

RHYTHM ELEVEN

포스트 모템

리듬을 통해 골을 만들 준비가 되었으면 화면을 눌러주세요

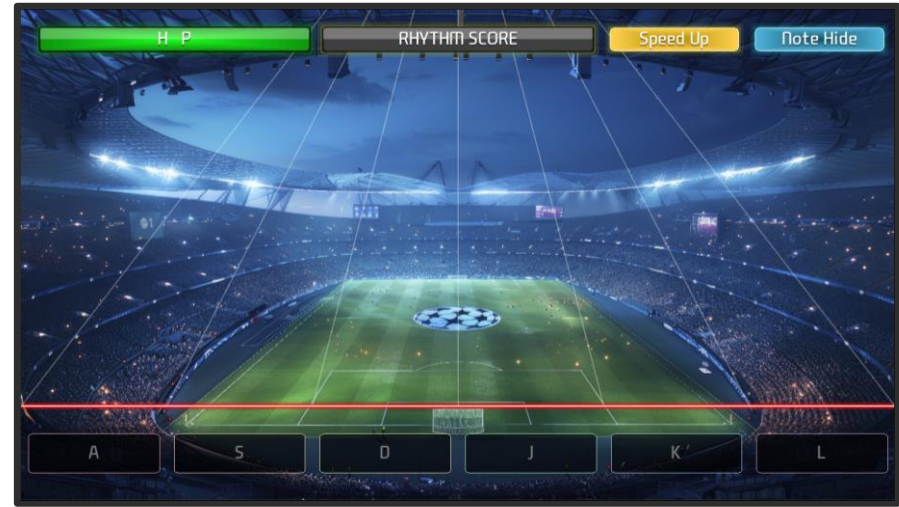
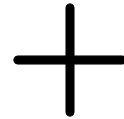
이재영



01

프로젝트 개요

프로젝트 개요



프로젝트 명	리듬 일레븐 (Rhythm Eleven)
장르	리듬 액션 축구
플랫폼	PC (GameMaker)
개발 시간	180H
핵심 개발자	이재영
핵심 컨셉	<ul style="list-style-type: none"> - 턴제 축구 시뮬레이션의 전략성과 리듬 게임의 짜릿한 손맛을 결합하여, 매 순간 긴장감 넘치는 플레이 경험을 제공하는 새로운 장르의 창출 - 축구를 잘 모르는 플레이어도 음악과 연출을 통해 축구 경기의 박진감을 느낄 수 있도록 하는 것을 목표



02

개발 히스토리

개발 히스토리

구분	총 개발 시간	주요 개발 내용
게임 시스템 설계	40H	<ul style="list-style-type: none">- 프로젝트 기반 구축 : GameMaker 초기 설정 및 핵심 오브젝트 구조 설계- 동적 난이도 시스템 : 행동에 따른 리듬 게임 난이도 변경 시스템 최초 도입- 핵심 게임 플로우 완성 : 3라운드 턴제, 스코어보드, 공 소유권, 최종 결과 판정 시스템 구현- 다단계 승리 판정 : 골 > 액션 횟수 > 리듬 점수 순으로 승패를 결정하는 정교한 규칙 확립
리듬 게임 개발	50H	<ul style="list-style-type: none">- 시스템 심화 : '롱 노트' 시스템 추가 및 무한 노트 생성 방식 도입- 원근감 레인 시스템 : 3D 원근감 레인 및 노트 움직임 구현, 각 레인별 출발점 지정- 디자인 및 로직 개선 : 롱노트 디자인 개선 및 라인 침범, 꺾임 문제 해결- 판정 시스템 강화 : PERFECT/GOOD 판정 범위 조절 기능- 타격 효과 추가 : 타격 라인에서 노트 입력 시 노트 별로 상이한 이펙트 효과 연출
UI/UX 및 연출 강화	60H	<ul style="list-style-type: none">- UI 전면 개편 : 포메이션 선수 아이콘, 리듬 게임 HP/스코어바 디자인 개선- 인터랙티브 UX : 선수 선택 시 대상까지의 조준선을 난이도별 색상으로 표시하고, 마우스 오버 시 예상 난이도 텍스트가 뜨도록 구현- 코드 기반 이펙트 : 다이내믹 판정 텍스트, 파티클 화면 전환 효과 등 직접 구현- 동적 버튼 효과 : 방해 스킬 버튼의 사용 가능/완료 상태를 시각적으로 표시- 사운드 시스템 : Persistent 오브젝트를 활용, 각 팀/라운드마다 다른 음악이 재생되는 동적 BGM 시스템 및 라인별 타격음(6종) 등 추가- 최종 결과 화면 개선 : 승패 결정 요인 강조 텍스트 연추, 좌우 분할 레이아웃 적용
버그 수정 및 안정화	30H	<ul style="list-style-type: none">- 액션 카운트 버그 해결 : 액션 횟수 중복 기록 및 미초기화 문제 수정 ('바통 터치' 시스템 도입)- 턴 전환 로직 재설계 : 리듬 게임 성공/실패, 슛/패스, AP 소진 등 복잡한 조건에 따라 턴이 정확하게 넘어가도록 수정- 동적 BGM 버그 해결 : Persistent 오브젝트 활용 및 로직 재설계를 통해 음악 미재생 문제 해결



