데몬스쿼드 키우기 (글로벌)

플레이스토어 :
https://play.google.com/store/apps/details?id=com.superplanet.demonsquad
ios :
https://apps.apple.com/us/app/demon-squad-idle-rpg/id6504470907

사용한 기술

- Admob,Admob 미디에이션(AppLovin,Meta)
- IronSource SDK
- NHN앱가드
- FirebaseAnalytics
- SingularSDK
- GamePackageManager - WebView
- PhotonChat

GPM WebView

플레이중 인터넷을 열어도 게임진행이 원활하게 하기위해 GPM의 WebView를 사용
웹이 켜질땐 세로로 설정 -> 웹 닫을 경우 다시 가로로 설정

```
private void ShowWebView(string url, ScreenOrientation originalOrientation)
{
    GpmWebView.ShowUrl(
    url,
    new GpmWebViewRequest.Configuration()
    {
        style = GpmWebViewStyle.FULLSCREEN,
        orientation = GpmOrientation.PORTRAIT,
        isClearCookie = true,
        isClearCache = true,
        isNavigationBarVisible = true,
        navigationBarColor = "#4B9EE6",
        title = "Super Reward",
        isBackButtonVisible = true,
        isForwardButtonVisible = true,
        isCloseButtonVisible = true,
        supportMultipleWindows = true,
#if UNITY_IOS
        contentMode = GpmWebViewContentMode.MOBILE
#endif
    },
    (callbackType, data, error) => OnCallback(callbackType, data, error, originalOrientation),
    new List<string>()
    {
    "USER_ CUSTOM_SCHEME"
    });
}

참조 1개
private void OnCallback(GpmWebViewCallback.CallbackType callbackType, string data, GpmWebViewError error, ScreenOrientation originalOrientation)
{
    switch (callbackType)
    {
        case GpmWebViewCallback.CallbackType.Close:
            if (error != null)
            {
                Debug.LogFormat("Fail to close WebView. Error:{0}", error);
            }
            else
            {
                StartCoroutine(RestoreOrientation(originalOrientation));
            }
            break;
    }
}

참조 1개
private IEnumerator RestoreOrientation(ScreenOrientation orientation)
{
    yield return new WaitForSeconds(0.1f);

#if UNITY_ANDROID
    Screen.orientation = orientation;
#elif UNITY_IOS
    if (orientation == ScreenOrientation.LandscapeLeft || orientation == ScreenOrientation.LandscapeRight)
    {
        Screen.orientation = orientation;
    }
    else
    {
        Screen.orientation = orientation;
```

광고 적용

```
public void Show_Ads_Reward_Action(int type, Action _action)
{
    if (UserInfo.instance.adsCount % 5 == 0 && IsOpenAdsPnaelCount == 1)
    {
        IsOpenAdsPnaelCount = 0;
        GameUIManager.instance.OpenAdsRemovePackagePanel();
        return;
    }
    AdsEndAction = _action;
    ads_type = type;
    ShowRewardedInterstitialAd();
}
```

```
public void ShowRewardedInterstitialAd()
{
    const string rewardMsg =
        "Rewarded interstitial ad rewarded the user. Type: {0}, amount: {1}.";

    if (rewardedAd != null && rewardedAd.CanShowAd())
    {
        Debug.Log("Admob광고 실행");
        rewardedAd.Show((Reward reward) =>
        {
            //보상 획득하기
            AdsEnd(false);
            Debug.Log("Admob 광고 끝");
            Debug.Log(string.Format(rewardMsg, reward.Type, reward.Amount));
        });
    }
    else
    {
        PlayIronAds();
    }
}
```

```
public void PlayIronAds()
{
    IronSourcePlacement placement = IronSource.Agent.getPlacementInfo("DefaultRewardedVideo");
    if (placement != null)
    {
        String rewardName = placement.getRewardName();
        Debug.Log("Iron rewardName : " + rewardName);
        int rewardAmount = placement.getRewardAmount();
        Debug.Log("Iron rewardCount : " + rewardAmount);
        if (rewardAmount > 0)
        {
            IronSource.Agent.showRewardedVideo("DefaultRewardedVideo");
        }
        else
        {
            CreateAndLoadRewardedAd();
            NoticeManager.instance.OpenPanel(LocalizationService.Instance.GetTextByKey("Shop_Ad_Load").Replace("₩n", Environment.NewLine));
        }
    }
    else
    {
        CreateAndLoadRewardedAd();
        NoticeManager.instance.OpenPanel(LocalizationService.Instance.GetTextByKey("Shop_Ad_Load").Replace("₩n", Environment.NewLine));
    }
}
```

