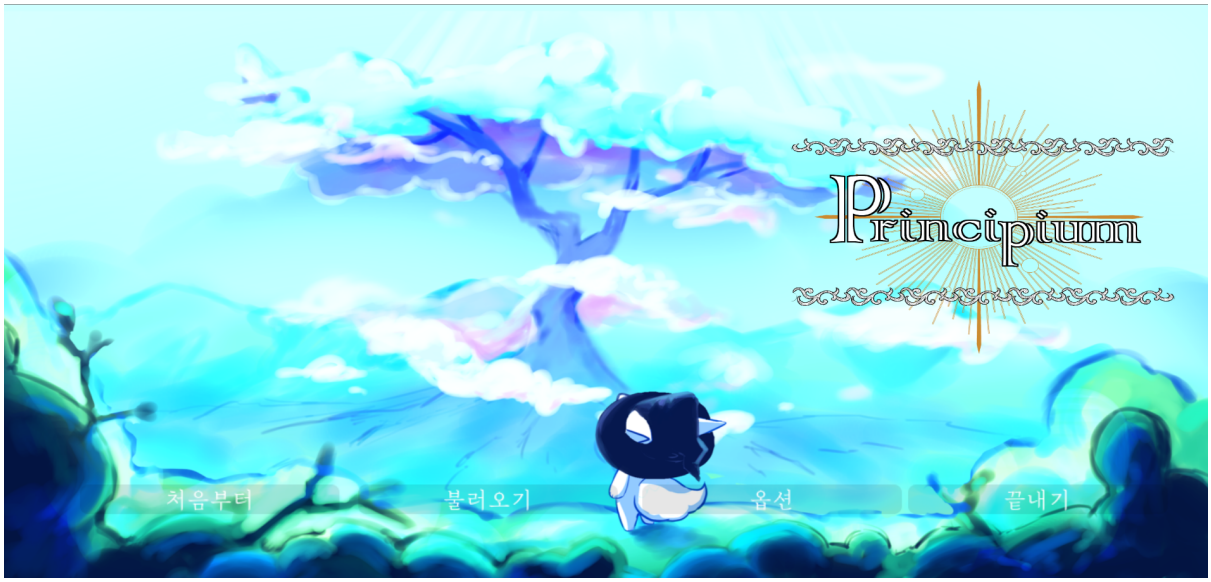
프린키피움(대상)

FrameWork	
category	2D 게임
develop period	@2020년 4월 1일 → 2020년 12월 15일
link	https://github.com/Leejeongseok0407/principium.git
roll	팀장 프로그래밍 프로젝트 매니저
size	Large
언어	C#

플레이 영상



- 플레이 영상

  https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f9ffe8a9-9a61-4634-a804-7551efc12b6e/Principium_2020-12-02_14-38-35.mp4

- 발표 ppt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c00d8d9b-750a-43da-af8c-410f6e14d124/__.pptx

- 광주 문화 산업 진흥원에서 주관하는 인디스타즈 참여

4인 1개조 팀(프로그래머 2 기획 1 그래픽 1)

`주 개발자` 및 `팀장`으로 로 참여함

▼ 간단 소개

```
4인 1개조 팀(프로그래머 2 기획 1 그래픽 1)
 주 개발자 및 팀장으로 로 참여함

전반적으로 게임의 모든 부분을 만들며 팀원들을 이끌어 갔다.

 - 플레이어 제작
 기본적으로 키 입력을 받아서 움직이게 하였으며 캐릭터를 직접 이동시키는 translate를 사용했다.
 레이캐스트를 이용해서 상단에 점프 가능한 블록이 있는 경우 충돌을 없애주어 점프가 가능하게 만들었으며, 하단 발판또한 체크하여 아래로 점프 할 수 있게 만들었다.

 - 몬스터 제작
 몬스터를 구분할 수 있는 레이어를 통하여 충돌 판정을 구분하였고, virtual 을 이용하여 특정 함수를 override가능하게 하여 코드의 중복을 줄였다.

- 상속과 오버라이딩등 다양성 활용
 상속을 활용하여 생산성을 높였다.
 몬스터, 오브젝트의 상위 객체를 생성하고 하위 객체에서 필요한 기능을 재정의하여 사용하여 코드의 중복성을 많이 낮추었던 프로젝트 였다.

- 팀장으로 프로젝트 진행
 에자일 프로세스의 발상에 착안하여 노션에 페이지를 생성하여 리소스 및 진행 상황을 공유하고, 회의를 최소화 하여도 프로젝트가 잘 진행 될 수 있도록 필요 없는 페이지줄

- 모듈화
플레이어 및 몬스터를 유니티의 기능인 프리팹으로 제작하여 모듈화 하였고, 이를 기획자에게 넘겨주어 기획자가 게임을 원하는 방향대로 만들 수 있게 제공하였다.

해당 개발 노션 페이지
https://adorable-birch-42e.notion.site/cc90ffab27a947c2b6aca95aa9d15b22
및 포토폴리오에 페이지를 기록 했습니다.
```