03

핵심 시스템

1. 동적 난이도 시스템

“ 선수의 행동이 곧 리듬 게임의 난이도가 된다! ”

핵심 특징

선수의 포지션(거리)과 행동(패스/슛)에 따라 리듬 게임의 난이도가 총 9단계로 실시간 변경됩니다.

기대 효과

플레이어의 전략적 선택이 플레이 경험에 직접적인 영향을 미치는 것을 체감하게 합니다.

개발 목표

단순한 리듬 게임을 넘어, 전략적 판단의 깊이를 강화하는 것을 목표로 했습니다

구현 상세

- obj_player_circle(포메이션 화면)에서 두 선수의 position 변수를 비교하여 global.rhythm_game_mode(난이도)를 설정합니다.
- obj_rhythm_game_manager는 이 mode 값에 따라 노트 속도, HP, 목표 점수 등을 동적으로 변경합니다. (모드별 밸런스 조절)



시작 선수	타겟 선수	행동	난이도
골키퍼(GK)	수비수(DF)	패스	하 (EASY)
수비수(DF)	미드필더(MF)	패스	하 (EASY)
골키퍼(GK)	미드필더(MF)	패스	중 (NORMAL)
미드필더(MF)	공격수(FW)	패스	중 (NORMAL)
수비수(DF)	공격수(FW)	패스	상 (HARD)
공격수(FW)	상대 골키퍼	슛	상 (HARD)
골키퍼(GK)	공격수(FW)	패스	최상 (VERY HARD)
미드필더(MF)	상대 골키퍼	슛	최상 (VERY HARD)
골키퍼(GK)	상대 골키퍼	슛	최최상 (GOD)

난이도

2. 턴제 및 정교한 승리 판정 규칙

“ 단순한 점수 경쟁를 넘어, 전략적 효율성을 겨룬다! “

게임 방식

기본 구조: 블루팀 선공, 총 3라운드 (6턴) 진행

턴 교체 조건

리듬 게임 실패 시

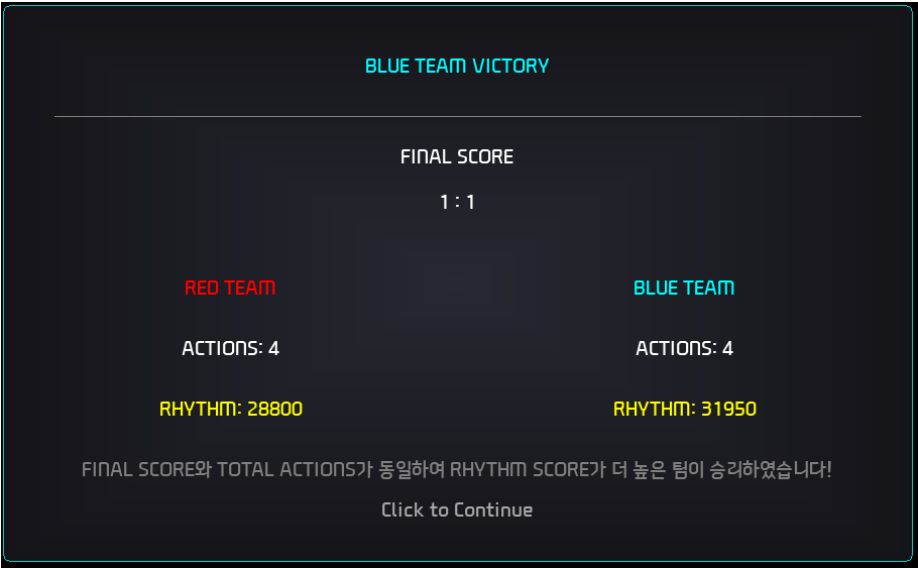
슛을 시도했을 때 (성공/실패 무관)

턴당 주어진 행동 횟수(AP)를 모두 소진했을 때

행동 횟수 (AP)

각 팀은 라운드마다 최대 3번의 행동(패스, 슛)이 가능합니다.
리듬 게임 성공 시에만 1씩 차감됩니다.