Admob&Admob미디에이션을통해
광고송출 -> 광고가 없을경우
IronSource에서 광고호출

Localization

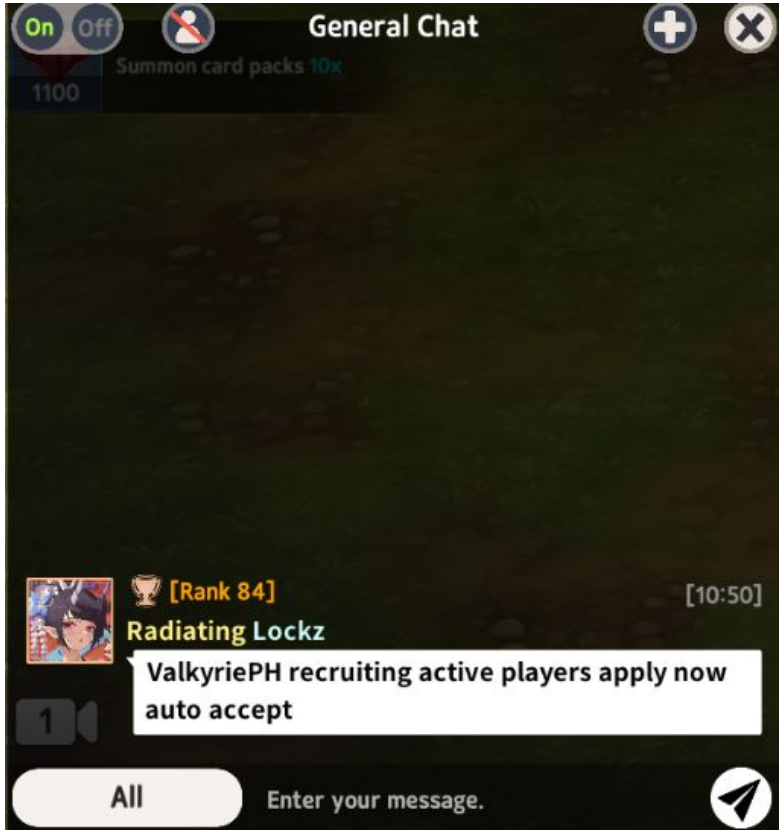
```
if (!File.Exists(savePath) || clientChecksum == "0" || clientChecksum != serverChecksum.ToString())
{
    StartCoroutine(GetSSLocale());
}
```

```
public IEnumerator GetSSLocale()
{
    WWWForm form = new WWWForm();
    form.AddField("client_key", Client_key);
    form.AddField("locale", Localizationstring);

    using (UnityWebRequest request = UnityWebRequest.Post(SuperHost + "[redacted]", form))
    {
        request.SendWebRequest();
        while (!request.isDone)
        {
            Debug.Log($"Download Progress: {request.downloadProgress * 100}%");
            yield return null;
        }
        if (request.result == UnityWebRequest.Result.ConnectionError || request.result == UnityWebRequest.Result.ProtocolError)
        {
            Debug.LogError("Error : " + request.error);
            StartCoroutine(ProcessError(request.responseCode));
        }
        else
        {
            string directoryPath = Path.GetDirectoryName(savePath);
            if (!Directory.Exists(directoryPath))
            {
                Directory.CreateDirectory(directoryPath);
            }

            if (File.Exists(savePath))
            {
                File.Delete(savePath);
                Debug.Log("Deleted existing CSV file.");
            }

            // 새 파일 저장
            File.WriteAllBytes(savePath, request.downloadHandler.data);
            yield return new WaitUntil(() => File.Exists(savePath));
            Event("app_game_model_init_end");
            SetPlayerPrefs();
            PlayerPrefs.Save();
            StartCoroutine(LoadYourSceneAsync());
        }
    }
}
```

```
public Dictionary<string, string> LoadLocalizeFileHelper()
{
    var dict = new Dictionary<string, string>();
    try
    {
        string[] lines = File.ReadAllLines(SuperServiceManager.instance.savePath);
        string[] headers = ParseLine(lines[0]).ToArray();

        int index = -1;
        for (int i = 0; i < headers.Length; i++)
        {
            if (headers[i].Contains(SuperServiceManager.instance.Localizationstring))
            {
                index = i;
                break;
            }
        }

        if (index == -1)
        {
            Debug.LogError("같은 언어가 없음");
            return null;
        }

        // 각 행을 읽어서 딕셔너리에 추가
        for (int i = 1; i < lines.Length; i++)
        {
            List<string> data = ParseLine(lines[i]);

            if (data.Count > index)
            {
                string key = data[0].Replace("₩", "");
                string value = data[index].Replace("₩", "");
                if (value.Contains("₩n")) value = value.Replace("₩n", "\n");
                if (DataManager.instance.csv_Lang == CSV_LangType.EN) value = value.Replace("~", "-");
                dict[key] = value;
            }
        }
        CanGetLang = true;
    }
    catch (Exception e)
    {
        Debug.LogError("Error loading CSV file: " + e.Message);
    }

    ChangeLangAction?.Invoke();
    return dict;
}
```

