Survival of Clover Code Documentation

Written by HYE&EUN

0. Data Structure

(1) Clover : 클로버의 데이터를 저장하는 클래스. Azure 와 연동

멤버 변수 :

public string ID : playerID. 애쥬어에서 데이터를 읽어올 때 필요함.

public int c_level : 클로버의 레벨

public string c_name : 클로버의 이름(닉네임)

public float c_sun_stat : 햇빛 수치

public float c_water_stat : 물 수치

public float c_energy_stat : 에너지 수치

public float c_energy_requirement : 레벨업을 위한 에너지 요구량

public int c_weather_ID : 현재 날씨(0 = 맑음 / 1 = 흐림 / 2 = 비)

public int c_bug_ID : 클로버 주변의 벌레 (-1 = 없음 / 0 = 무당벌레 / 2 = 벌 / 3 = 진딧물)

public int c_bug_time : 벌레가 출몰하는 시간(단위 : 초)

public int weatherCycle : 날씨가 변화하는 시간(단위 : 초)

(2) Collections : 콜렉션 오픈 정보를 저장하는 클래스. Azure 와 연동

멤버 변수 :

public string ID : playerID. 애쥬어에서 데이터를 읽어올 때 필요함.

public int n_collection : 전체 컬렉션 갯수

public string opened : 각 컬렉션이 open 되어 있는 지를 0 아니면 1 로 저장하여 ,로 연결하여 string 으로 저장한 변수

(3) GameSetting : 게임 세팅 정보를 저장하는 클래스. Azure 와 연동

멤버 변수 :

public string ID : playerID. 애쥬어에서 데이터를 읽어올 때 필요함.

public string language : 언어설정

1. Loading Scene

(1) GameMusic : 게임 내에서 나오는 음악을 관리하는 클래스

멤버 변수:

private static GameMusic gameMusic: 모든 씬에서 음악을 틀기 위해 Singleton 으로 설정한다.

private bool musicOn = true: 배경음악을 재생하는지 여부. 초기값은 true 로 설정된다.

void Awake(): GameMusic 이 씬이 바뀌어도 삭제되지 않도록 DontDestroyOnLoad 를 실행한다.

void Start(): musicOn 이 true 인 경우, 로딩 씬에서는 노래를 시작(Play)하게 하고, 다른 씬에서는 씬이 바뀜에 따라 정지된 음악을 다시 재생(UnPause)한다.

멤버 함수:

public void GameMusicControl(bool turnOn): musicOn 을 반대로 만들고 gameMusic 을 뮤트하거나 언뮤트한다.

(2) AzureMobileServiceClient : Azure 웹 서버와 프로젝트를 연결해주는 클래스 (reference from https://github.com/dantogno/UnityAzureSample/tree/master/Documentation)

멤버 변수 :

private const bool ignoreTls : Unity 에서 Certification 과 관련한 값

private const string backendUrl : Azure backend 의 url

private static MobileServiceClient client : Azure url 로 접속하는 클라이언트 클래스 인스턴스

멤버 함수 :

public static MobileServiceClient Client() : Azure 클라이언트를 반환하는 함수

(3) LoadingSceneDirector : 로딩씬에서 Azure 에 연동되는 데이터를 읽어오는 클래스

멤버 함수 :

async Task LoadGame() : Playerpref 에 저장된 플레이어 아이디를 이용하여 Azure 에 해당 아이디가 있는 지 확인해서 있으면 데이터를 불러오고, 없으면 Azure 에 해당 아이디의 데이터를 insert 함. 언어 설정이나 클로버의 닉네임 설정이 되어 있지 않은 경우 프롤로그 씬으로, 그렇지 않으면 게임씬으로 씬을 옮김.

public async void OnGameStartButtonClick() : 게임시작 버튼을 누르는 경우 LoadGame task 를 시작하도록 하는 함수

(4) GameSettings : 게임 셋팅과 관련한 로딩과 업로드를 컨트롤하는 클래스

멤버 변수 :

private static GameSettings gameSettings : 게임에서 gameSettings 는 하나만 존재하므로 싱글톤으로 사용하기 위해 선언해줌.

private static GameSetting myGameSettings : Azure 에 연동될 GameSetting 데이터 instance 를 선언

private static int numberOfAttemptsToLoadData : Azure 에서 데이터를 읽어오기 위해 시도하는 횟수

public string playerID : 플레이어 아이디를 체크하기 위한 임시 변수

void Awake() : 씬이 바뀌어도 GameSettings 가 없어지지 않도록 함

멤버 함수 :

public static GameSettings GetInstance() : 싱글톤 GameSettings 반환

public async Task GetData() : Azure 에서 데이터를 가져옴. numberOfAttempsToLoadData 만큼 시도함

public async Task SetData() : Azure 에 데이터를 업데이트

public async void SetLanguage(string lan) : 인자로 string 값(ex) “kor”)을 받아 이에 대응하는 언어로 설정 후 Azure 에 업데이트

public string GetLanguage() : 현재의 언어 설정을 반환

public string GetID() : Azure 에 연동된 data 에서 playerID 를 가져옴

public void SetID(string id) : playerID(id)를 Azure 에 연동될 데이터에 저장

public GameSetting GetGameSetting() : myGameSettings 을 반환해서 로딩씬에서 접근할 수 있게 함. Gamesetting 이 null 인 지 확인하기 위해 예외처리용으로만 사용.

