게임 구조 및 주요 로직 설명

본 문서는 bossGame.js, brickGame.js, 그리고 이들이 상속받는 gameManager.js의 주요 구조와 핵심 로직에 대한 사용자의 이해를 돕기 위해 정리한 문서입니다.

---

1. 전체 구조 개요

- **GameManager**
  - 게임의 공통 기능(입력 처리, 게임 루프, UI, 포켓몬 슬롯/체력/능력 시스템 등)을 담당하는 추상 클래스입니다.
  - 벽돌깨기(BrickGame), 보스전(BossGame) 모두 이 클래스를 상속받아 각 게임별 특화 기능을 구현합니다.
- **BrickGame**
  - 벽돌깨기(조합 시스템, 포켓몬/아이템 블록, 동적 벽돌 생성 등) 전용 기능을 GameManager에서 확장합니다.
  - 포켓몬 능력(풀/불/전기/물/얼음 타입) 및 각종 아이템, 조합 완성 보너스 등 벽돌깨기 특화 로직이 포함됩니다.
- **BossGame**
  - 보스전(탄막, 보스 페이즈, 순간이동, 레이저 등) 전용 기능을 GameManager에서 확장합니다.
  - WASD/방향키 기반 이동, 자동 총알 발사, 보스 페이즈 전환, 다양한 공격 패턴, 보스 이미지/애니메이션 관리 등 보스전 특화 로직이 포함됩니다.

---

2. GameManager 주요 로직

- **입력 처리**
  - 키보드/마우스 입력을 바인딩 및 해제하며, 슬롯 전환/포켓몬 능력 사용/일시정지 등 공통 조작을 처리합니다.
- **게임 루프**
  - update()에서 프레임 단위로 게임 상태를 갱신하며, 각 게임별 updateGame()을 호출합니다.
  - 일시정지, 게임 종료, 시간 제한, UI 업데이트 등 공통 흐름을 관리합니다.
- **포켓몬 슬롯/체력/능력 시스템**
  - 슬롯별 포켓몬 관리, 체력 소모/회복, 기절 상태, 능력 쿨타임, 능력 발동 등 포켓몬 관련 공통 기능을 제공합니다.
- **UI 및 메시지**
  - 생명, 점수, 체력바, 인게임 메시지 등 공통 UI 요소를 그립니다.

---

3. BrickGame 주요 로직

- **조합(Combination) 시스템**
  - 다양한 패턴의 벽돌 조합을 동적으로 생성, 이동, 충돌 처리합니다.
  - 각 조합에는 포켓몬 블록, 아이템 블록이 포함될 수 있습니다.
- **포켓몬/아이템 블록**
  - 포켓몬 블록: 구출 시 슬롯에 추가, 점수 획득, 능력 사용 가능
  - 아이템 블록: 슬롯 포켓몬 체력 회복 등 효과 적용
- **포켓몬 능력**
  - 풀: 생명력 회복

    - 불: 공 속도 증가
    - 전기: 점수 2배
    - 물: 패들 크기 증가
    - 얼음: 조합 이동 속도 감소
    - 각 능력은 쿨타임, 지속시간, 사운드 효과와 함께 동작

- **승리/패배 조건**
    - 스토리 모드, 점수 모드: 시간 종료까지 진행

---

4. BossGame 주요 로직

- **플레이어 조작**
    - WASD/방향키로 2D 평면 이동(회전/가속), 자동 총알 발사
- **보스 패턴**
    - 페이즈 1: 일반 이동 + 전방향 탄막
    - 페이즈 2: 순간이동, 레이저, 플레이어 조준 탄막 등 패턴 강화
    - 체력 30% 이하에서 페이즈 전환(이미지, 메시지, 연출 포함)
- **보스 이미지/애니메이션**
    - 상태(일반/피격/공격)에 따라 이미지 교체, 애니메이션 타이밍 관리
- **충돌 및 체력 관리**
    - 플레이어 총알-보스, 보스 탄막/레이저-플레이어 충돌 처리
    - 보스/플레이어 체력바, 게임 종료(클리어/실패) 처리

---

5. BrickGame 주요 메서드 및 로직 상세

- **createNewCombination()**
    - 벽돌 조합(패턴)을 동적으로 생성합니다.
    - 각 조합은 포켓몬 블록, 아이템 블록 등으로 구성되며, 패턴은 스테이지별로 다양하게 바뀝니다.
    - 생성된 조합은 화면에 등장해 오른쪽으로 이동합니다.
- **dynamicCollisionDetection()**
    - 공과 각 조합 내 벽돌의 충돌을 실시간으로 감지합니다.
    - 충돌 시 벽돌이 파괴되고, 포켓몬/아이템 효과가 즉시 적용됩니다.
- **포켓몬 능력 실행 (executePokemonAbility)**
    - 풀타입: 생명력 회복 (최대 생명력 제한)
    - 불타입: 공 속도 증가 (5초간, 중복 방지 및 일시정지 대응)
    - 전기타입: 점수 2배 (8초간)
    - 물타입: 패들 크기 증가 (7초간)
    - 얼음타입: 조합 이동 속도 감소 (6초간)
    - 각 능력은 사운드 효과와 쿨타임, 상태 관리가 포함됨
- **updateGame()**
    - 공 이동, 벽돌/조합 이동, 충돌 처리, 승리/패배 조건 체크, UI 갱신 등 벽돌깨기 게임의 메인 루프를 담당합니다.