서버에서 현재 언어 버전을 체크 후 Localization.csv를 다운로드 후 파싱해주는 기능

Photon Chat

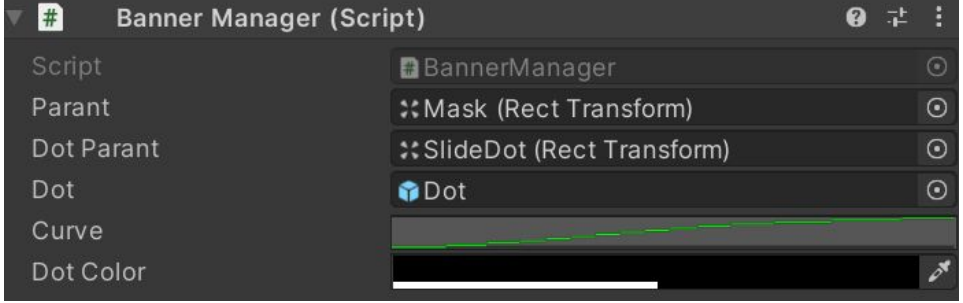



```
void Start()
{
    userName = UserInfo.instance.userNickname;
    userId = UserInfo.instance.userId;
    currentChannelName = ServerFunction.Instance.host_server + SuperServiceManager.instance.Localizationstring;
    chat_type = 1; // 일반 채팅

    // 서버간 시스템 분리
    systemChannelName = systemChannelName + "_" + currentChannelName;


    if (UserInfo.instance.userGuildInfo.guildIdx > 0)
    {
        guildChannelName = "GM_Guild_" + UserInfo.instance.userGuildInfo.guildIdx.ToString();
    }

    chatClient = new ChatClient(this);
    chatClient.Connect(ChatSettings.Instance.AppId, chat_version, new AuthenticationValues(userName));
}
```

```
if (!isOnChat)
    return;

string[] words = chat_message.Split("}P$T@S8");

if (UserInfo.instance.blockUserInfo.ContainsKey(int.Parse(words[1])))
{
    Debug.Log(user_name + " " + chat_message);
    return;
}

if (current_count >= worldChatSlots.Count)
    worldChatSlots.Add(Instantiate(chatSlotPrefab, chat_content).GetComponent<Chat_Child>());

worldChatSlots[current_count].Chat_Set_Message(user_name, chat_message, true);
worldChatSlots[current_count].gameObject.transform.SetAsLastSibling();

if (ChatTest.instance.chat_type == 1)
{
    chat_content_sizefitter.enabled = true;
    LayoutRebuilder.ForceRebuildLayoutImmediate(chat_content);
    chat_content_sizefitter.enabled = false;
}

current_count++;

if (current_count >= CHATSLOTSIZE)
    current_count = 0;
```

PhotonChat을 사용하여
서버,언어를 비교하여
해당 채널로변경 & 텍스트
상태에 맞춰 텍스트창
설정

배너 제작(영상입니다)




```
private void DetectSwipe()
{
    if (bannerSlot.Count > 0)
    {
        Vector2 swipeDelta = endTouchPosition - startTouchPosition;

        if (swipeDelta.magnitude > 50f)
        {
            if (Mathf.Abs(swipeDelta.x) > Mathf.Abs(swipeDelta.y))
            {
                if (swipeDelta.x > 0)
                {
                    OnSwipeRight();
                }
                else
                {
                    OnSwipeLeft();
                }
            }
        }
        else
        {
            bannerSlot[curIndex].BtnClick();
        }
    }
}

참조 1개
private void OnSwipeRight()
{
    if (isMoving) return;
    if (cor != null)
    {
        StopCoroutine(cor);
        cor = null;
    }
    cor = StartCoroutine(PlayBannerRightCo
}

참조 1개
private void OnSwipeLeft()
{
    if (isMoving) return;
    if (cor != null)
    {
        StopCoroutine(cor);
        cor = null;
    }
    cor = StartCoroutine(PlayBannerLeftCor
}
```

```
public class NewServerBanner : BannerSlot
{
    참조 2개
    public override void BtnClick()
    {
        GameUIManager.instance.OpenNewServerOpenEventPanel();
    }

    참조 2개
    public override void Setting()
    {
        EventManager.Instance.NewServerOpenEvent.EventRedLight = RedRight;
        EventManager.Instance.NewServerOpenEvent.Setting();
    }
```

배너 생성후 움직임은
Lerp와
애니메이션커브를
통해 무빙.

배너기능은
추상화함수를통해
확장성 증진.