역활 및 개발한 사항

주 개발자로 참여

전반적인 개발을 볼 경우 하단의 페이지, 제가 한 역할을 보고싶은 경우 하단의 삼각형 모양을 눌러주세요

해당 노션 페이지 ⇒ 프린키 피움

모든 애니메이션을 작업함

- 모든 애니메이션은 2D Sprite 파일을 받고 유니티 내부의 애니메이팅 기능을 이용해서 만듬

몬스터, 플레이어, 오브젝트의 주 스크립트 담당

다른 프로그래머가 유니티를 처음 하기 때문에 버그 픽스 및 코드리뷰를 해줌

기획자에게 프리팹을 전달 및 설명 해주며 커스터마이징 가능하게 하여 기획자가 원하는 방식의 게임을 만들 수 있게 함

`세부사항은 삼각형을 눌러서 확인해주세요`

애니메이션 작업

- 모든 애니메이션은 2D Sprite 파일을 받고 유니티 내부의 애니메이팅 기능을 이용해서 만듦
  몬스터, 플레이어, 오브젝트의 주 스크립트 담당
  다른 프로그래머가 유니티를 처음 하므로 버그 픽스 및 코드리뷰를 해줌
  기획자에게 프리팹을 전달 및 설명해 주며 커스터마이징 가능하게 하여 기획자가 원하는 방식의 게임을 만들 수 있게 함
- 몬스터
  1. 몬스터를 구분 할 수 있는 레이어 설정
  2. 몬스터의 상위 스크립트를 제작하고 각각의 몬스터가 상속하여 공통 기능을 사용할 수 있게 함
     특정 패턴이 있으면 특정 패턴을 주기적으로 실행할 수 있게 상위 오브젝트를 구상함

⇒ virtual을 이용하여 특정 함수(void cirtual Skill())를 override해서 제작하고, 코루틴을 통해 특정 시간이 지날 경우 스킬을 사용하고 시간을 0으로 초기화 해줌

▼ 몬스터 종류

1. 겟민숭 달팽이
   매우큰 몬스터로 평소에는 가만히 있다가 느리게 움직임
   ⇒ 이를 유도하여 특정 위치로 옮기기 위해 주로 사용
2. 슬라임
   포탑과 같은 캐릭터로, 총알은 `오브젝트 풀링` 을 이용하여 재사용에 용의하게 함
   ⇒ 처음에 몇 개 만들고 부족할 경우 생성한 뒤에 active, unactive를 통해 보이지 않게만 해서 부하를 줄임 `(GC 줄임)`
   각각의 총알의 스피드를 다르게 할 수 있게 총알을 쏘는 슬라임이 오브젝트 풀에서 총알을 꺼내면서 스피드를 할당해주어 각각 `총알이 다른 속도로 나갈 수 있게` 제작함
   데미지 또한 동일하게 슬라임이 줄 수 있게 설정함
   변수를 직접 쓰는 것이 변수의 값을 바꿀 수 있기 때문에 함수를 통해(return) 참조하게 하였음
3. 호저
   특정 위치 사이를 왕복하는 몬스터로 플레이어를 발견하면 가시를 세우고 따라옴
4. 아귀
   함정형 몬스터로 플레이어가 위에 올라올 경우 입을 닫는 애니메이션을 통해 플레이어에게 데미지를 줌
   플레이어가 감지 범위에 들어올 경우 애니메이션을 시작하고, 코루틴을 이용하여 몬스터에 달린 충돌 감지를 켜서 몬스터와 충돌한 효과를 줌.
   skill을 Ovrride 해서 쿨타임을 돌게 했음
   플레이어 무적 시간에 스킬을 사용할 경우에는 사용하지 않게 설정함
   ⇒ 플레이어의 스크립트에 접근하여 무적을 판별하였으나 개선점이 필요하다고 생각됨

