



[목차]

1. 프로젝트 개요
2. 역할 및 기여
 - 클라이언트 프로그래머
 - 프로그래밍 파트장
3. 개발 환경 구축
 - 버전 관리
 - 협업 규칙 결정
4. 주요 기능 및 구현 내용
 - 세부 과정 정리

1. 프로젝트 개요



경찰로 위장한 도둑이 주인공의 강아지를 훔쳐,
그 뒤를 추적하는 '**러닝 액션 게임**'입니다.



- 게임의 메인 화면

프로젝트 개요

장르	개발 엔진	플랫폼	개발 인원	개발 기간
액션	Unity 5	PC	20명	22.09 ~ 23.11

2. 역할 및 기여



클라이언트 프로그래머로서,

1. 주인공 캐릭터의 이동 및 가속 시스템 구현
2. 주인공 캐릭터의 펀치 시스템 구현
3. 주인공 캐릭터의 초고속 질주 시스템 구현

Aa 이름	태그	날짜
❶ C# 코드 작성 규칙	규칙	2022년 9월 7일
❷ 그래픽 리소스 네이밍 규칙	규칙	2022년 9월 7일
유니티 HDRP→URP 교체	공지	2022년 9월 14일
❸ 프로토타입 시스템 구현 일정	공지	2022년 9월 19일
❹ 라운드 테이블 프로그래밍 파트 PPT	공지	2022년 9월 26일
❺ 게임 컨셉 변경 및 프로그래밍 리셋	공지	2022년 9월 28일
❻ 프로그래밍 파트 재분배 및 일정	공지	2022년 9월 28일
❼ 중간 프로토타입 완료	작업 현황	2022년 10월 11일
❽ 중간 프로토타입 이후 작업 분배	공지	2022년 10월 27일
❾ 카메라 연출 관련 회의	공지	2022년 11월 15일
❿ 3차 라운드 테이블 빌드	작업 현황	2022년 11월 15일
➌ 21일 까지 플밍 작업 분배	공지	2022년 11월 16일
➍ 최종 빌드까지 작업	공지	2022년 11월 27일
➎ 최종 빌드	공지	2022년 12월 12일

프로그래머 파트장으로서,

1. 타 파트들과 소통을 하며 일정 조율, 업무 분배 및 지시
2. 노션 페이지를 통해 프로그래머 파트 공지
3. 프로젝트의 버전 관리
4. 프로젝트 파일 규칙 관리

3. 개발 환경 구축 - 버전 관리



해당 프로젝트 에서는 **Github** 저장소를 사용했습니다.

파트 별로 Branch 를 구분하여 개발 했습니다.

3. 개발 환경 구축 - 협업 규칙

파일 관리 양식

I. 파일 관리 (File Management)

폴더 규칙

이름 규칙

II. 그래픽 리소스 규칙(유니티 각 폴더 안에 예제 이름 파일 있음)

III. Github 깃허브 방식

I. 파일 관리 (File Management)

- 파일 이름은 반드시 영어로 기재*

폴더 규칙

- 전용 리소스는 전용 폴더에, 공용 리소스는 공용 폴더에 예를 들어 Graphic > Character > Player > MrBig_Albedo.tga Graphic > Effect > GlowSpike.png
- 폴더 생성 시 AD or 파트장에게 이야기 하기.

이름 규칙

- 띄어쓰기 금지. 띄어 써야 할 경우 "_"(언더바)로 대체
- 자신의 작업물이 연상되는 이름 짓기(구체적으로)
- 이름이 너무 길어지지 않도록 적당히 단축 명칭 사용 Albedo → Alb
밑에 [그래픽 리소스 규칙 참고](#)
- 파스칼 케이스 사용 첫 단어와 그 다음 나오는 중간 단어들의 시작은 대문자로 통일합니다.
(예시)
 - blackColor → BlackColor
 - Redcat → RedCat
 - ComputerRAMsize → ComputerRAMSize

C# 코드 작성 규칙

태그

규칙

날짜

2022년 9월 7일

작성자

속성 추가

댓글 추가

▶ 꽃 Private , Public 표시

- ▶ Private 다음 public 변수 선언, 함수들도 Private 다음 Public 함수 순서로 정리
- ▶ 파스칼 표기법 사용
- ▶ 카멜 표기법 사용
- ▶ 인터페이스에는 I를 추가
- ▶ bool 값을 가지는 변수 또는 프로퍼티는 Is를 사용
- ▶ 단순히 값을 가져오는 경우에는 Get을 셋팅하는 경우에는 Set을 사용
- ▶ Class 안에 멤버 변수를 다른곳에서 참조하려고 할때는 직접적으로 가져다가 쓰기보다는 get; set;을 활용
- ▶ ENUM은 항상 대문자만을 사용
- ▶ 한 파일에는 하나의 클래스만 담길 수 있도록 해야함
- ▶ 함수의 코드가 길어지는 것을 회피, 함수가 길어진다면 함수를 분리하는 것을 고려해야 함
- ▶ 함수와 변수의 이름은 줄임말을 사용하지 않고 명확한 단어를 사용
- ▶ 규칙
- ▶ 주의가 필요한 함수 또는 이슈에 대한 처리 시 해당 부분에 주석을 달아주는것이 좋음

프로젝트 파일 관리 양식 및 리소스 네이밍 규칙을 설정했습니다.