패스 제작(영상입니다)


```
public int MaxPassLevelCheck() //해당 타입의 내가 보상받기 가능한 패스 단계 체크
{
    if (typeNum < 100)
    {
        for (int i = (typeNum) * 5; i < (typeNum + 1) * 5; i++)
        {
            if (DataManager.instance.passData[selectType].Count <= i) continue;
            for (int j = 0; j < DataManager.instance.passData[selectType][i].slotCount; j++)
            {
                if (GetFreeRewardCheck(selectType, i, j) == 1 || GetPayRewardCheck(selectType, i, j) == 1) //받을 수 있는 보상이 존재한다면
                {
                    return i;
                }
            }
        }
    }
    else
    {
        for (int i = (typeNum - 100) * 5; i < (typeNum - 99) * 5; i++)
        {
            if (DataManager.instance.passData[selectType].Count <= i) continue;
            for (int j = 0; j < DataManager.instance.passData[selectType][i].slotCount; j++)
            {
                if (GetFreeRewardCheck(selectType, i, j) == 1 || GetPayRewardCheck(selectType, i, j) == 1) //받을 수 있는 보상이 존재한다면
                {
                    return i;
                }
            }
        }
    }
    return 0;
}
```

```
public void PassRedLightCheck(int _type, int _typenum = -1) //false: 레드라이트가 꺼진 상태에서만 체크 true: 레드라이트 여부 상관없이 체크
{
    bool isLightActive = false;

    if (_type == 0)
    {
        if (_typenum == -1)
        {
            for (int i = 0; i < DataManager.instance.passData[0].Count / 5; i++)
            {
                if (UserInfo.instance.userSquadLevel >= DataManager.instance.passData[0][i * 5].startVal)
                {
                    _typenum = i;
                    typeNum = i;
                }
            }
            if (_typenum == -1)
            {
                typeNum = DataManager.instance.passData[0].Count / 5 - 1;
            }
        }
        typeBtnRedLight[typeNum].SetActive(false);

        for (int j = typeNum * 5; j < (typeNum + 1) * 5; j++)
        {
            if (MoveBtnRedLightCheck(0, j, true))
            {
                typeBtnRedLight[typeNum].SetActive(true);
                isLightActive = true;
            }
        }
    }
    else
    {
        if (_typenum == -1)
        {
            for (int i = 0; i < DataManager.instance.passData[1].Count / 5; i++)
            {
                if (UserInfo.instance.userSquadLevel >= DataManager.instance.passData[1][i * 5].startVal)
                {
                    _typenum = i + 100;
                    typeNum = i + 100;
                }
            }
            if (_typenum == -1)
            {
                typeNum = 100 + DataManager.instance.passData[1].Count / 5 - 1;
            }
        }

        typeBtnRedLight[typeNum - 98].SetActive(false);

        for (int j = (typeNum - 100) * 5; j < (typeNum - 99) * 5; j++)
        {
            if (MoveBtnRedLightCheck(1, j, true))
            {
                typeBtnRedLight[typeNum - 98].SetActive(true);
                isLightActive = true;
            }
        }
    }
```