(5) GameSettings_button : 게임 셋팅과 관련한 버튼들을 관리하는 클래스

멤버변수:

private static string[] tutorialScripts : 튜토리얼 스크립트를 line by line 으로 받아 저장하는 배열

private static int currentLine : 현재 스크립트의 몇번째 line 을 읽고 있는 지 보는 변수

private int length : 스크립트의 전체 길이.

void Awake() : 튜토리얼 스크립트를 현재 언어 설정에 따라 읽는 함수

멤버 함수 :

public void OnGameSettingsButtonClick() : 게임 세팅 버튼이 클릭될 경우 게임세팅 팝업을 여는 함수

public void OnLanguageButtonClick() : 언어설정 버튼이 클릭될 경우 언어선택 팝업을 여는 함수

public void OnAllResetButtonClick() : 모두 다 리셋하는 버튼을 클릭할 경우 클로버와 컬렉션 정보를 다 지우는 함수

public void OnGameSettingsPopUpCloseButtonClick() : 게임 셋팅 팝업을 닫는 버튼을 클릭했을 때 게임세팅 팝업을 끄는 함수

public void OnLanguagePopUpCloseButtonClick() : 언어선택 팝업을 닫는 버튼을 클릭했을 때 언어선택 팝업을 끄는 함수

public void OnTutorialButtonClick() : 튜토리얼 버튼이 클릭됐을 때 튜토리얼 팝업을 켜는 함수.

public void OnTutorialNextButtonClick() : next 버튼을 누르면 튜토리얼 스크립트를 넘기는 버튼

public void OnTutorialPopUpCloseButtonClick() : 튜토리얼 버튼을 누르면 팝업을 끄는 함수

public void OnMusicOnOffButtonClick() : 배경음악 on off 버튼을 누르면 음악을 켜고 끄는 함수

public void OnSoundFxOnOffButtonClick() : 효과음 on off 버튼을 누르면 음악을 켜고 끄는 함수

public void OnEndingPopUpResetClick() : 클로버가 죽어서 엔딩이 나왔을 때 리셋버튼을 누르는 경우 클로버의 정보를 초기화시키는 함수

public void OnWatchPrologueButtonClick() : 클릭하면 프롤로그를 다시 보게 해주는 함수

2. Prologue Scene

(1) Prologue : 프롤로그씬과 스크립트를 관리하는 클래스

멤버 변수 :

private static Prologue prologue : 싱클톤 prologue 를 사용하기 위해 선언

private string[] script : 프롤로그 스크립트를 저장하는 배열(각 배열의 원소는 팝업에 한번에 뜨는 문장으로, 한문장 정도로 이루어져있음)

private int length : 프롤로그 스크립트의 길이

private int current_script : 현재 스크립트가 어디에 있는 지를 기록하기 위한 변수

private bool needAnswer : 프롤로그가 진행되다가 answer popup 으로 바뀌게 하기 위한 불리언 변수
private string answer : 플레이어의 답을 저장하기 위한 변수(Yes / No)

async void Start() : 프롤로그 씬이 시작되면 모든 팝업을 닫고, 언어 세팅 설정과 프롤로그 팝업을 켜고 프롤로그를 진행하는 함수를 호출

멤버 함수 :
public static Prologue GetInstance() : 싱글톤 prologue 를 반환
public string[] GetScript(string script, string type) : 스크립트의 종류(script)와 언어(type)를 입력하면 거기에 맞는 .txt 파일을 읽어오는 함수. 컬렉션이나 튜토리얼 등에서도 사용하기 위해 public 으로 선언.
private async Task ShowScript() : 프롤로그 팝업부터 시작해서 플레이어의 응답을 받는 Ans 팝업, 게임시작 팝업으로 넘기는 역할을 함.
public void SetCurrentScript(int value) : next, prev 버튼을 누를 때 스크립트가 바뀔 수 있게 하는 함수. prolouеButtons 에서 사용하기 위해 public 으로 선언.
public void GetAnswer(string ans) : player 에게서 답을 받는 데 사용하는 함수