1. 이동 스크립트 제작
   두 가지 상태를 제작하고 bool 값으로 상태를 체크함
   `디폴트` : 안 보이는 게임 오브젝트 사이를 움직이게 설정

- 플레이어
  - 플레이어의 상태를 설정하여 해당 애니메이션에 맞는 움직임을 넣음

    1. 디폴트

       플레이어가 서 있을 경우 애니메이션을 루프함

    2. walk

       움직이는 상태일 경우 애니메이션을 계속 틀어줌

       유니티 내의 인풋 매니저를 통해 x값에 스피드를 곱하여 좌우로 움직일 수 있게함

       이때 TransLate를 사용함 ⇒ 직접 오브젝트의 위치를 이동시켜줌

       움직이는 x벡터를 정규화해서 크기를 1,-1로 토글함

       이때 0일 경우 크기가 0이 되기때문에 체크해줌

    3. jump

       레이 케스트를 이용하여 하단에 땅이 있을 경우 점프 카운트를 초기화 해줌(2번까지 점프 가능)

       레이 케스트를 이용해 하단에 아래로 갈 수 있는 땅이 있을 경우 아래 점프를 가능하게함

       아래점프는 플레이어의 물리 충돌을 없앴다가 다시 줌

       레이 케스트를 이용해 머리위에 올라갈 수 있는 발판이 있을 경우 통과 할 수 있게함

       머리 위에 콜라이더를 씌워 해당하는 발판이 있을 경우 플레이어의 물리충돌을 잠시 꺼줌

  ▼ 플레이어 설정

  MaxHp를 만들고 현재Hp를 만들어서 따로 관리해줌

  - 플레이어가 최대치이상의 체력을 가지는 것을 방지하고, 새로운 스테이지에 진입 할 경우 체력을 초기화 해줌
  - 체력이 없을 경우 플레이어 움직임을 막고, 죽는 애니메이션을 출력

  플레이어가 몬스터와 부딪힐 경우 데미지를 줌

  - 플레이어가 몬스터와 부딪힐 경우 몬스터의 위치와 내 위치를 비교해서 몬스터 반대 방향으로 튕겨 나오게함

- 중복적으로 데미지를 입는 경우를 방지하기 위해 코루틴을 통해 특정 시간동안 데미지를 받지않게 설정함
- 데미지를 입을경우 투명도 조절을 통해서 깜빡거리는 효과를 줌

스킬을 제작함

스킬의 쿨타임을 설정해주고, 각각의 쿨타임을 돌릴 수 있게함

⇒ 쿨타임과, 대기시간의 2개의 배열을 통해 관리함

- 대쉬

  두가지 방법을 고안함

  1. transform.position을 변경
  2. Addforce를 이용하여 뒤에서 밀어주는 방법

  1안일 경우 순간이동 하는 느낌이고 2안은 빠르게 이동하는 느낌이다.

  1안은 갑자기 이동하는 효과로인해 플레이어가 순간이동으로 느낄 수 있기 때문에 뒤에 이미지나 파티클을 통해 잔상 및 대쉬하는 느낌을 줄 수 있음

  2안은 x초 뒤에 velocity를 0으로 하는 것으로 튕겨져 나가는 느낌을 없앨 수 있음

  ⇒ 두안중 고민하다가 velocity를 직접 조절하는 방법을 선택함

  vector2 (헌재 보고 있는 방향 * dashDirction * dashSpeed, 0)으로 하여 빠르게 움직이는 효과를 줌

- 부유

  그래비티를 0으로 하여 플레이어가 상하좌우로 움직일 수 있게함

  코루틴을 통해서 특정 시간동안 중력을 꺼주고 일정 시간 이후 다시 켜주게 했음

  ⇒ invoke를 활용해서도 똑같은 방법이 가능함

  코루틴을 사용한 이유

  1. 오브젝트 비활성화시 코루틴은 멈추지만 인보크는 상태를 유지하여 실행이 보장된다

     이는 엑티브와 인 엑티브할 경우 중복 실행이 될 수 있기 때문에 이와 같이 하였다.

     하지만 비활성화 하는 일이 없었기 때문에 invoke를 사용해도 상관 없었다.

