뱀 게임 디자인

게임의 진행 순서

1. 게임이 시작하면 두 가지 화면이 보여짐. 하나는 게임 플레이 화면이고 다른 하나는 점수 화면임.
2. 플레이어가 입력을 주면 그 때부터 게임이 시작함. 뱀은 알아서 움직이며 플레이어는 좌우로만 입력을 줄 수 있음.
3. 뱀이 벽에 닿게 되면 체력이 달거나 죽음. 아이템을 먹게 되면 꼬리 길이가 길어짐. 꼬리가 다른 꼬리와 닿게 되면 죽음. 4.모든 아이템을 먹어서 꼬리가 최대 수치가 되면 승리, 그렇지 않다면 점수를 표시해줌.

간단한 프로그램 실행 순서

1. 프로그램이 시작하면 필요한 초기화 작업을 진행함. (초기화 함수 + 필요한 변수 초기화)
2. 플레이어 입력이 있으면 비로소 메인 루프에 진행함. 메인 루프에서는 플레이어 입력을 받고, 여러 연산을 실행함. -> 해당 루프는 고정된 FPS로 실행됨 (e.g 30프레임)
3. 다음은 메인 루프에서 실행되는 입력임

- 플레이어 입력으로 좌 우 키
- 플레이어 입력으로 일시정지

4 다음은 메인 루프에서 실행되는 연산임.

- 뱀의 진행 방향을 전환함.
- 뱀과 벽의 충돌 여부를 감시함
- 뱀과 아이템의 충돌 여부를 감시함.
- 뱀의 크기를 늘리는 연산을 실행함.
- 텔레포트 벽의 생성 여부를 연산함.
- 게임 종료를 검사함.

간단한 의사 코드

```
int main(void) {
    초기화();
    만약 입력이 타당하다면 코드 진행
    주요 반복문 (플레이어 죽음이 아니라면) {
        플레이어 입력 검출();
        뱀의 상태 확인();
        벽의 상태 확인();
        점수판의 상태 확인();
    }
    프로그램 종료();
}
```

Data Oriented Design 개념

- Entity, Component, System 으로 구성

- Entity는 Component를 묶는 단위. Cluster의 형태로 저장될 필요 없음.
- Component는 여러 Entity가 공유하는 Trait(특성) Component에는 로직이 담기지 않고 오로지 Data만이 포함됨. 예시 : Position, Render Information
- System은 자기가 주관하는 Component를 처리하는 관리자. 로직은 시스템에 존재함.
- OOP 와는 달리 어떤 방법이 정석이라는 것은 없음. (최소한 아직은)

클래스 디자인 (Data oriented design 기반)

1. 플레이어 시스템
    - 플레이어 클래스(Entity)
        - 플레이어 몸체 (좌표 배열)
        - 플레이어의 진행 방향
    - 플레이어 시스템 클래스
        - 입력(인풋) 처리
        - 방향 전환
        - 플레이어 움직임
2. 벽 시스템
    - 벽 (클래스) (Entity)
        - 실제로 클래스는 아님.
        - <std::pair<int x, int y> 로 구현
    - 벽 시스템 클래스
        - 벽 정보를 소유
        - 벽을 생성
        - 텔레포트를 생성, 소멸시킴
3. 아이템 시스템
    - 아이템 (클래스) (Entity)
        - 실제로 클래스는 아님
        - <std::pair<int x, int y> 로 구현
    - 아이템 시스템 클래스
        - 아이템 정보를 소유
        - 아이템을 생성, 소멸시킴
4. 인풋 시스템
    - 인풋 시스템 클래스
        - 표준 입력 처리
5. 피직스 시스템
    - 플레이어 피직스 클래스(Entity)
        - 물리 정보
    - 피직스 시스템 클래스
        - 물리 법칙 검사
6. 스코어 보드 시스템

- 스코어 클래스(Entity)
  - 플레이어 정보
- 스코어 보드 시스템 클래스
  - 플레이어 정보 처리

7. 렌더러 시스템

- 렌더러 시스템 클래스
  - 다른 시스템에서 정보를 가져옴
  - 플레이 화면 렌더링
  - 스코어 보드 렌더링

8. 게임 매니저 (시스템)

- 게임 매니저 클래스
  - 게임 실행 방향 결정
    * 게임 실행
    * 메인 루프 실행
    * 일시 정지
    * 게임 종료