(2) PrologueButtons : 프롤로그씬에서 사용되는 버튼을 관리하는 클래스
멤버 함수 :
public void OnKoreanButtonClick() : 한국어 버튼이 눌려졌을 때 언어 설정을 바꾸고 컬렉션 스크립트를 바꿔주고 언어 설정 팝업을 꺼줌.
public void OnEnglishButtonClick() : 영어 버튼이 눌려졌을 때 언어 설정을 바꾸고 컬렉션 스크립트를 바꿔주고 언어 설정 팝업을 꺼줌.
public void OnPrevButtonClick() : Prev 버튼이 눌러졌을 때 Prologue 의 current_script 숫자를 1 빼줌
public void OnNextButtonClick() : Next 버튼이 눌러졌을 때 Prologue 의 current_script 숫자를 1 증가함
public void OnYesButtonClick() : Yes 버튼이 눌러졌을 때 플레이어의 응답정보를 Yes 로 주는 함수.
public void OnNoButtonClick() : No 버튼이 눌러졌을 때 플레이어의 응답정보를 No 로 주는 함수.
public void OnGameStartButtonClick() : 게임 스타트 버튼이 눌러졌을 때 씬을 게임씬으로 넘겨주는 함수.
public void OnQuitButtonClick() : 종료 버튼이 눌러졌을 때 게임을 종료하는 함수.

(3) MowerController

멤버 변수:

private RectTransform mowerRect: 현재 오브젝트의 RectTransform

void Start(): 현재 오브젝트의 RectTransform 을 mowerRect 로 가져옴

멤버 함수:

public void MoveMower(): 잔디 기계를 움직이는 코루틴 Move 를 실행한다.

private IEnumerator Move(): 잔디 기계가 클로버가 있는 위치인 350 에 갈 때까지 오른쪽으로 자연스럽게 움직이게 한다.

3. Main Scene

(1) CloverDirector

멤버 변수:

private static CloverDirector myCloverDirector: 게임에서 CloverDirector 는 하나만 존재하므로 싱글톤으로 사용하기 위해 선언해줌.

private static int numberOfAttemptsToLoadData = 3: 아쥬어에서 데이터를 가져오기 위해 시도해보는 횟수

public string playerID = string.Empty: 플레이어 아이디를 확인할 때 쓰는 임시 변수

public GameObject rainSoundObject: 날씨 변화에 따라 빗소리를 켜고 끌 때 필요한 AudioSource 를 가지고 있는 게임 오브젝트

private static Clover myClover = null: 클로버 정보(레벨, 이름, 등등)를 가지고 있는 퍼블릭 클래스

bool weatherRunning: WeatherChange 코루틴과 WeatherStatChange 코루틴이 작동하고 있는지를 저장하는 불리언 변수

Awake(): 씬이 바뀌어도 클로버가 삭제되지 않게 한다. 클로버의 이름이 없을 경우 이름 팝업을 띄운다.

Start(): 날씨 코루틴이 돌아가게 한다.(WeatherFunctionOn)

Update(): GameScene 에 있을 경우에 UI 를 업뎃(UpdateStats)하고, 엔딩 조건을 확인(CheckEnding)하고 날씨가 돌아가는지 확인(WeatherRunning)하고 날씨를 돌아가게 한다.(WeatherFunctionOn)

멤버 함수:

public async Task GetData(): Azure 에서 playerID 를 이용해 클로버 데이터를 가져온다. numberOfAttemptsToLoadData 횟수만큼 시도해보고 playerID 를 찾지 못하면 새로운 플레이어를 생성한다.

public async Task SetData(): Azure 에 클로버 데이터를 저장한다.

private void LoadCloverTexture(int level): 클로버의 레벨을 인자로 받아 레벨에 맞는 이미지를 보이게 한다. UpdateStats()함수가 실행될 때마다 실행된다.

private void LoadWeatherTexture(int weather_ID): 날씨에 맞는 이미지를 보이게 한다. UpdateStats()함수가 실행될 때나 WeatherChange 코루틴에서 날씨정보가 바뀔 때마다 실행된다.

private void LoadRainSound(bool on): 빗소리를 가지고 있는 게임오브젝트를 찾아 빗소리를 시작하거나 정지한다. LoadWeatherTexture 에서 함께 실행된다.

private IEnumerator WeatherChange(): 메인 씬이 활성화되어있을 동안 일정 시간 주기마다 랜덤으로 날씨를 바꾸고 날씨 정보를 저장하고(SaveData) 날씨에 맞는 이미지를 로드(LoadWeatherTexture)한다.

private IEnumerator WeatherStatChange(): 메인 씬이 활성화되어있을 동안 2 초마다 날씨에 따른 스탯을 변경(SetStat)한다. 햇빛이 날 때(c_weather_ID = 0)는 햇빛은 3 증가, 물은 2 감소한다. 구름이 꼈을 때(c_weather_ID = 1)은 햇빛은 1 증가, 물은 1 감소한다. 비가 올 때(c_weather_ID = 2)는 햇빛은 2 감소, 물은 1 증가한다.