---

6. BossGame 주요 메서드 및 로직 상세

- **updatePlayer()**

    - WASD/방향키 입력에 따라 플레이어의 위치, 회전, 가속도를 갱신합니다.
    - 마찰력, 최대 속도, 회전 속도 등도 반영됩니다.

- **shootPlayerBullet()**
    - 플레이어가 일정 간격으로 총알을 발사합니다.
    - 총알은 플레이어의 회전 방향을 따라 이동합니다.

- **updateBoss()**
    - 보스의 현재 페이즈에 따라 이동/공격 패턴을 갱신합니다.
    - 체력, 페이즈 전환, 애니메이션 상태 등도 관리합니다.

- **updatePhase1() / updatePhase2()**
    - 페이즈 1: 일반 이동, 전방향 탄막, 쿨타임 기반 이동/공격
    - 페이즈 2: 순간이동, 레이저, 조준 탄막 등 고난이도 패턴
    - 페이즈 전환 시 이미지, 메시지, 연출 효과가 함께 적용됩니다.

- **startBossMove(), updateBossMovement()**
    - 보스의 목표 위치를 랜덤하게 설정하고, 부드럽게 이동시킵니다.
    - 이동 간격, 속도, 얼음타입 능력 효과(이동 속도 감소)도 반영됩니다.

- **startBossTeleport(), updateBossTeleport()**
    - 페이즈 2에서 보스가 순간이동하는 로직을 담당합니다.
    - 이동 후 쿨타임, 위치 랜덤화, 애니메이션 효과 등이 포함됩니다.

- **shootLaserAttack(), shootTargetedBullets(), shootBossBullets()**
    - 보스의 다양한 공격 패턴(레이저, 조준탄, 일반 탄막)을 구현합니다.
    - 각 공격은 쿨타임, 애니메이션, 사운드 효과와 연동됩니다.

- **충돌 처리**
    - 플레이어 총알-보스, 보스 탄막/레이저-플레이어 충돌을 각각 별도의 메서드로 처리합니다.
    - 체력 감소, 피격 애니메이션, 게임 종료 조건 등이 함께 적용됩니다.

---

7. 주요 고민점

- make 파일 ->json파일들 번들로 js로 변환
- 윈도우 cors 에러 -> 로컬 html은 json 읽지 못함 - 로컬에서 CORS policy 관련 에러가 발생하는 이유 :: Take Knowledge's Tech & Knowledge
- **게임 루프 및 시간 관리**
    - requestAnimationFrame을 활용해 60FPS 기준으로 부드러운 게임 루프를 구현했습니다.
    - 시간 보정(timeMultiplier)로 프레임 드랍 시에도 게임 속도가 일정하게 유지됩니다.
    - MDN - requestAnimationFrame 참고

      requestAnimationFrame() - 🌐 웹 애니메이션 최적화 requestAnimationFrame 가이드
- 게임 프레임 원하는대로 설정 - javascript - Controlling fps with requestAnimationFrame? - Stack Overflow
- 현재 시각 표시: performance.now() - Performance: now() method - Web API | MDN

---

8. 참고 자료들

- **상속 구조의 장점**

    - 공통 기능(입력, UI, 포켓몬 시스템 등)은 GameManager에서 일괄 관리하여 코드 중복을 줄이고, 유지보수성을 높였습니다.
    - 각 게임별 특화 로직은 하위 클래스에서만 구현하여 확장성과 가독성을 확보했습니다.
    - MDN - JavaScript 클래스 상속 참고

- MDN Web Docs - Classes
- MDN Web Docs - requestAnimationFrame

---

9. 추가 토의사항

1. **게임 루프 최적화**: requestAnimationFrame과 setInterval의 차이점 및 성능 비교
2. **상속 구조 리팩토링**: GameManager-하위 게임 구조의 장단점, SOLID 원칙 적용 가능성
3. **상속 vs 컴포지션**
    - GameManager 구조를 컴포지션 패턴으로 바꾼다면 어떤 장단점이 있을까요?
4. **게임 밸런싱**
    - 각 포켓몬 능력의 지속시간, 효과, 쿨타임을 어떻게 조정하면 게임 난이도와 재미를 최적화할 수 있을까요?
5. **애니메이션 최적화**
    - requestAnimationFrame 외에 Canvas/WebGL 등 다른 렌더링 기술을 도입할 때의 장단점은 무엇일까요?