FlappyBird (학습) — 컨셉기획서

> 재구성 기획서 (역설계) / 작성 기준 시점: 2021-03-29 (프로젝트 시작 시점) / 문서 버전: Ver 0.1 프로젝트 ID: P036 / 근거 수준: C / 작성: Claude (재구성) 이 문서는 결과물과 제작 기록으로부터 기획 의도를 역추정해 당시 작성한 것처럼 재구성한 것이다. 확인된 사실과 추정을 [근거]/[추정] 태그로 구분한다.

1. 한 줄 컨셉과 기획 의도

한 줄 컨셉: 탭 한 번으로 새를 띄워 파이프 사이를 통과하는 원터치 2D 아케이드를 PlayMaker로 완성하는 학습 과제. [근거: raw/d2.md, BirdController.cs]

기획 의도. - 검증된 단순 규칙의 게임을 끝까지 완성해 Unity 2D 물리, 충돌, 프리팹, FSM의 기본 흐름을 익힌다. [추정: 학습 과제 분류, 세션 없음] - C#은 물리/스폰 같은 반복 로직, PlayMaker는 상태 흐름(시작, 사망, 재시작, 점수 UI)으로 분담해 비주얼 스크립팅 연동을 연습한다. [근거: PlayMakerExtension 폴더, FlappyBirdDiedChecker.cs] - 제출 가능한 완결된 한 판 루프(시작→플레이→사망→재시작, 최고점 저장)를 목표로 한다. [근거: GameController.cs]

2. 타깃, 플랫폼, 엔진, 범위, 인원, 도구

- 타깃: 수업 제출용, 모바일 탭형 2D 아케이드 [근거: raw/d2.md]. 플랫폼은 에디터 실행 기준, 빌드 기록 없음.
- 엔진: Unity 2019.4.22f1 + PlayMaker [근거: raw/scan_D_pass2.jsonl]
- 범위: 단일 씬(`FlappyBird.unity`), 무한 플레이, 스테이지 없음.
- 인원: 1인(박도원). 도구: Unity, PlayMaker, Tap and Fly 스프라이트 팩(Gold Experience Team) [근거: Readme.txt]

3. 핵심 재미와 차별점

- 재미: 탭 타이밍으로 높이를 조절하는 짧은 긴장과 즉시 재도전. [추정: 장르 특성]
- 차별점: 원작 클론이므로 차별화 요소 없음. 점진적 가속(점수마다 속도 증가, 스폰 간격 단축)이 난이도 곡선을 만든다. [근거: GameController.cs]
- 레퍼런스: Flappy Bird. [추정: 장르/파일명]

4. 세계관, 톤, 아트 방향

- 세계관 없음. 픽셀아트 스프라이트(배경 3종, 새 애니메이션, GUI)와 `pixelart.ttf` 폰트로 통일된 레트로 톤. [근거: raw/d2.md, Readme.txt]
- 효과음: Flap, Hit, Score, Fall, Jingle, Button 6종 보유. [근거: Sounds 폴더]

5. 게임 흐름 개요

대기(중력 0) → 첫 탭으로 시작 → 파이프 주기 스폰, 통과 시 +1 → 충돌 시 사망 → 최고점 갱신, 재시작. [근거: GameController.cs, ScoreController.cs]

6. 제작 방향 (파트별 설계 블록)

시스템 파트

- 의도(Why): 한 판 루프를 최소 구성으로 안정 동작시킨다.
- 설계(How): `GameController`가 정적 상태 `isRunning/isFinished`를 갖고 새, 파이프, 바닥이 이를 참조한다. 이벤트는 델리게이트로 FSM에 전달한다. 전역 정적 플래그는 구현이 단순해 학습 규모에 맞아 택했다. [추정: 단순성]
- 필요(What): 스크립트 9종, 태그(`GameController`, `Player`, `Obsticles`, `Destroyer`).
- 작업 방법: C# 컴포넌트 작성 → 씬 배치 → FSM 연결 → 에디터 플레이 확인.
- 완료 기준: 시작, 사망, 재시작 반복이 오류 없이 돈다.
- 리스크/대안: 정적 플래그는 씬 재로드 시 초기화 필요(Start에서 리셋 중).
- 근거: [근거: GameController.cs]

콘텐츠 파트

- 의도(Why): 콘텐츠 없이 파이프 높이 무작위성만으로 변화를 준다.
- 설계(How): 파이프 Y 오프셋을 `Random.Range(-1, 3)` 정수로 선택한다.
- 필요(What): `pipes` 프리팹.
- 작업 방법: 프리팹 제작 → 스폰 위치 지정.
- 완료 기준: 통과 가능한 간격이 항상 유지된다.
- 리스크/대안: 정수 단위라 변주가 4단계뿐. 실수 범위 확장 가능.
- 근거: [근거: GameController.cs, FlappyBird.unity]

아트, 사운드, UI 파트

- 의도(Why): 자체 제작 대신 에셋으로 시간을 학습 핵심에 쓴다.
- 설계(How): Tap and Fly 팩 사용, 점수, 시작, 게임오버 UI는 PlayMaker와 UI로 구성. UI 세부는 미확인.
- 필요(What): 스프라이트, 효과음, 폰트.
- 작업 방법: 에셋 임포트 후 프리팹 구성.
- 완료 기준: 모든 화면 상태에 대응 화면이 있다.
- 리스크/대안: 라이선스상 크레딧 표기 권장(Readme).
- 근거: [근거: Readme.txt] [추정: UI 구성은 FSM 확인 못 함]

데이터 파트

- 의도(Why): 최고점만 유지한다.
- 설계(How): `PlayerPrefs` 키 `High Score`, `Last Score`.

- 필요(What): 없음.
- 작업 방법: 사망 시 FSM이 `UpdateHighScore` 호출.
- 완료 기준: 재실행 후 최고점 유지.
- 리스크/대안: 로컬 한정.
- 근거: [근거: GameController.cs]

7. 제작 단계(마일스톤)

단계	목표	산출물
M1	새 점프/중력	BirdController, 새 프리팹
M2	파이프, 바닥, 배경	스폰/루프 스크립트
M3	점수, 사망, 재시작	GameController, FSM 연동
M4	UI, 사운드, 제출	씬, 제출 zip
순서는 [추정: 의존 관계]. 실제 일정 기록 없음.		

8. 리스크와 결정이 필요한 사항

- 직접 작성 범위가 불명확하다(예제 기반 가능성).
- 남은 약 30%의 정의 없음(사운드 연동, 난이도 조정, 빌드 등 추정).
- 실행 빌드가 없어 동작 검증 기록이 없다.