public void WeatherFunctionOn(bool functionOn): 인자로 받은 functionOn 이 true 일 때에는 WeatherChange 와 WeatherStatChange 를 시작하고, false 일때는 두 코루틴을 정지한다. Start()와 SetName()에서 코루틴을 시작하고, Reset()과 CheckEnding()에서 코루틴을 정지하는데 사용된다.

public static CloverDirector GetInstance(): 싱글톤 CloverDirector 를 가져오는 함수. CloverDirector 스크립트 밖에서 클로버 정보를 수정할 때 쓰인다.

public string GetName(): 클로버의 이름을 반환한다.

public void SetName(string name): 클로버의 이름을 지정하고 데이터를 저장하고 (SaveData) UI 를 업뎃(UpdateStats)한다. 이름이 String.Empty 일 때 새로 지정할 때만

사용되므로 Reset 후에 실행된다. 따라서 Reset 에서 정지된 날씨를 다시 시작(WeatherFunctionOn)한다.

public int GetWeather(), public void SetWeather(): c_weather_ID 를 반환하고 지정하는 함수.

public float GetStat(string stat): 인자로 받은 이름에 따라 각각 물("water"), 햇빛("sun"), 에너지("energy") 지수와 레벨("level")을 반환한다.
public int SetStat(string statName, float amount): 인자로 받은 이름에 따라 각각 물("water"), 햇빛("sun"), 에너지("energy") 지수에 amount 만큼을 더한다. 물게임에서 얻은 물을 넘길 때와 날씨에 따른 햇빛과 물 지수를 변경할 때, 에너지 버튼을 눌러 에너지를 합성할 때 등 클로버 수치를 변경할 때 쓰인다. 스탯을 변경함에 따라 레벨업(LevelUp) 하는지, 게임 씬일 때 엔딩 조건을 만나는지 확인(CheckEnding)한다.

private void UpdateStats(): 메인 씬에서 클로버 수치에 따라 달라지는 UI 들을 업뎃한다. 클로버의 레벨, 이름, 물/햇빛/에너지 수치에 해당하는 Text UI 들을 모두 찾고 이를 CloverDirector 의 인스턴스의 값을 가져와(GetStat, GetName) 수정한다. 레벨에 맞는 클로버 이미지(LoadCloverTexture)와 날씨 이미지(LoadWeatherTexture)를 로드한다.

public void LevelUp(): 에너지 지수가 레벨업하는데 필요한 수치 이상이 될 경우 레벨을 증가시키고 다음 레벨업에 필요한 에너지수치를 두 배로 늘린다. 변경한 데이터를 저장(SaveData)하고 UI 를 업데이트(UpdateStat)한다. 엔딩 조건을 만족했는지 확인(CheckEnding)한다.

public int GetBugID(): 어떤 벌레가 있는지 c_bug_ID 를 반환한다. BugTimer 클래스에서 벌레 이미지를 띄울 때 벌레 종류를 확인하는데 쓰인다.
public void SetBugID(int bugID): c_bug_ID 를 바꾸고 이를 저장(SaveData)한다. -1 이 아닌 숫자로 변경될 경우 BugTimer 의 StartBugTimer 함수를 실행해 화면에 벌레 타이머를 띄운다. 벌레 버튼을 눌렀을 때와 벌레 시간이 끝났을 때 호출된다.

public int GetBugTime(), public void SetBugTime(int time): GetBugTime 함수는 c_bug_time 을 반환한다. SetBugTime 함수는 인자로 받은 시간으로 c_bug_time 을 수정하고 데이터를 저장(SaveData)한다.

public void Reset(): CloverDirector 의 레벨, 물/햇빛/에너지 지수, 이름, 날씨정보, 벌레 ID 와 벌레 시간(BugIDChange, SetBugTime), 레벨업에 필요한 에너지 수치를 모두 초기화한다. 초기화한 데이터를 저장하고 UI 를 업뎃한다. 기존의 벌레 타이머를 없애고 (BugTimer.Getbugtimer().UpdateBugTimer(-1)) 날씨를 잠시 종료 (WeatherFunctionOn(false))한다.

public void CheckEnding(): 물/햇빛 수치와 레벨에 따른 엔딩을 확인한다. 각각의 경우에 맞게 메시지를 출력하고 엔딩 팝업을 띄운다. 물/햇빛 수치가 바뀔 때(SetStat)와 레벨업할 때(LevelUp) 실행된다.

(2) EndingDirector

멤버 변수:

private int endingFrequency = 300: RandomEnding 함수가 실행되는 주기(초단위)

Start(): EndingFunctionOn(true)실행.