승리조건



순위	조건	설명
1순위	FINAL SCORE (골 득점)	더 많은 골을 넣은 팀이 승리합니다.
2순위	TOTAL ACTIONS (총 행동 횟수)	골 동점 시, 더 적은 행동으로 효율적인 플레이를 한 팀이 승리합니다.
3순위	RHYTHM SCORE (리듬 게임 총점)	위 두 조건 모두 동점 시, 리듬 게임 총점이 더 높은 팀이 승리합니다.

3. BGM 및 사운드 시스템

“ 리듬을 완성시켜 골(GOAL)을 만든다! “

음악 구성

블루팀 3곡, 레드팀 3곡 (총 6개 라운드별 BGM)

재생 규칙

턴 기반 재생 : 현재 공격 턴인 팀의 해당 라운드 음악만 재생됩니다.

상황 별 재생 / 일시정지 : 리듬 게임/결과 화면에서는 재생, 포메이션 화면에서는 부드럽게 일시정지됩니다.

연속 재생 : 같은 팀이 턴을 이어갈 경우, 음악은 끊김 없이 계속 재생됩니다.

구현

obj_music_manager를 Persistent 오브젝트로 설정하여 안정성을 확보하고, Step 이벤트에서 현재 룬, 턴, 라운드를 실시간으로 확인하여 lerp 함수로 볼륨을 부드럽게 조절하며 페이드 인/아웃 효과를 구현했습니다.

4. 코드 기반 시각 효과

“ 스프라이트를 넘어, 코드로 구현한 생동감 넘치는 비주얼 “

원근감 레인

노트가 화면 상단의 소실점에서부터 판정선을 향해 날아오는 듯한 원근감 효과를 적용하여 깊이와 몰입감을 더했습니다.

다이내믹 판정 텍스트

PERFECT, GOOD, MISS 텍스트가 나타날 때 확대/축소 및 페이드 효과를 적용하여 타격감을 극대화했습니다.

타격 파동 효과

일반 노트 판정 시, 해당 레인에 원형 파동이 퍼져나가는 시각적 피드백을 추가했습니다.

파티클 화면 전환

수많은 사각 픽셀 파티클이 파도처럼 화면을 휩쓸며 장면을 전환하는 '픽셀 스윕' 효과를 직접 구현하여 세련미를 더했습니다.



04

개발 후기

개발 후기

잘된 점

독창적인 컨셉 구현

'축구'와 '리듬 게임'의 독특한 컨셉을 성공적으로 결합하고, 선수의 행동이 난이도에 직결되는 '동적 난이도 시스템'을 통해 핵심 재미를 구현했습니다.

코드 기반의 유연한 비주얼

대부분의 시각 효과를 코드로 직접 구현하여, 향후 디자인 변경 및 확장에 용이한 유연한 구조를 만들었습니다.

빠른 프로토타이핑과 반복 개선

빠른 프로토타이핑과 지속적인 테스트를 통해 UI, 규칙, 시각 효과 등을 점진적으로 개선하며 게임의 완성도를 높였습니다.

개발 후기

아쉬운 점

데이터 관리 구조의 부재

개발 초기, 체계적인 데이터 관리 구조 없이 global 변수에 의존하여 '액션 카운트'나 'BGM 미재생' 같은 치명적인 버그가 발생했습니다.

오브젝트 역할 분리의 중요성 간과

여러 오브젝트가 비슷한 역할을 동시에 수행하며 코드가 꼬여, 액션 횟수가 2씩 오르는 등의 예상치 못한 버그가 발생했습니다.

초기 비주얼 컨셉의 부재

명확한 비주얼 컨셉 없이 기능 구현에만 집중하다 보니, 파티클 효과나 롱노트 디자인 등 여러 시각적 요소들을 수차례 재작업하게 되었습니다.

개발 후기

배운 점

Persistent 오브젝트의 올바른 이해와 활용

Persistent 오브젝트를 활용해 룸 이동 시 데이터 유실 문제를 해결하며, GameMaker의 핵심 데이터 관리 방식을 체득했습니다.

단일 책임 원칙(SRP)의 중요성

각 오브젝트에 명확한 책임을 부여하는 '단일 책임 원칙'의 중요성을 깨닫고, '바통 터치' 시스템 등으로 버그를 해결했습니다.

체계적인 디버깅의 필요성

show_debug_message를 활용한 'CCTV' 기법으로, 눈에 보이지 않는 문제의 원인을 논리적으로 추적하는 체계적인 디버깅 프로세스를 익혔습니다.

감사합니다.