레벨과
스테이지에 따라
패스보상체크&
현재 받을수있는
레벨로 이동

시즌패스 제작(영상입니다)


```
private IEnumerator UpdateRemainingTime()
{
    DateTime endTime = Eventpanel.EndData;
    while (true)
    {
        DateTime currentTimeKST = DateTime.UtcNow.AddHours(9);

        string remainingTime = GetRemainingTime(currentTimeKST, endTime);
        SeasonPassRemainText.text = remainingTime;

        yield return new WaitForSeconds(1f);

        if (currentTimeKST >= endTime)
        {
            NoticeManager.instance.OpenPanel(LocalizationService.Instance.GetTextByKey("Text_SeasonPassEnd"));
            BannerManager.instance.BannerSetting();
            ClosePanel();
            yield break;
        }
    }
}
참조 1개
private string GetRemainingTime(DateTime currentTime, DateTime endTime)
{
    TimeSpan remainingTimeSpan = endTime - currentTime;

    if (remainingTimeSpan.TotalDays >= 1) // 1일 이상 남은 경우
    {
        int days = (int)remainingTimeSpan.TotalDays; // 전체 일수
        int hours = remainingTimeSpan.Hours; // 나머지 시간
        int minutes = remainingTimeSpan.Minutes; // 나머지 분
        int seconds = remainingTimeSpan.Seconds; // 나머지 초

        if (hours == 0 && minutes == 0 && seconds > 0) // X일 0시간 0분 59초
        {
            return LocalizationService.Instance.GetTextParseByKey("Text_SeasonPassTime2", days);
        }
        else if (hours >= 1) // X일 1시간 이상
        {
            return LocalizationService.Instance.GetTextParseByKey("Text_SeasonPassTime1", days, hours);
        }
        else if (minutes >= 1) // X일 0시간 1분 이상
        {
            return LocalizationService.Instance.GetTextParseByKey("Text_SeasonPassTime3", days, minutes);
        }
        else
        {
            return LocalizationService.Instance.GetTextParseByKey("Text_SeasonPassTime2", days);
        }
    }
    // 1일 미만 남은 경우
    else if (remainingTimeSpan.TotalHours >= 1) // 1시간 이상
    {
        int hours = (int)remainingTimeSpan.TotalHours; // 전체 시간
        int minutes = remainingTimeSpan.Minutes; // 나머지 분
        int seconds = remainingTimeSpan.Seconds; // 나머지 초

        if (minutes == 0 && seconds > 0) // X시간 0분 59초
        {
            return LocalizationService.Instance.GetTextParseByKey("Text_SeasonPassTime5", hours);
        }
```

```
/// <summary>
/// 패스 위치 변경(슬롯세팅할때 자동으로해줘야하니 Init에서 제외)
/// </summary>
참조 1개
public void PassSliderSetting()
{
    int smallestFalseIndex;

    if (Eventpanel.specialRewardGetIdx > -100) // 패스 구매.
        smallestFalseIndex = Eventpanel.normalRewardGetIdx < Eventpanel.specialRewardGetIdx ? Eventpanel.normalRewardGetIdx : Eventpanel.specialRewardGetIdx;
    else
        smallestFalseIndex = Eventpanel.normalRewardGetIdx;

    float _default = -210;
    float _plusValue = 220;

    if (smallestFalseIndex == 0)
        Content.anchoredPosition = Vector2.zero;
    else
        Content.anchoredPosition = new Vector2(_default - _plusValue * (smallestFalseIndex - 1), Content.anchoredPosition.y);

    expSlider.value = Eventpanel.Event_Token;

    if (Eventpanel.CurLevel + 1 >= Eventpanel.RewardNeedPoint.Count() && (float)Eventpanel.CurNeedPoint / (Eventpanel.RewardNeedPoint[Eventpanel.CurLevel] - Eventpanel.RewardNeedP
        finalExpSlider.value = ((float)Eventpanel.CurNeedPoint / (Eventpanel.RewardNeedPoint[Eventpanel.CurLevel] - Eventpanel.RewardNeedPoint[Eventpanel.CurLevel - 1]) - 0.5f) *
    else
        finalExpSlider.value = 0;

    if (Eventpanel.CurLevel + 1 >= Eventpanel.RewardNeedPoint.Count())
        maxExpSlider.value = 1;
    else
        maxExpSlider.value = 0;
}

/// <summary>
/// 일괄수령 버튼 활성화체크
/// </summary>
참조 1개
public void TotalButtonSetting()
{
    bool _CanGetReward = false;

    for (int i = 0; i < normalSlot.Count; i++)
    {
        if (Eventpanel.CanGetPassReward(0, i) == 1)
        {
            _CanGetReward = true;
            break;
        }

        if (Eventpanel.CanPurchsePass)
            continue;

        if (Eventpanel.CanGetPassReward(1, i) == 1)
        {
            _CanGetReward = true;
            break;
        }
    }
```

```
public void TouchSpecialServerCheck(int _index)
{
    ServerFunction.Instance.Season_Pass_Mission_Clear(UserInfo.instance.userId, Eventpanel.Event_Index, false, _index + 1, callback =>
    {
        if (callback.Equals("OK"))
        {
            // 경험치 초과 처리.
            if (Eventpanel.Event_Token >=  Eventpanel.RewardNeedPoint[Eventpanel.RewardNeedPoint.Length - 1]) // 이미 만렙.
            {
                int overExp = specialMissionExp[_index];

                List<InAppReward> _temp = new List<InAppReward>();
                _temp.Add(new InAppReward(0, 2, overExp * 100));

                for (int i = 0; i < _temp.Count; i++)
                    _temp[i].SetReward();
            }
            else if (Eventpanel.Event_Token + specialMissionExp[_index] > Eventpanel.RewardNeedPoint[Eventpanel.RewardNeedPoint.Length - 1]) // 획득시 만렙.
            {
                int overExp = (Eventpanel.Event_Token + specialMissionExp[_index]) - Eventpanel.RewardNeedPoint[Eventpanel.RewardNeedPoint.Length - 1];
                int getExp = Eventpanel.RewardNeedPoint[Eventpanel.RewardNeedPoint.Length - 1] - Eventpanel.Event_Token;

                List<InAppReward> _temp = new List<InAppReward>();
                _temp.Add(new InAppReward(0, 39, getExp));
                _temp.Add(new InAppReward(0, 2, overExp * 100));

                for (int i = 0; i < _temp.Count; i++)
                    _temp[i].SetReward();
            }
            else
            {

                for (int i = 0; i < specialRewardList[i].Count; i++)
                    specialRewardList[i][_index].SetReward();
            }

            Eventpanel.MissionSpecialClear[_index] = true;

            panel.SetRewardSlider();

            GameUIManager.instance.rewardPanel.OpenPanel();
            CanOnRedLight();
        }
```

```
public class SeasonPassEvent : EventInterface
{
    public InAppReward WeaponSkin;

    public readonly int[] RewardNeedPoint = new int[31] { 0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 130

    public int normalRewardGetIdx;
    public int specialRewardGetIdx;

    참조 6개
    public int CurNeedPoint // 누적 필요 경치 리턴.
    {
        get
        {
            if (CurLevel + 1 >= RewardNeedPoint.Length)
                return RewardNeedPoint[RewardNeedPoint.Length - 1];

            return RewardNeedPoint[CurLevel + 1];
        }
    }

    참조 6개
    public int LevelUpRuby
    {
        get
        {
            if (CurNeedPoint <= Event_Token) // 경험치 충분함.
                return 0;
            else
                return (CurNeedPoint - Event_Token) * 500;
        }
    }

    참조 25개
    public int CurLevel
    {
        get
        {
            int i = 0;

            for (; i < RewardNeedPoint.Length; i++)
            {
                if (Event_Token < RewardNeedPoint[i])
                    return i - 1;
            }

            return RewardNeedPoint.Length - 1;
        }
    }

    public bool CanPurchsePass; //패스를 구매할수있는지(false = 패스구매)

    public bool GetRewardWeaponSkin; //보상받았는지
```

시즌패스 세팅&추상화클래스를 이용한 확장성고려

기타 기능 & 연출 제작(영상입니다)

