FlappyBird (학습) — 세부기획서: 플레이 흐름과 수치

> 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. 목적과 범위

- 목적: 한 판의 상태 전이, C#↔PlayMaker 이벤트 계약, 수치, UI/리소스를 한 문서로 고정해 재구현 가능하게 한다.
- 범위: 단일 씬 무한 모드. 콘텐츠 구조(레벨)는 없으므로 해당 없음(스테이지 없음).

2. 설계

상태 전이

- 의도(Why): 전체 흐름을 4상태로 단순화해 FSM과 정적 플래그가 일치하게 한다.
- 설계(How): 대기(`isRunning=false,isFinished=false`) → 플레이(`isRunning=true`) → 사망(`isRunning=false,isFinished=true`) → 씬 재로드로 대기 복귀. 대기에서는 중력 0, 점프 입력이 시작을 겸한다. 사망 후 점프 입력은 `!isFinished` 조건으로 무시된다. [근거: GameController.cs, BirdController.cs]
- 필요(What): 정적 플래그 2개, 재시작 FSM 액션(`FlappyBirdRestartGame`).
- 작업 방법: 상태 표를 만들고 각 상태에서 허용 입력 점검.
- 완료 기준: 모든 상태에서 입력 결과가 표와 일치.
- 리스크/대안: 정적 변수 방식은 다중 인스턴스에 약하다. 과제 규모에서는 허용.
- 근거: [근거: GameController.cs]

이벤트 계약

- 의도(Why): C#과 FSM의 책임 경계를 이벤트 이름으로 고정한다.
- 설계(How): C#→FSM: `Bird Died`, `Score Updated`(Inspector에서 이름 변경 가능). FSM→C#: 점프, 시작, 재시작, 점수 초기화, 최고점 갱신, 점수/최고점 조회 액션. [근거: PlayMakerExtension 폴더 파일명, FlappyBirdDiedChecker.cs, FlappyBirdScoreUpdater.cs]
- 필요(What): `PlayMakerFSM` 참조, 커스텀 액션 9종 중 해당 파일들.
- 작업 방법: 이벤트마다 송신자/수신 상태 표 작성 후 FSM에서 전이 연결.
- 완료 기준: 사망/점수 이벤트가 각 1회씩 FSM에 도달.
- 리스크/대안: 델리게이트가 단일 대입(=)이라 구독자가 둘 이상이면 덮어쓴다. += 변경 필요. [근거: FlappyBirdScoreUpdater.cs]
- 근거: [근거: 위 파일들]

난이도 곡선

- 의도(Why): 규칙 추가 없이 반응 시간을 줄이는 방식으로 긴장을 높인다.
- 설계(How): 스폰마다 `timeDelay -= 0.01`, `speed += 5`. 파이프 Y는 정수 4단계. [근거: GameController.cs]
- 필요(What): 수치 하한(미구현), 플레이 테스트.
- 작업 방법: 20, 50, 100점 구간 체감 확인.
- 완료 기준: 50점 이상에서도 통과 가능한 간격 유지.
- 리스크/대안: `timeDelay` 하한과 `speed` 실제 적용 여부 확인이 필요하다. [제안값] `timeDelay` 최소 0.6초.
- 근거: [근거: GameController.cs] [추정: 하한 제안은 산술 기반]

UI/사운드

- 의도(Why): 상태 변화를 시각/청각으로 즉시 알린다.
- 설계(How): 화면: 시작 안내, 점수, 최고점, 게임오버/재시작. 효과음 매핑(Flap=점프, Score=통과, Hit=충돌, Fall=낙하, Jingle=게임오버, Button=버튼). 매핑은 파일명 기준 설계이며 씬 연결은 미확인. [근거: Sounds 폴더] [추정: 매핑]
- 필요(What): 아래 리소스 목록.
- 작업 방법: FSM 상태 진입 시 오디오 재생 액션 배치.
- 완료 기준: 6개 사운드가 각 이벤트에 연결.
- 리스크/대안: 연결 여부 불명. 남은 약 30%에 해당할 가능성.
- 근거: [추정: 완성도 70%와 파일 존재]

3. 데이터/리소스 명세

이름	타입	값	근거
jumpSpeed	float	8	FlappyBird.unity
rotationAngle	float	30	FlappyBird.unity
회전 감쇠	float	프레임당 -1.5	BirdController.cs
gravityScale(bird)	float	3.5	FlappyBird.unity
speed / speedPace	float	200 / 5	FlappyBird.unity
timeDelay / timePace	float	1 / 0.01	FlappyBird.unity
pipeMinY / pipeMaxY	int	-1 / 3 (Range 상한 제외)	FlappyBird.unity
moveSpeed(pipe), groundSpeed	float	기본 5	PipeController.cs, GroundController.cs
backgroundSpeed	float	기본 3(미사용)	BackgroundController.cs
PlayerPrefs 키	string	`High Score`, `Last Score`	GameController.cs
태그	string	GameController, Player, Obsticles, Destroyer	스크립트 전반

리소스: Tap and Fly 팩(스프라이트/배경 3종, GUI, 효과음 6종), `pixelart.ttf`, 프리팹 `bg`, `floor`, `ground`, `pipes`, 씬 2개. [근거: raw/d2.md, Readme.txt]

4. 작업 순서

1. 상태/이벤트 표 확정 2. 새, 바닥, 파이프 프리팹 3. 스폰, 점수, 사망 구현 4. FSM 연결 5. UI/사운드 6. 수치 튜닝과 하한 추가 7. 제출 zip 정리. 순서는 [추정].

5. 검수 기준

- 첫 탭 전 정지, 탭당 동일 높이 상승.
- 파이프 통과당 +1 한 번, 재실행 후 최고점 유지.
- 충돌 시 이벤트 1회, 바닥/파이프 정지, 재시작 후 상태 초기화.
- 100점 이상 플레이에서 스폰 간격이 음수가 되지 않는다.