GET 6 CHALIES

1. **게임 기획하기**

* 평소에 포켓몬스터나 스타듀밸리, 바람의나라 같은 게임을 선호하여 2D RPG 게임을 만들고자 함.
* 이를 위해 학기 초부터 교내 도서관에 있는 유니티 관련 책을 많이 참고하고 유튜브 강좌나 유니티 도큐먼트를 보면서 유니티 엔진 자체에 대해 공부하려고 노력했고, 학교 각 과제에 그를 적용해보면서 유니티 기본기를 다짐.
* 지난 겨울의 동계 올림픽을 재밌게 보아서 쇼트트랙 게임을 제작하고자 생각하고 곽윤기 선수 캐릭터를 도트로 찍었으나, 빙판 얼음 처리 대각선 이동 그림 등의 효과가 어렵고 정보가 부족하여 새롭게 기획하게 됨.
* ‘히트맨’이란 3D 게임에서 영감을 받아 2D 버전 히트맨을 생각함. 따라서 도둑이 아이템을 훔쳐서 도망가는 게임을 생각하고, 아이템을 훔친 것에 대해 순찰을 보는 경찰 Enemy들이 캐릭터를 쫓아오는 게임을 기획.
* 맵 기획에 있어서는 경찰들이 도둑을 쫓아온다는 점이 ‘팩맨’이 플레이어를 따라오는 것과 비슷하다고 생각해서 ‘팩맨’의 맵 구성을 참고해 유니티 Tilemap의 Collision을 배치.
* 게임을 만드는 것은 스크립트를 쓰는 것이 영화 제작과 비슷하다고 생각해 스토리가 있으면 좋겠다고 생각함. 따라서 Player가 미션을 받고 수행하는 게임을 만들고자 함.
* 히트맨에서 누군가를 포획하거나 Kill하는 미션을 구현하긴 어렵다고 생각해서 맵 전체를 탐험할 수 있도록 6개의 성배를 배치해 성배를 구해오는 미션을 생각함. 성배인 이유는 게임에서 많이 사용되는 아이템으로 에셋을 구하기 쉬웠고, 마을의 제사를 위해 등 스토리를 엮기 쉽다고 생각했기 때문임.
* 탐험 후 다른 맵으로 이동해 이후의 스토리에 대한 엔딩을 넣고자 함.
* Player가 처음부터 게임의 키를 알지 못하기 때문에 튜토리얼로 기본 동작을 알려주고자 함.
* 따라서, 게임 시작 화면 -> 튜토리얼 씬 -> 주요 게임화면(성 안) -> 미션 성공 후(동글) 이러한 순서로 기획함.
* 여러 게임 오브젝트들 파괴할 수 있는 아이템과 사용할 수 있는 아이템 등을 sprite만 있다면 쉽게 확장 할 수 있도록 처음에 오래 걸리더라도 꼼꼼하게 작성하고자 하였고, 게임을 만들어 나가면서 추가하려고 함.
* Enemy가 특정 지역을 순찰하고 있거나 그 자리를 지키고 있다가 특정 Range 안에 들어가면 Player를 추격하도록 함. Enemy와 Player간 공격하고 HP가 소모되게 함과 같은 내용들은 게임을 하는 사람이 직관적으로 알 수 있기에 만들면서 여러 요소를 생각함.

1. **사용한 오브젝트**

* Unity Tilemap 배경 제작  
  : Overworld, rowboat, cave, Castle2, tiles 등, assets-art-worlds에 있는 것들 것, Inner 폴더 (집 내부용),   
  + 모두 사용하진 않음(추후 맵 수정이나 제작을 위해 미리 넣어 놓은 것들과 사용한 것들)
* 캐릭터  
  : 주인공 캐릭터(이미 존재하는 Sprite를 받아온 것. 게임 완성 후 도둑 느낌으로 수정하려 했으나 중간에서 알파버전으로 임시 완성하고 제출해서 수정하지 못함), Enemy 경찰 캐릭터(주인공 캐릭터를 포토샵으로 불러와 원하는 색을 팔레트화 시켜놓고 도트 찍음), NPC 2명  
  \* 3d 에셋에 비해 2d 에셋은 적고, 4way asset의 경우 특히 부족함
* 캐릭터 체력(하트), 깰 수 있는 아이템(항아리), 먹을 수 있는 아이템(하트), 집을 수 있는 아이템(성배), 푯말 등 -> Objects 폴더
* 게임 시작 화면 배경 및 시작 화면 버튼 : 포토샵을 통해 성배 이미지를 따서 다시그려(화질때문) 직접 제작