멤버 함수:

private void RandomEnding(): 0~100 사이의 숫자를 랜덤으로 발생시켜 그 숫자에 따라 랜덤한 엔딩(토끼, 발, 닭, 잔디깎이기계, 태풍, 진딧물)이 일어나고 팝업창이 뜨게 한다. 진딧물 이벤트의 경우 숫자가 해당 범위이고 진딧물이 존재해야 일어난다.

public void EndingFunctionOn(bool functionOn): EndingEvent IEnumerator 를 켜고 끄는 함수이다. functionOn 이 true 면 켜고 false 면 끈다.

private IEnumerator EndingEvent(): 메인 씬에 있을 때 일정 시간마다 RandomEnding 함수를 실행한다. 메인 씬이 아닐 경우 코루틴을 종료(EndingFunctionOn)한다.

(3) BugTimer

멤버 변수:

private static BugTimer bugtimer: 벌레 타이머를 싱글톤으로 선언한다.
클로버다이렉터에서 벌레 타이머를 실행하기 위해 필요하다.
private int timeLeft: 벌레에게 남은 시간. 초기값은 60 초이다.
public Text countdown: 남은 시간을 띄울 UI 텍스트.
public Image bugImage: 벌레 종류에 따라 바꿀 UI 이미지.
public Sprite[] bugs: 벌레 종류에 따른 스프라이트를 저장하는 어레이.

Start(): 벌레 타이머를 시작(StartBugTimer)하고 값을 업데이트(UpdateBugTimer)한다.

멤버 함수:
public static BugTimer Getbugtimer(): 싱글톤 BugTimer 를 반환하는 함수. CloverDirector 에서 벌레 정보를 바꾸고 이에 따라 버그타이머를 시작할 때 쓰인다.
public void StartBugTimer(): CloverDirector 에서 남은 시간과 벌레 정보를 가져온다. 남은 시간이 0 보다 작을 경우 60 초로 초기화하고, 벌레 아이디가 -1 이 아닐 경우 버그타이머 코루틴을 시작(BugTimerCoroutine)한다.
public void UpdateBugTimer(int bugID): 벌레 남은 시간 Text 와 벌레 이미지를 띄울 Image 오브젝트를 찾는다. 인자로 받은 벌레 종류와 BugTimer 의 멤버변수 timeLeft 에 맞게 이미지를 띄운다. 벌레가 없을 경우 BugTimerCoroutine 을 멈춘다. CloverDirector 의 Reset 함수와 BugTimer 의 BugTimerCoroutine 안에서 쓰인다.
private IEnumerator BugTimerCoroutine(): GameScene 이 활성화되어 있을 때 1 초가 지날 때마다 남은 시간을 1 씩 감소시키고 이를 저장하고 버그타이머를 업뎃(UpdateBugTimer)한다. 남은 시간이 0 보다 작아질 때 CloverDirector 의 BugID 를 -1 로 바꾼다. (CloverDirector.BugIDChange(-1)) GameScene 이 활성화되어있지 않을 때에는 null 을 양보반환한다.

(4) Water_button
public void OnWaterGameClick(): 물게임 씬으로 전환한다.

(5) Energy_button
private energy_convert_ratio: 에너지 1 을 만드는 데 필요한 햇빛/물 지수.
public void OnEnergyButtonClick(): 햇빛/물지수가 energy_convert_ratio 값보다 클 때 햇빛/물지수를 각각 energy_convert_ratio 만큼 감소시키고(Clov erDirector.SetStat) 에너지를 증가시킨다. 에너지를 증가시키면서 레벨업할 수 있는지 확인한다. (CloverDirector.LevelUp)

(6) Bug_button
멤버 변수:
private float energy_convert_ratio = 5: 벌레를 부르는데 필요한 에너지량
private AudioSource buttonClick: 버튼 클릭 효과음을 위한 변수

Awake(): AudioSource Component 를 가져온다.

멤버 함수:
public void OnBugButtonClick(): 벌레 버튼이 눌렸을 때 실행되는 함수. BugPopU 을 로드한다.
public void OnGenerateMoleculeButtonClick(): BugPopUp 에서 벌레 부르기를 선택했을 때 실행되는 함수. 에너지가 요구량 이상이고 현재 벌레가 없을 때 랜덤으로 벌레를 부르고 팝업을 종료한다.
public void OnCancelButtonClick(): BugPopUp 을 종료한다.

(7) Exit_button & ExitOnEscape
public Exit_button/OnExitButtonClick(): 데이터를 저장하고 어플리케이션을 종료한다.
public ExitOnExcape/Update(): 뒤로가기 키가 눌리면 데이터를 저장하고 어플리케이션을 종료한다.

(8) Ending_button
멤버변수: private AudioSource buttonClick: 버튼 클릭 소리를 내기 위한 AudioSource

void Awake(): buttonClick AudioSource 를 가져온다.

멤버함수: public void OnEndingButtonClick(): 버튼 클릭 소리를 내고 EndingCollectionScene 으로 전환한다.

(9) Submit_button
멤버변수: AudioSource buttonClick: 버튼 클릭 소리를 내기 위한 AudioSource

void Awake(): buttonClick AudioSource 를 가져온다.

