Planet explore HIVE 적용 게임 개발

최초 작성일 : 2018.11.19

수정일 : 2020.12.15

대외비

목 차

● HIVE SDK 시작하기

a. HIVE SDK 란?

i. HIVE는 모바일 게임을 글로벌 유저에게 서비스하기 위해 필요한 각종 기능을 제공하는 플랫폼입니다.

ii. HIVE를 이용하면 모바일 게임을 개발하는데 소요되는 시간을 단축시키고 게임 유저를 효과적으로 증가시키며 유저들의 참여를 촉진할 수 있습니다.

iii. 현재 HIVE 모바일 게임 플랫폼은 게임빌과 컴투스에서 퍼블리싱하는 게임에서 이용할 수 있습니다.

iv. HIVE를 이용해 모바일 게임을 서비스하기 위해서는 게임 클라이언트에 HIVE SDK를 통합하는 작업이 필요합니다.

v. HIVE SDK는 iOS와 Android 단말 플랫폼을 지원하며, C++, Objective-C 언어와 Unity, Unreal Engine, Cocos2d-x 게임엔진 지원에 필요한 플러그인 3종을 제공합니다.

vi. 이 문서에서 소개하는 Planet explore 샘플앱은 Unity 플러그인을 사용했습니다.

vii. 참고: http://developers.withhive.com/about-hive/

b. 알아 두어야 할 것들

i. 샘플앱 홈 화면(Home Scene)에는 게임 타이틀이나 로그인 같은 HIVE 연동 항목이, 메인 화면(Main Scene)에는 게임 로직 부분이 주로 구현되었습니다.

ii. 앱 초기화는 HIVE Set-up, HIVE Sign-In, Game Sign-In이 순서대로 완료되어야 다음 단계를 진행합니다.

iii. 최초 로그인 시에는 각 OS 의 플랫폼 IdP(Google의 Play Game Services, Apple의 Game Center)로 바로 로그인을 시도하고(묵시적 로그인) 이후 로그아웃 등으로 로그인 화면에 돌아왔을 경우 IdP 선택창이 뜨도록 구현되었습니다.

iv. 각 OS의 플랫폼에서 제공하는 업적과 리더보드를 사용할 때는 HIVE PlayerID에 연동된 IdP ID와 로그인한 IdP ID를 동기화하도록 구현했습니다.
(HIVEManager.cs의 OnApplicationFocus()에서 AuthV4.Helper.syncAccount())

v. 위 ii. iii. iv.에서 설명한 기능을 쉽게 구현하도록 도와주는 ***AuthV4.Helper*** 를 사용했습니다.

vi. HIVE API를 호출하는 게임 로직은 녹색 상자로, HIVE 콜백 처리는 파란 상자로 표시되어있습니다.

vii. Unity 2018.4.28f1 버전으로 개발되었습니다.

viii. HIVE SDK v4.15.0.2 KS 버전으로 개발되었습니다.

1. Planet explore

Planet explore은 게임 개발 스튜디오에서 HIVE SDK를 손쉽게 적용할 수 있도록 가이드 역할을 제공하기 위해 GAMEVIL COM2US PLATFORM에서 제작한 샘플 게임입니다.

본 게임에는 HIVE SDK가 제공하는 인증, 설정, 캠페인, 이벤트, 빌링, 노티피케이션 기능이 적용되어 있습니다.

게임 개발환경은 아래와 같습니다.

a. Unity 버전: Unity 2018.4.28f1 - 다운로드
b. HIVE SDK 버전: HIVE SDK 4.15.0.2 KS - 다운로드
c. 지원 플랫폼: iOS, Android

2. Unity 빌드

본 가이드는 Mac 운영체제 기준으로 작성되었으며, Unity 설치와 라이선스에 관한 내용은 다루지 않습니다.

Unity 빌드 가이드는 HIVE 개발자 사이트 내 Unity 설정 페이지에서도 확인할 수 있습니다.