미션&패키지&
연출 등등
여러가지 기능
제작

시스템,코드 구조 개선

이벤트 설계

```
public abstract class EventInterface
{
    참조 0개
    public EventInterface()
    {
        IsFirstLoad = true;
        IsGetData = false;
        EventManager.Instance.events.Add(this);
        Init();
    }
    참조 30개
    public virtual DateTime EndData { get; set; }
    참조 1개
    public virtual DateTime StartData
    {
        get
        {
            return new System.DateTime(1111, 11, 11, 11, 11, 11);
        }
        set
        {
            ;
        }
    }

    참조 5개
    public string RemainTime
    {
        get
        {
            return string.Format(LocalizationService.Instance.GetTextByKey("Text_End_Date_Detail"), EndData.Month, EndData.Day, 12, "00");
        }
    }
    참조 10개
    public abstract void Init();
    참조 15개
    public abstract void Setting();

    참조 59개
    public abstract void OnRedLight();

    참조 15개
    public virtual int AttendCnt { get; set; } //총 접속일
    참조 16개
    public virtual int AttendClear { get; set; } //몇일클리어했는지(Cnt보다 높으면안됨)
    참조 5개
    public virtual int TotalAttend { get; set; }
    참조 0개
    public virtual int Event_type { get; set; }

    참조 18개
    public virtual int Event_Index { get; set; }
    참조 56개
    public virtual int Event_Token { get; set; }
    참조 4개
    public virtual int Inapp_Type { get; set; }
    참조 4개
    public virtual int[] Inapp_Index { get; set; }
    참조 17개
    public virtual bool IsGetData { get; set; }
    참조 81개
    public virtual int[] Package_Count { get; set; }
```

여러 이벤트에서 공통적으로 사용되는 기능들을 만들때마다 새로만들어서 사용하는것이 비효율적이라 판단해 추상화클래스를 만들어 유지보수와 확장성을 높일수있다.

```
public override void Init()
{
    Package_Count = new int[3];
    Inapp_Type = 250;
    Inapp_Index = new int[2] { 5, 6 };
    ServerFunction.Instance.Event_Mission_Load(UserInfo.instance.userId, (string callback) =>
    {
        if (callback.Equals("OK"))
        {
            TotalAttend = 7;
            EndData = new System.DateTime(2025, 02, 10, 12, 0, 0);
            ReadNewYearMissionData();
            IsGetData = true;
            Debug.Log("설 이벤트 로딩");
        }
        else if (callback.Equals("END"))
        {
            Debug.Log("설 이벤트 종료");

            AttendCnt = 1;
            AttendClear = 1;
        }
        else
        {
            Debug.Log("Error");
        }
        IsGetData = true;
    });
}

참조 25개
public override void OnRedLight()
{
    for (int i = 0; i < 35; i++)
    {
        if (CanGetNewYearMissionReward(i) == 1)
        {
            ActiveRedlight(true);
            return;
        }
    }
```

```
public override void Init()
{
    Package_Count = new int[3];
    ServerFunction.Instance.Event_Special_Load(UserInfo.instance.userId, callback =>
    {
        if (callback.Equals("OK"))
        {
            EndData = new System.DateTime(2024, 12, 31, 12, 0, 0);
            CSV_Read();
            Debug.Log("크리스마스 데이터 로드 성공");
        }
        else
        {
            Debug.Log(callback);
            Debug.Log("Error");
        }
        IsGetData = true;
    });
}
```

이벤트 설계

```
public abstract class EventPopupInterface<T> : MonoBehaviour where T : EventInterface
{
    public InAppGoodsSlot[] InAppGoodsSlots;
    public List<List<InAppReward>> RewardList = new List<List<InAppReward>>();
    [HideInInspector]
    public GameObject RedLight;
    [HideInInspector]
    public T Eventpanel;
    참조 6개
    public abstract void Setting(GameObject _redLight = null);
    참조 6개
    public abstract void OnPanel();
    참조 7개
    public abstract void ClosePanel();
    참조 25개
    public abstract void CanOnRedLight();
    참조 2개
    public void SetEventPanelData(T data)
    {
        Eventpanel = data;
    }

    참조 18개
    public T GetEventPanelData()
    {
        return Eventpanel;
    }
    참조 2개
    public void SettingReward(params InAppReward[] _reward)
    {
        RewardList.Add(_reward.ToList());
    }
    참조 0개
    public void SettingInappSlot(int _index)
    {
        InAppGoodsSlots[_index].Init(RewardList[_index].ToArray());
    }
    참조 0개
    public void SetReward(int _index)
    {
        for (int i = 0; i < RewardList[_index].Count; i++)
        {
            RewardList[_index][i].SetReward();
        }
        GameUIManager.instance.rewardPanel.OpenPanel();
        CanOnRedLight();
        if (Eventpanel != null)
            Eventpanel.OnRedLight();
    }
    참조 4개
    public void SetRewardNoAction(int _index)
    {
        for (int i = 0; i < RewardList[_index].Count; i++)
        {
            RewardList[_index][i].SetReward();
        }
    }
}
```

T : EventInterface 제네릭을 사용하여 유연하게 이벤트 패널을 변경할 수 있도록 함.
이벤트패널의 확장성을 고려해서 추상화 클래스를 만듬.

```
public class SeasonPassPanel : EventPopupInterface<SeasonPassEvent>
```

```
public override void Setting(GameObject _redLight = null)
{
    SetEventPanelData(EventManager.Instance.SeasonPassEvent);

    for (int i = 0; i < SeasonPassPopupPanel.Length; i++)
    {
        SeasonPassPopupPanel[i].SetEventPanelData(EventManager.Instance.SeasonPassEvent);
        SeasonPassPopupPanel[i].Setting(categoryUIButtons[i].iconImage.gameObject);
    }

    titleText.text = LocalizationService.Instance.GetTextByKey("Text_SeasonPass") + " [" +
    rewardImage.sprite = Eventpanel.WeaponSkin.RewardImage();
    EventRewardInfo.text = LocalizationService.Instance.GetTextParseByKey("Text_SeasonPass
    //EventDataTime.text = LocalizationService.Instance.GetTextParseByKey("Text_EventPerio

    isFirst = false;
}
```