멤버함수: public void OnSubmitButtonClick(): 버튼 클릭 소리를 내고 입력받은 이름으로 클로버 이름을 바꾼다(CloverDirector.SetName).

(10) Buttons
멤버함수: public void OnResetButtonClick(): 버튼 클릭 소리를 내고 클로버를 리셋한다(CloverDirector.Reset).

(11) PopUp : popup 을 띄우고 팝업의 text 나 이미지, 버튼에 접근하여 값을 변경하는 등 popup 과 관련한 기능을 하는 클래스

멤버 변수 :

private static PopUp myPopUp : 게임에서 다른 클래스가 팝업과 관련한 기능을 편하게 사용하기 위해서 PopUp 을 싱글톤으로 사용

private CanvasGroup[] PopUps : 씬에 있는 모든 팝업들을 저장

private Button[] Buttons : 씬에 있는 모든 버튼들을 저장

private int n_OpenedPopUps : 화면에 떠있는 팝업의 수를 저장하는 변수

private string playerName : 유저 인풋에서 받을 플레이어 이름을 저장할 임시 변수

public void Awake() : 씬이 로드되면, 해당 씬의 모든 팝업과 버튼을 PopUps 와 Buttons 에 저장하고 모든 팝업을 안보이게 닫음.

멤버 함수 :

public static PopUp GetInstance() : 싱글톤 PopUp 을 가져오는 함수.

public CanvasGroup FindPopUp(string type) : PopUp 클래스 내의 팝업 object 를 반환. type 이 팝업의 이름과 동일한 경우에 해당 팝업을 반환

public Text FindText(string type) : type 에 맞는 팝업의 text component 을 반환

public Image FindImage(string type) : type 에 맞는 팝업의 image component 를 반환

public void ShowPopUp(string type, bool shouldShow) : type 에 맞는 팝업이 현재 씬에 존재하는 경우 팝업의 on/off 를 제어. type 이 만약 엔딩 팝업이라면 그전의 모든 팝업을 끄고 엔딩 팝업을 띄움.

public void ShowPopUp(CanvasGroup PopWindow, bool shoudShow) : ShowPopUp 의 overloaded version. : popup 의 이름이 아니라 팝업 캔버스 그룹을 인자로 받음

public void ShowButton(string name, bool shouldShow) : 버튼의 이름을 인자로 받아서 해당 버튼을 on

public void InactivateOtherButtons(string type, shouldShow) : 팝업이 띄워지면 그 팝업 외의 다른 모든 버튼의 interactive 를 false 로 만듬

public async Task ShowNamePopUp() : 유저로부터 input 을 받는 name popup 에 대해서만 특별하게 사용되는 함수. 유저가 이름을 작성하고 OK 버튼을 눌러서 Clover 의 이름이 바뀌면 팝업을 종료함

public async Task UpdateNewNameAsync() : OK 버튼이 눌려서 유저가 받은 이름 인풋을 받게 되면 그 이름을 CloverDirector 에 넘겨주는 함수.

public void SetName() : name input field 에 적힌 문자열을 임시 변수인 playerName 에 저장하는 함수. OK 버튼이 버튼이 눌리면 호출됨.

(12) BubbleController

멤버 변수:

public Sprite[] bubbles: 말풍선 이미지들(하트, 웃음, 슬픔)

private RectTransform rectTransform: 말풍선의 위치

private int[] yPositions = new int[5]: 레벨에 따른 말풍선의 y 좌표

private Image bubbleImage: 말풍선의 이미지 Component

void Start(): 멤버 변수들 초기화. 레벨에 따른 말풍선의 y 좌표 초기화.

void Update(): CloverDirector 에서 햇빛, 물, 레벨, 벌레 종류를 가져와서 경우에 맞는 말풍선을 띄운다. 진딧물이 와 있을 경우 슬픈 말풍선을 띄운다. 진딧물이 없고 햇빛과 물 수치가 모두 40 이상 60 이하일 때는 하트, 30 이상 70 이하일 때는 웃음 말풍선을 띄운다. 앞의 경우에 모두 해당하지 않을 경우 말풍선을 투명하게 만든다.

4. MineWaterGame Scene & MineWaterGameClear Scene

(1) Element: 물게임의 각 칸의 클래스

멤버 변수:

public bool mine: 돌인지 아닌지를 저장. 돌이면 true, 아니면 false.

public static float rockProbability: 각 칸이 돌이 될 확률. GameSettings 의 모드에 따라 바뀌므로 GameSettings 에서 가져온다.

public int x, y: 격자판에서의 위치를 저장한다. 왼쪽 상단이 (0,0).

public Sprite emptyTextures, rockTexture, flagTexture, unknownTexture: 종류에 따라 다르게 나타나는 이미지를 저장.

초기화(void Start()): x 값 설정(y 값은 인스펙터에서 설정). 돌인지 아닌지를 설정(mine)하고 Grid 의 판 안에 등록하고 방문안함으로 체크한다.