프로그래밍 파트에서는 C# 코드 작성 규칙을 작성했습니다.

4. 주요 기능 및 구현 내용 - 개발 과정 개요

- 
1. 캐릭터의 이동 시스템
 2. 캐릭터의 가속 시스템
 3. 캐릭터의 펀치 시스템
 4. 캐릭터의 초고속 질주 시스템

1. 캐릭터의 이동 시스템



플레이어는 자동으로 앞으로 달려갑니다.
도로는 3줄이 있으며, 양 옆으로 이동할 수 있습니다.

좌우 방향키를 눌러서
부드럽게 옆으로 이동하며, 항상 해당 라인의
중앙에 올 수 있도록 구현 했습니다.

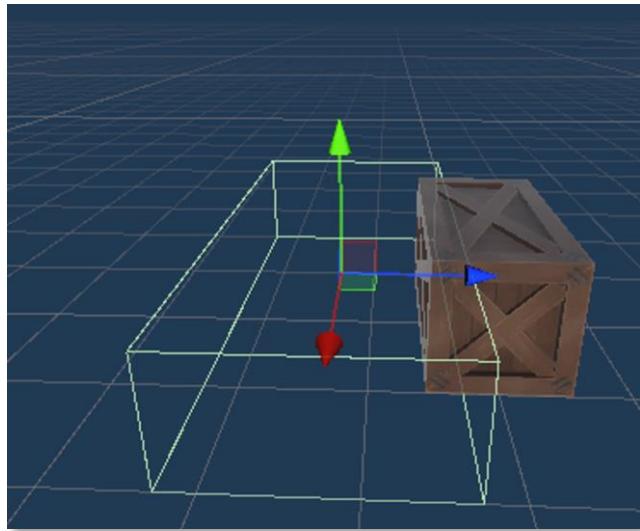
2. 캐릭터의 가속 시스템



캐릭터는 장애물을 아슬아슬하게 피하면
슬로우 모션 후 가속합니다.

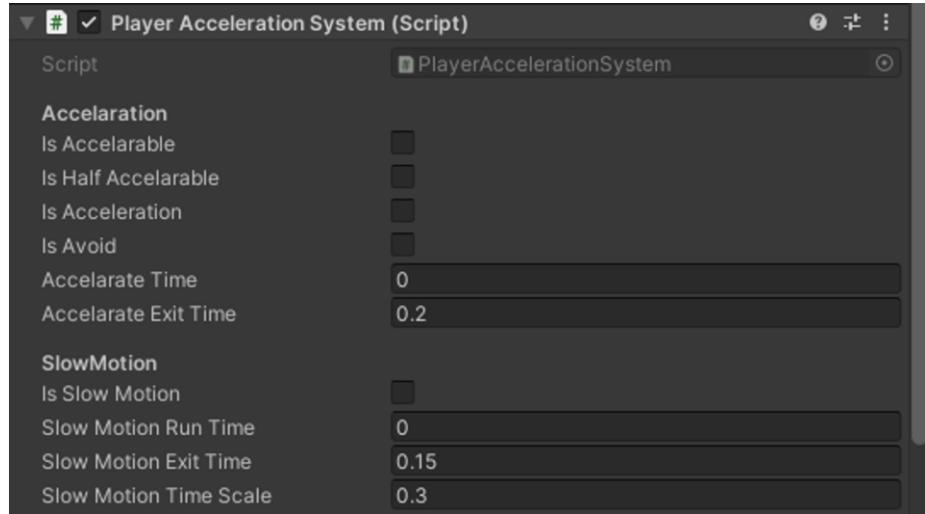
가속은 최대 6번까지 중첩할 수 있으며,
스크린 이펙트가 진해지고 **캐릭터의 속도가 빨라집니다.**

2. 캐릭터의 가속 시스템 - 세부 구현



장애물들 앞에 Collision 을 설치해 아슬아슬하게 피하는 기능을 구현했습니다.

1. 캐릭터가 Overlap 된 상태에서
2. Collision 에서 부딪히지 않고 나가면
3. 가속 상태가 됩니다.



프로퍼티들을 Inspector 에 Visible 하게 만들어, 기획자들이 수정할 수 있도록 구현했습니다.

가속 속도, 슬로우 모션 속도 등등을 세부적으로 조절할 수 있습니다.

3. 캐릭터의 펀치 시스템



- 나무 상자를 펀치 했을 때

캐릭터는 아래 방향키를 눌러 펀치를 할 수 있습니다.

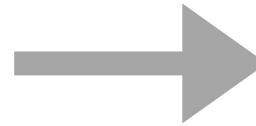
나무 박스는 펀치를 통해 부실 수 있습니다.
또한, 가속 효과를 얻습니다.



- 철제 상자를 펀치 했을 때

철제 박스를 펀치 하면 캐릭터가 기절합니다.
좌우 버튼을 연타해 기절 상태에서 빠져 나올 수 있습니다.

4. 캐릭터의 초고속 질주 시스템



가속이나 편치를 성공 시, 질주 게이지를 얻을 수 있습니다.
게이지가 모두 충전되면, 위 방향키로 초고속 질주를
사용할 수 있습니다.

캐릭터는 게이지가 모두 소모될 때까지
장애물을 모두 부시며 앞으로 달려갑니다.

아래 방향키를 연타해야 게이지 소모가 느려집니다.
이 때, 도둑을 따라잡으면 게임에서 승리합니다.