광고 보상 설계

광고를 보고난후 보상과 액션을 모든버튼마다 만들지않고
추상화 클래스를통해 확장성을 가짐

```
public abstract class AdsInterface : MonoBehaviour
{
    참조 2개
    public enum AdsType{...}
    참조 1개
    UIButton UIButton { get; set; }

    public AdsType Adstype;

    private GameObject SoldOutPanel;

    public delegate void AdsSuccessCallback();

    public AdsSuccessCallback successCallback;

    참조 4개
    public abstract void CallAds();
    참조 5개
    public virtual void ComponentSetting()
    {
        if (SoldOutPanel == null)
        {
            SoldOutPanel = this.transform.Find("Sold Out Panel").gameObject;
        }
        if (UIButton == null)
        {
            UIButton = GetComponent<UIButton>();
        }
    }
    참조 10개
    public virtual void Init()
    {
        if (Adstype == AdsType.Etc) return;
        if (SoldOutPanel != null)
        {
            SoldOutPanel.SetActive(UserInfo.instance.userShopAdInfo[(int)Adstype - 1]);
        }
        if (UIButton != null)
        {
            if (UIButton.button != null)
            {
                UIButton.button.interactable = !UserInfo.instance.userShopAdInfo[(int)Adstype - 1];
            }
        }
    }
    참조 3개
    public virtual void Setting()
    {
        CallAds();
        if (UIButton.button.onClick.GetPersistentEventCount() > 0)
        {
            UIButton.button.onClick.RemoveAllListeners();
        }
        if (successCallback == null)
        {
            if (UserInfo.instance.userPackageInfo[0] == 1)
            {
                UIButton.button.onClick.AddListener(() => AdsManager.instance.Ads_Free((int)Adstype, Init));
            }
            else
            {
                UIButton.button.onClick.AddListener(() => AdsManager.instance.Show_Ads_Reward_Action((int)Adstype, Init));
```

```
public class ArenaAdsButton : AdsInterface
{
    참조 2개
    public override void CallAds()
    {
        successCallback = SettingCallback;

    }
    참조 9개
    public override void Init()
    {
        ComponentSetting();
        base.Init();
        base.Setting();
    }
    참조 1개
    void SettingCallback()
    {
        GameUIManager.instance.pvpPanel.pvpRankingPanel.SetStartButton();
        base.Init();
    }
}
```

```
public class SquadBlessedAdsButton : AdsInterface
{
    참조 2개
    public override void CallAds()
    {
        successCallback = null;
    }
    참조 9개
    public override void Init()
    {
        ComponentSetting();
        base.Init();
        base.Setting();
    }
}
```

프로파일러를 통한 최적화

```
private IEnumerator AutoSleepModeTimerCoroutine(int _idx)
{
    int tempCnt = 0;

    if (_idx == 1)
    {
        tempCnt = 5;
    }
    else // 2.
    {
        tempCnt = 15;
    }

    if (ScenesManager.instance.sceneIdx != 1)
        yield break;

    if (isSleepMode)
        yield break;

    for (int i = 0; i < tempCnt; i++)
    {
        yield return oneMin2;

        //Debug.Log(System.DateTime.Now - GameUIManager.instance.doubleRaycast.lastInputTime);

        if (GameUIManager.instance != null && GameModeManager.instance.monsterSpawnManager.contentType == MonsterSpawnManager.ContentType.Stage && GameModeManager.instance.
        {
            calcTime = System.DateTime.Now - GameUIManager.instance.doubleRaycast.lastInputTime;

            if (_idx == 1 && calcTime.TotalSeconds >= 300f || _idx == 2 && calcTime.TotalSeconds >= 900f)
            {
                autoSleepContainer = null;

                OpenPanel();
                yield break;
            }
            else
            {
                if (i == tempCnt - 1)
                    autoSleepContainer = StartCoroutine(AutoSleepModeTimerCoroutine(_idx));
            }
        }
        else if (GameUIManager.instance != null && GameModeManager.instance.monsterSpawnManager.contentType == MonsterSpawnManager.ContentType.Christmas && GameModeManager.
        {
            calcTime = System.DateTime.Now - GameUIManager.instance.doubleRaycast.lastInputTime;

            if (_idx == 1 && calcTime.TotalSeconds >= 300f || _idx == 2 && calcTime.TotalSeconds >= 900f)
            {
                autoSleepContainer = null;
                OpenPanel();
                yield break;
            }
            else
                autoSleepContainer = StartCoroutine(AutoSleepModeTimerCoroutine(_idx));
        }
        else
            autoSleepContainer = StartCoroutine(AutoSleepModeTimerCoroutine(_idx));
    }
}
```

```
private IEnumerator AutoSleepModeTimerCoroutine(int _idx)
{
    int temp = startCnt++;

    int tempCnt = 0;

    if (_idx == 1)
    {
        tempCnt = 5;
    }
    else // 2.
    {
        tempCnt = 15;
    }

    if (ScenesManager.instance.sceneIdx != 1)
        yield break;

    if (isSleepMode)
        yield break;

    for (int i = 0; i < tempCnt; i++)
    {
        yield return oneMin2;

        //Debug.Log(System.DateTime.Now - GameUIManager.instance.doubleRaycast.lastInputTime);

        if (GameUIManager.instance != null && GameModeManager.instance.monsterSpawnManager.contentType == MonsterSpawnManager.ContentType.Stage && GameModeManager.instance.isStageLoop)
        {
            calcTime = System.DateTime.Now - GameUIManager.instance.doubleRaycast.lastInputTime;

            if (_idx == 1 && calcTime.TotalSeconds >= 300f || _idx == 2 && calcTime.TotalSeconds >= 900f)
            {
                autoSleepContainer = null;

                OpenPanel();
                yield break;
            }
            else
            {
                if (i == tempCnt - 1)
                {
                    StartAutoSleepTimer(_idx, true);
                    yield break;
                }
            }
        }
        else if (GameUIManager.instance != null && GameModeManager.instance.monsterSpawnManager.contentType == MonsterSpawnManager.ContentType.Christmas && GameModeManager.instance.christmasManager.isAuto)
        {
            calcTime = System.DateTime.Now - GameUIManager.instance.doubleRaycast.lastInputTime;

            if (_idx == 1 && calcTime.TotalSeconds >= 300f || _idx == 2 && calcTime.TotalSeconds >= 900f)
            {
                autoSleepContainer = null;
                OpenPanel();
                yield break;
            }
            else
            {
                if (i == tempCnt - 1)
                {
                    StartAutoSleepTimer(_idx, true);
                    yield break;
                }
            }
        }
        else
        {
            if (i == tempCnt - 1)
            {

                StartAutoSleepTimer(_idx, true);
                yield break;
            }
        }
    }
```