멤버 함수:

Public bool isCovered(): 해당 칸이 기본 디폴트 이미지일 경우 true, 아니면 false 를 반환한다.

private void OnButtonClick(): 칸이 클릭되었을 때 실행된다.

해당 칸이 이미 방문되었다면 아무것도 실행되지 않는다. 방문되지 않았고 깃발 버튼이 눌린 채로 클릭되었다면 칸의 이미지를 깃발 이미지로 바꾼다.
위의 경우에 해당하지 않고 만일 해당 칸이 돌일 경우에는 판의 모든 돌을 보여주고 게임을 종료한다.
돌이 아닐 경우, 해당 칸 주변의 돌 개수에 따라 칸의 이미지를 바꾼다. 해당 칸과 인접하면서 주변에 돌이 없는 칸들을 드러내고, 찾은 물 개수를 업데이트한다. 보드에 존재하는 모든 물을 찾았는지 확인하고 모든 물을 찾았을 경우 게임을 종료한다.
쓰이는 함수: Grid.uncoverMines, Grid.adjacentRocks, Grid.FFuncover, Grid.isFinished, MineGameDirector.UpdateCount, ClearDirector.endOfGame, Element.loadTexture

public void LoadButtonTexture(int adjacentCount): 깃발 버튼이 클릭되어 있는 경우 모르는 칸을 깃발 칸으로 바꾸거나 깃발 칸을 다시 원래대로 모르는 칸으로 바꿔놓는다. 깃발 버튼이 클릭되어있지 않을 경우 칸 주변의 돌 개수에 따라 맞는 이미지를 로드한다. 0 개일 경우 물방울칸, 1~8 개일 경우 주변 돌의 개수가 적힌 물방울칸, 돌일 경우 돌 이미지 칸으로 이미지를 변경시킨다.

(2) Grid: 물 게임의 칸들이 모여있는 격자판의 클래스

멤버 변수:

public static int w, h : 격자판의 너비(12)와 높이(20)

public static Element[w, h] elements: 각 칸을 위치에 따라 저장하는 2d array

public static bool[,] visited: 각 위치의 칸이 방문되었는지 저장.

초기화: Element 의 초기화 때 같이 초기화됨.

멤버 함수:

public static void UncoverMines(): 게임이 끝났을 때 돌이 있는 칸들을 모두 드러나게 하고 판의 모든 칸을 방문한 것으로 표시하는 함수.

public static bool MineAt(int x, int y): x, y 에 있는 칸이 돌인지 아닌지를 반환하는 함수.

public static int AdjacentMines(int x, int y): x, y 에 있는 칸 주변의 돌이 몇 개인지를 반환하는 함수.

public static void UncoverNearElems(int x, int y, bool[,] visited): x, y 좌표의 칸 주변에 있는 칸 중 돌이 주변에 없는 칸을 모두 드러낸다. (지뢰찾기할 때 0 인 칸 누르면 자동으로 주변에 돌이 주변에 있는 칸들까지 확 드러내주는 함수)

public static int RockCount(), Public static int WaterCount(): 판에 존재하는 돌 / 현재까지 찾은 물의 개수를 반환한다. MineGameDirector 에서 UI 를 업데이트할 때 쓰인다.
public static bool IsFinished(): 격자판의 모든 물을 찾았는지 여부를 반환하는 함수.

(3) FlagButton

멤버 변수:
public static bool isClicked: 버튼이 눌린 상태면 true, 아니면 false.
public Sprite clickedTexture: 클릭된 상태의 버튼 이미지
public Sprite unclickedTexture: 클릭되지 않은 상태의 버튼 이미지

초기화(void Start()): isClicked 를 false 로 초기화.

멤버 함수:
public void OnFlagButtonClick(): isClicked 를 반대로 바꾸고 이에 맞는 이미지를 설정한다. 깃발 버튼이 눌릴 때 실행된다.

(4) MineGameDirector

멤버 변수:
public static float water: 지금까지 찾은 물의 개수를 저장.
private static int rock: 판에 존재하는 돌의 개수
public static bool clear: 판에 존재하는 물을 모두 찾았는지를 표시하는 불리언.
static GameObject Water, Rock, Unknown: 각각 찾은 물, 총 돌, 아직 찾지 못한 물의 개수를 띄우는 UI object.
초기화: UpdateUIs() 함수 실행
멤버 함수:
public static void EndOfGame(bool cleared): 인자로 받은 불리언으로 물을 모두 찾았는지를 받아오고 클리어 씬으로 넘긴다.
public static void UpdateUIs(): 돌의 개수, 찾은 물의 개수, 찾지 못한 물의 개수를 인자로 받아 해당 Text UI 를 수정한다. (Grid.WaterCount(), Grid.RockCount() 사용)

(5) gainedWater

멤버 변수:
GameObject waterGain: 클리어 씬에서 게임으로 얻은 물 수치를 나타내는 GameObject

초기화(void Start()): MineGameDirector 에서 얻은 물을 가져와서 클로버의 물 수치를 증가시킨다. 게임을 클리어했을 경우 에너지도 3 얻는다.