1. **각 게임 오브젝트에 설정된 컴포넌트(Transform은 생략)  
   \* BoxCollider2D가 2개인데 is Trigger가 체크된 것과 체크되지 않은 것이 함께 들어가 있는 이유는 isTrigger 체크는 충돌 여부를 넓게 잡기 위함이고 체크 되지 않은 것은 그 오브젝트 자체의 영역으로 지나가지 못하게 하기 위함.**

* Player: Sprite Renderer, Rigidbody 2D, BoxCollider2D(is Trigger check), BoxCollider2D(is Trigger not check), Animator, PlayerMovement, ContextClue, SignalListener
* Enemy(경찰) : Sprite Renderer, Rigidbody2D, BoxCollider2D(is Trigger check), BOxCollider2D(is Trigger not check), Knockback,  
   PoliceManEnemy/PatrolPoliceMan
* Sign(푯말), npc
* 집 입구, 동글 들어가는 입구 등 empty Object를 포함하여 Scene을 전환하는 것 : BoxCollider2D(is Trigger check), Dungeon Transition/
* 하트(체력)= HeartHolder + HeartContainers + Heart  
  : HeartHolder의 경우 enemy를 kill 할 때마다 경험치를 쌓게 해서 레벨업시 공격력을 증가시키는 방향으로 할려고 Signal Listener(스크립터블 오브젝트인 Signal을 인지하는 스크립트)를 달았지만 사실상 현재 제출에선 사용하지 않음 + Horizontal LayoutGroup, HeartManager
* 성배(Chalices) : Sprite Renderer, BoxCollider2D(is Trigger check), BoxCollider2D(is Trigger not check), Interactable, PhysicalInventoryItem
* \* 아래쪽으로는 UI 관련 요소들 (Canvas는 제외하고 그 안으로 달린 것을 Panel 기준으로 나눠서 설명 \* (Canvas Renderer, Image 등 생략)
* Inventory Panel = Inventory Background + ScrollView + Item Description  
  : InventoryManager = Image, Text + Image, Scroll Rect + Image, text, button(컴포넌트는 아니지만 밑으로 달린 오브젝트들을 함께 적음)
* ChalicesLack Panel , Lack Text = Image, Text
* GameOver Panel = GameOver Text + RestartButton   
  = Image, Text, Image, Button 등
* Pot : SpriteRenderer, BoxCollider2D, Animator, Pot
* MainCamera : 기본 설정 + AudioSource, CinemachineBrain
* FadeFromWhite/ FadeToWhite : Canvas로 만들어 그 하위의 Panel에 Animator를 넣어 씬 전환시 화면 깜빡하는 효과
* EndingCredit : Ending, BoxCollider2D + 하위로 Canvas와 Panel(크레딧 위에 Text 작성을 위함)
* Title 씬에 있는 캔버스 등은 정적으로 이미지 제작해서 넣는 것으로 하고 위에서 계속 사용한 Canvas, Panel, Button의 구조를 이용함