프로파일러를 통해 메모리가 과도하게 누적되는부분을찾아 최적화

Enemy 탐색

```
public void TargetCheck() //가장 가까운 대상을 지정합니다. => 추후 대상이 죽었을때만 타겟 체크 하는걸로
{
    if (GameModeManager.instance.GetCurrentStageEnemyList().Count == 0 && GameModeManager.instance.raidManager.raidBoss == null)
    {
        target = null;
        return;
    }
    //내가 수동으로 이동하는 경우가 아니라면?

    float minDis = 999999f; //제일 가까운 거리
    float tempDis;


    for (int i = 0; i < GameModeManager.instance.GetCurrentStageEnemyList().Count; i++) //모든 대상을 찾고 최단거리를 계산한다.
    {
        tempDis = Vector3.SqrMagnitude(GameModeManager.instance.GetCurrentStageEnemyList()[i].transform.position - transform.position);

        if (GameModeManager.instance.GetCurrentStageEnemyList()[i].isLast && GameModeManager.instance.GetCurrentStageEnemyList().Count > 1) //마지막일 때만 타겟
            continue;


        if (minDis > tempDis && GameModeManager.instance.GetCurrentStageEnemyList()[i].state != EnemyStatus.CharacterState.Dead && GameModeManager.instance.GetC
        {
            minDis = tempDis;

            if (GameModeManager.instance.GetCurrentStageEnemyList()[i].isRadius)

                myStatus.targetRadius = GameModeManager.instance.GetCurrentStageEnemyList()[i].isRadiusValue * 5;
            else
                myStatus.targetRadius = 0;

            target = GameModeManager.instance.GetCurrentStageEnemyList()[i];
        }
    }

}
```

```
public void TargetCheck() //가장 가까운 대상을 지정합니다. => 추후 대상이 죽었을때만 타겟 체크 하는걸로
{
    if (GameModeManager.instance.GetCurrentStageEnemyList().Count == 0 && GameModeManager.instance.raidManager.raidBoss == null)
    {
        target = null;
        return;
    }

    float searchRadius = 30f;
    LayerMask enemyLayer = LayerMask.GetMask("Enemy");

    Collider[] colliders = Physics.OverlapSphere(transform.position, searchRadius, enemyLayer);
    float minDis = Mathf.Infinity;
    EnemyStatus closestEnemy = null;

    foreach (Collider col in colliders)
    {
        EnemyStatus enemy = col.GetComponent<EnemyStatus>();
        if (enemy == null || enemy.state == EnemyStatus.CharacterState.Dead || enemy.isCannotAttack)
            continue;

        float tempDis = Vector3.SqrMagnitude(enemy.transform.position - transform.position);
        if (tempDis < minDis)
        {
            minDis = tempDis;
            closestEnemy = enemy;
        }
    }

    target = closestEnemy;
}
```

기존엔 거리가 멀어도 체크하게되는데 일정거리안에서만 적을 찾는시스템으로 최적화작업

소개 영상 :

https://www.youtube.com/watch?v=LQJtbnbTZDU

BJ 영상 :

https://www.youtube.com/watch?v=C7kgGJjabWY&t=552s

https://www.youtube.com/watch?v=Q177Jg7zF-M