(6) OKbutton

멤버 함수: void OnOKButtonClick(): GameScene 으로 돌아간다. OK 버튼이 눌릴 때 실행된다.

5. Collection Scene

(1) Collection : Collection 과 관련한 데이터를 저장하고 관리하는 클래스

멤버 변수:

private static Collection collectionList : 게임에서 collection list 는 하나만 존재하므로 싱글톤으로 사용하기 위해 선언해줌.

private static Collections myCollection : Azure 와 연동될 Collecton Instance 선언

private static int numberOfAttemptsToLoadData : Azure 에서 데이터를 읽어오기 위해 시도하는 횟수

public string playerID : 플레이어 아이디를 체크하기 위한 임시 변수

private bool[] opened : 전체 엔딩에 대해서 해당 컬렉션이 이미 열려있는 지를 저장하는 배열. 배열의 크기는 Task Initialization 에서 초기화될 때 n_collection 만큼 동적할당됨

private string[] collection_name : 각 컬렉션의 제목들을 저장하는 배열. opened 와 마찬가지로 배열의 크기는 초기화될 때 동적할당됨

private string[] collection_info : 각 컬렉션들의 내용을 저장하는 배열. 마찬가지로 배열의 크기는 초기화될 때 동적할당됨

private Sprite[] collection_image : 각 컬렉션들에 해당하는 이미지를 저장하는 배열.

void Awake() : 씬이 바뀌어도 GameSettings 가 없어지지 않도록 함

멤버 함수 :

public static Collection GetInstance() : 싱글톤 Collection 반환

public void SetScript() : 컬렉션 스크립트를 언어설정에서 가져옴.

public async Task GetData() : Azure 에서 데이터를 가져옴.
numberOfAttempsToLoadData 만큼 시도

public async Task SetData() : Azure 에 데이터를 업데이트

public string GetString(string type, int i) : 전체 컬렉션 데이터 중 i 번째 인덱스(i = 0:n_collection-1)의 컬렉션의 type(“name”, “info” 중 하나의 값)에 해당하는 문자열을

반환하는 함수. 컬렉션 UI 를 띄우기 위해서 Collection_button 클래스가 호출하므로 public 으로 선언

public Sprite GetImage(int i) : 전체 컬렉션 데이터 중 i 번째 인덱스의 컬렉션의 type 에 해당하는 image 를 반환하는 함수. 컬렉션 UI 를 띄우기 위해서 Collection_button 클래스가 호출하므로 public 으로 선언

public bool Opened(int i) : 전체 컬렉션 데이터 중 i 번째 인덱스의 컬렉션이 열려있는 지 여부를 반환하는 함수. 마찬가지로 컬렉션 UI 를 띄우는 데 사용되므로 public 으로 선언

public void SetOpened(int i) : i 번째 인덱스의 컬렉션을 opened 로 설정하는 함수.

public async void Reset() : 컬렉션을 모두 지우기 위해 만들어진 함수. Task 인 SetData 를 사용하기 위해 async 로 선언. 게임 셋팅의 All reset 버튼에서 사용하므로 public 으로 선언

(2) Collection_button : Collection scene 에 있는 button 들을 관리하는 클래스(UI)

멤버 변수 :

멤버 변수들은 refererence 를 inspector 에서 걸어주기 위해서 SerializeField 로 선언

// 버튼과 관련한 변수

private int CollectionNumber : 컬렉션의 번호를 저장(각 버튼 = 각 컬렉션에 대응, prefab 으로 만들어짐)

private CollectionButton : 각 컬렉션에 대한 버튼(collection name 을 위에 가지고 있음)

private Text CollectionName : Collection 의 name 을 UI 에서 보여줄 수 있는 text

void Start() : 씬이 로드되면 실행되는 스크립트. collection 정보를 가져오고, 컬렉션 버튼의 text 를 열린 컬렉션은 컬렉션이름을 보여주고 버튼을 누를 수 있게 하고, 열리지 않은 컬렉션은 ???로 바꿔줌.(버튼들의 default 는 interactable 이 false)

멤버 함수 :

public void OnCollectionButtonClick() : 컬렉션 버튼이 클릭되면 호출되는 함수. 관련된 컬렉션 정보를 알려주는 팝업을 띄움

public void OnOKButtonClick() : 컬렉션 팝업에서 OK 를 누르면 호출되는 함수. 컬렉션 팝업을 끄도록 함

(3) Back_button

멤버 함수:

public void OnBackButtonClick() : 뒤로가기 버튼이 눌러졌을 때 게임씬으로 씬을 전환하는 함수