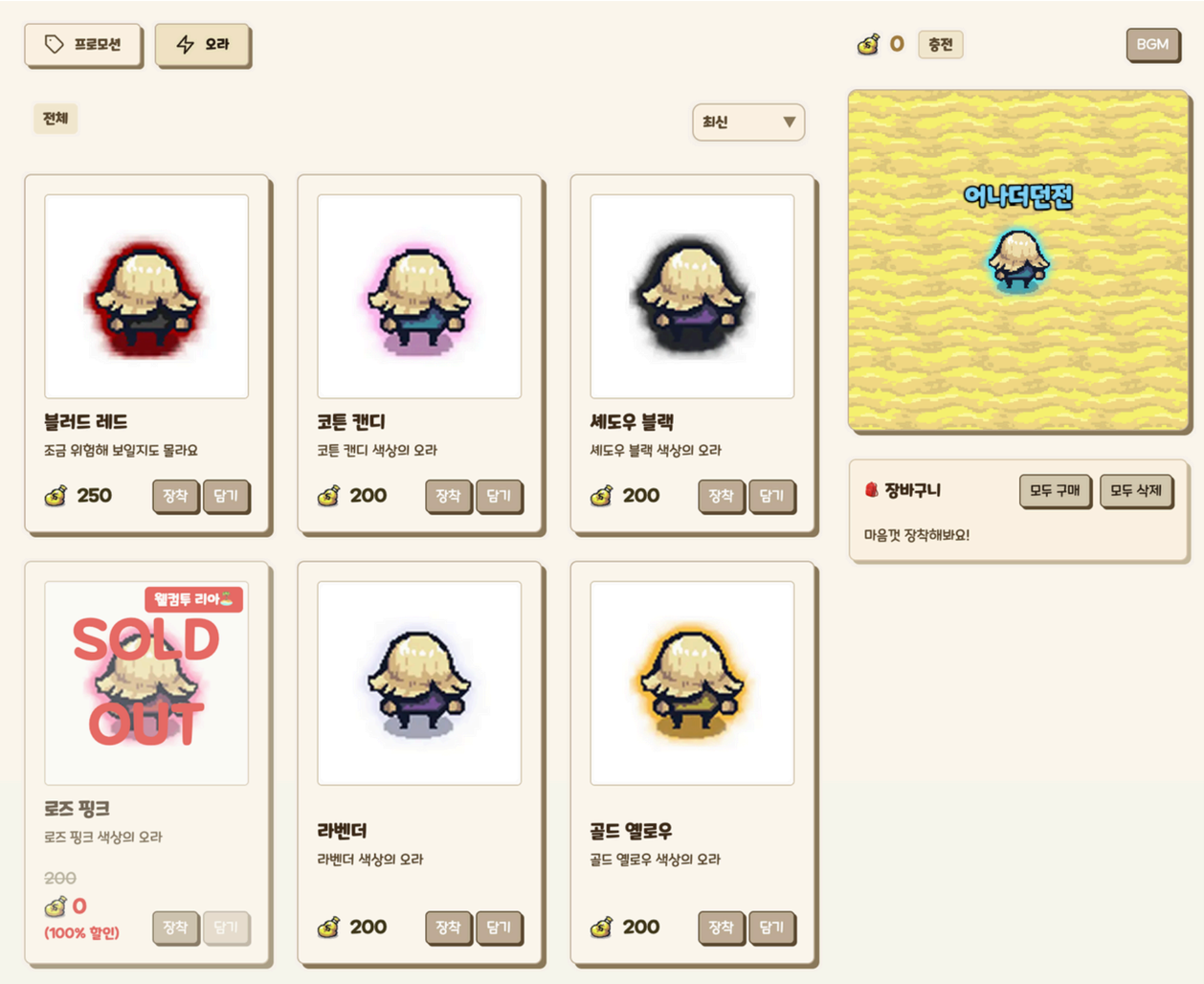
동시성



적은 골드로 전부 사버려야지 크크

Redis Lock 🔒

동시성



적은 골드로 전부 사버려야지 크크

Redis Lock 🔒

안돼..

동시성

하나의 오브젝트에 대한 동시 공격

Lua script?

동시성

공격

→ 같은 섬의 플레이어들 간에 발생

→ 싱글 스레드 메모리 레벨에서 해결

방향성

- 결제 심사
- 광고
- 놀이, 작업 공간 다방면 호환
- 미니게임
- 여러 상호작용 요소
- 재화 추가

감사합니다!