1. **작성한 스크립트 (코드 삽입하고 코드에 대한 설명 추가하여 제출)  
   \* 후반 부에 작성한 코드일수록 급하게 동작하게 하기 위해 같이 작동 시킬 수 있는 것도 스크립트를 추가해서 넣었기 때문에 비효율적인 코드가 많습니다. Enemy에 오브젝트 풀링 등으로 최적화 작업도 마찬가지로 되어있지 않습니다.**
2. **Player 관련 스크립트**  
   - PlayerMovement

|  |
| --- |
|  |
| Animation 처리 및 상태전이를 위해, 보편적으로 사용하는 FSM(유한상태머신)패턴을 참고함. <https://ozt88.tistory.com/8> |
|  |
| 함수 (update 제외) |
| // update함수에서 어떻게 동작하는 지 |

|  |
| --- |
|  |

- PlayerHit  
- Knockback

|  |
| --- |
| **위에는 변수 부분**  **아래는 실질적인 함수** |

1. **Enemy(경찰) 관련 스크립트**  
   - Enemy (경찰 Enemy 제외하고 추가적인 Enemy가 생길 시, 이 Enemy라는 클래스를 상속받아서 사용하면 됨)

|  |
| --- |
| 상태전이를 위한 Enum 열거    전체적으로 Player와 Enemy의 움직임 자체가 유사하기 때문에 Player와 비슷한 코드 |

- PoliceManEnemy  
: Enemy 클래스를 상속받아서 죽음과 Knock 처리는 필요 없음.  
이 스크립트를 컴포넌트로 가진 Enemy는 자기 자리를 지키다가 Range 이내로 Player가 들어오면 추격

|  |
| --- |
| 변수 및 초기화    FixedUpdate    함수들 |

- PatrolPoliceMan  
: 기본형태를 Enemy class를 상속받은 PoliceMan Enemy로 두고, 그것을 상속 받은 클래스로 animation과 동작 원리는 PoliceManEnemy와 동일하지만 PatrolPoliceMan은 고정위치 순찰이 아니라 특정 위치를 동적으로 순찰하고 있음.

|  |
| --- |
|  |

1. **Scene 전환 관련 스크립트**  
   - DungeonTransition

|  |
| --- |
|  |

- SceneTransition  
: Dungeon Transition과 사실상 동일함 성배 체크 등이 없는 것만 다를 뿐임.

|  |
| --- |
|  |

- RoomMove  
: CamearMovement와 함께 카메라에 대한 처리를 할 때, 씬 전환이 아닌 해당 씬 내에서 구역을 나누어서 구역에 대한 카메라 처리를 할 때 카메라에 바운드를 주며 이동시키기 위해 필요했던 거였는데 씨네머신 처리로 인해서 사실상 필요 없어진 스크립트

|  |
| --- |
|  |

1. **Inventory 관련 스크립트**  
   - InventoryItem

|  |
| --- |
|  |

- InventorySlot

|  |
| --- |
|  |

- PhysicalInventoryItem  
: 대표적인 구더기 코드의 하나로, GameManager에서 성배를 count할 필요가 없는데 들어간 부분

|  |
| --- |
|  |

- PlayerInventory

|  |
| --- |
|  |

1. **Manager 스크립트**- InventoryManager

|  |
| --- |
|  |

- AudioManager  
: 현재는 제일 처음의 게임 타이틀이 들어간 씬에서만 마우스 클릭시 소리가 나는 정도로 사용되고 있는데, 아이템 pot 등을 부셨을 때 나는 효과음이나 공격 모션시 효과음 추가가 필요함. \*유튜버 케이디님 코드 참고  
+ Manager 클래스와 Sound 클래스를 함께 적음

|  |
| --- |
| 사운드 클래스 이 자체를 serializeable 화 (직렬화 함) |

- PauseManager

|  |
| --- |
|  |

- Title

|  |
| --- |
|  |

- GameoverManager  
: 윗부분은 start와 변수이므로 생략

|  |
| --- |
|  |

- GameManager  
: 성배에 대한 개수를 체크하기 위한 역할만 하고 있는 구더기 코드인데 위쪽의 GameOverManager와 합쳐져서 일해야하는 스크립트임. \*수정이 매우 많이 필요한 스크립트\* 그래서 현재는 주석 처리로 막아놓은 것이 많음.