▼ 오브젝트

1. 포탈

   오브젝트가 포탈의 위치에 들어올 경우 플레이어인지 확인한 뒤에 플레이어 일 경우 포탈의 이미지를 바꾸고 위 방향키 이미지를 출력하여 플레이어가 다음 스테이지로 넘어갈 수 있게 안내해줌

   나가면 원래 상태로 돌아감

2. 트리거

   플레이어가 접근하면 위 방향키를 누를 경우 특정 오브젝트가 사라지거나 나타남

   OnTriggerSaty을 사용하여 제작하려 하였으나 콜라이더가 두개 있어서 두번 호출됨

   ⇒ OnTriggerEnter/Exit을 사용하여 해결함

3. 발판

   몬스터 및 플레이어가 올라갈 경우 특정 오브젝트를 사라지게 하는 발판을 생성함

2020.08.26 회고

알게된점

1. 상속을 통한 코드의 중복성 낮추기

   ⇒ 상속과 오버라이딩의 개념을 알고는 있지만 사용하지 않아서 붕뜬 느낌이였는데 이 프로젝트를 하며 개념을 잡음

2. 코루틴과 인보크의 차이점

   a. 루틴은 실행되는 동안 특정 시간이 경과된 후에 특정 동작을 작동할 수 있었지만 인보크는 함수 하나를 통으로 실행하기 때문에 시간에따른 특정값을 함수 안에서 invoke를 통해 관리 할 수 밖에 없어서 코드의 가독성이 저하됨

b. 코루틴과 인보크인 경우 스크립트가 꺼지면 둘다 멈추지만 오브젝트가 꺼질 경우 코루틴은 멈추고 인보크는 실행됨, 이는 오브젝트가 켜질 경우 될 경우 여러번 실행될 수 있는 위험이 있음

⇒ 인보크

세밀조정이 안되지만 단순하게 사용할 경우 코드의 가독성이 높음, 엑티브가 꺼져있더라도 작동이 보장됨

⇒ 코루틴

시간에 따른 세밀 조정이 가능함 메소드가 길어지며, 엔트리 포인트를 다양하게 구현가능

특별히 시도한 점

1. 기획자에게 프리팹을 주며 프리팹의 사용법을 알려줌
   a. 만들고 나서 기획자의 의도와 달라 피드백 받아 수정하는 시간을 단축할 수 있음
   b. 여러가지 상황을 능동적으로 생각할 수 있어서 성장에 도움이됨
2. 노션 페이지를 이용한 리소스 관리
   a. 디자이너에게 노션 페이지를 활용하여 리소스를 보여주고 바로 아래 리소스 다운로드 가능한 링크를 요청함
   b. 이를 통해 리소스 찾는 시간을 단축할 수 있어서 생산성이 향상됨
3. 상속을 이용한 중복 패턴 제거
   a. 상속을 막 알아가던 시절 상속을 통해 부모에서 패턴을 먼저 선언하고 자식 스크립트에서 오버라이딩 하여 코드의 중복을 줄임

아쉬운점

1. 발판 및 트리거에서 유니티 인스팩터 기능을 너무 의존한점

   ⇒ 스크립트에서 함수를 할당하고 실행하는 방식으로 하였으면 더 좋았을것 같다는 생각이 듦
2. 플레이어 상태를 bool값을 이용하여 관리 한점

   ⇒ FSM(유한 상태 머신)을 통해 관리하면 좋았을것 같다는 생각을함
3. 플레이어의 무적 시간을 플레이어 스크립트를 참조하여 확인했는데 이것보다 더 좋을 방법이 있지 않을까 싶음

   ⇒ 플레이어 스크립트를 계속 찾는 것이 아니라 싱글톤으로 플레이어 스크립트를 관리하면 해결될거라 생각됨
4. 파티클을 잘 활용하지 못한점이 안타까움

   ⇒ 파티클을 더 공부하여 파티클을 만들 수 있도록 개선
5. 스테이지를 더욱 잘 만들어서 재미까지 챙기면 좋았겠다는 생각

   ⇒ 좋은 기획자를 기다리는 것이 아닌 내가 스테이지를 만들어 보도록 할것
6. 타일맵을 활용하고 콜라이더를 합치는 과정에서 생긴 여러 오류들이 있었는데 해결하지 못한점이 아쉽다.

   ⇒ transLate를 사용할 경우 많은 힘을 줄 경우 콜라이더 안으로 들어가 버리는데 Math.Clamp로 최대 최소값을 설정하여 해결해야 함.

의문점

1. 몬스터 스킬을 정의하고 하위 오브젝트에서 만드는 디자인 패턴이 있을법 한데 사용은 해도 이름이 뭔지 모르겠음 ⇒ 찾아볼것