|  |
| --- |
|  |

- CameraMovement  
: RoomMove와 함께 카메라에 대한 처리를 위해 Camera가 캐릭터를 따라가게 하기 위한 것이었는데 씨네머신 처리로 인해 사실상 필요 없는 클래스 ( 너무 길어졌고, 사용하지 않으므로 생략합니다…… )

|  |
| --- |
| // Pass |

- HeartManager

|  |
| --- |
|  |

1. **Object 관련 스크립트**- Interactable

|  |
| --- |
|  |

- Heart

|  |
| --- |
|  |

- Sign

|  |
| --- |
|  |

- Pot

|  |
| --- |
|  |

- Powerup  
: Signal에 대한 정보만 받고 있고 이를 상속 받아서 PowerUp 되게 하는 Item을 처리할 수 있게 했기 때문에 변수 하나만 들고 있는 상속용 스크립트 클래스

|  |
| --- |
|  |

1. **ScriptableOjbects 관련 스크립트  
   \*** [**https://everyday-devup.tistory.com/88**](https://everyday-devup.tistory.com/88) **유니티의 다양한 직렬화에 대한 설명 블로그 링크 (국내용 블로그로 택하여 실제 참고와는 조금 다름)  
   \* 가장 핵심은 ISerializationCallbackReceiver는 class가 아닌 Interface로 다중 상속첨 사용할 수 있게 하고 이를 통해 코드 작성자가 직렬화가 가능한 클래스를 만들 수 있다는 것이 포인트**- FloatValue

|  |
| --- |
|  |

- Signal

|  |
| --- |
|  |

- VectorValue  
: FloatValue와 마찬가지이므로 생략

|  |
| --- |
| **Pass ( 그 값이 Float이 아니고 Vector일 뿐임)** |

- SignalListener

|  |
| --- |
|  |

1. **기타 (엔딩크레딧, Menu 등)**- Menu

|  |
| --- |
|  |

- Ending

|  |
| --- |
|  |

1. **프로젝트 작업 시 느낀 점**

유니티 강좌를 통해 따라 만드는 것 말고, 직접 프로젝트를 기획부터 구현까지 하는 것은 ‘내가 얼만큼 구현할 수 있는 지’를 확신할 수 없어서 어려운 일이었다. 그래도 학기 초부터 여러 유튜브와 책들을 찾아보며 기본기를 다진 것을 나의 프로젝트를 위해 수행함으로써 배운 것을 직접적으로 익힐 수 있는 좋은 경험이었다. 또한 대부분의 RPG 게임은 지속적으로 Player와 상호작용할 수 있는 Objects를 업데이트 하는 경우가 많은데, 왜 오랜 시간이 걸리는 지 알 수 있었다. RPG 게임의 경우 Objects와의 접촉 및 접촉 후 Player에게 어떤 상호작용을 할 지 등에 대한 처리가 정말 작은 것이어도 세세하게 많이 필요했다.

초등학생 시절 컴퓨터실에서 몰래한 ‘병든 공 튀기기 게임’같은 것이 얼마나 아이디어 측면에서 단순하고도 재미있는 지 알 게 되었다. 그리고 구현도 비교적 간단한 게임인지 다시 한번 느끼게 되었다. 다른 학생들의 프로젝트를 시청하면서 다양한 아이디어에 자극받았고, 간단하고 깔끔한 게임은 무엇인 지 생각해보는 계기가 되었다.

2D게임은 3D게임에 비해 몰입도가 적고 RPG게임은 완성이 없는 게임이나 마찬가지여서 어느 정도의 규모가 있어야 완성도 있는 게임처럼 느껴진다. 그래서 단순히 학교 과제를 넘어서 개인 프로젝트로 진행해볼 예정이다. 특히 급한 제출을 위해서 동작만 가능한 코드들(일명 구더기 코드)을 급하게 작성하였다. Enemy에 대한 처리를 Objects Pooling 기법 처리로 바꾸고 destroy 대신 setactive(false)를 활용해 runtime에 있는 메모리를 적게 소모할 수 있게 하면서 빠르고 안정감있는 게임이 되도록 해야겠다.