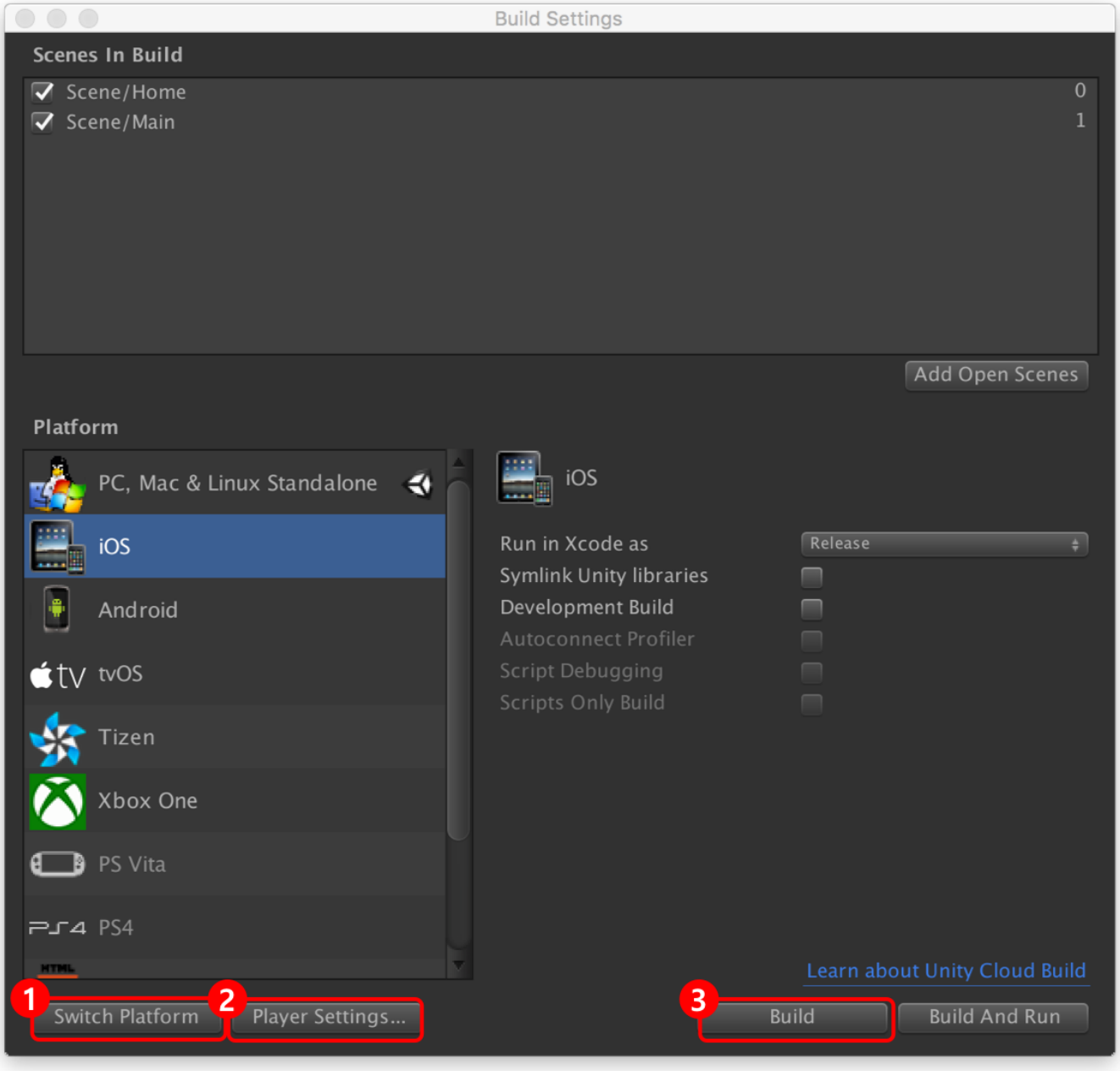
~~a. Blender~~

~~i. 캐릭터의 모델링은 Blender를 이용하여 구현되었으며 이를 Unity에서 사용하기 위해서는 빌드 머신에 Blender가 설치되어 있어야 합니다.~~
~~ii. Blender가 설치되어있지 않은 경우, SpaceMan 캐릭터가 렌더링 되지 않습니다.~~
~~iii. Blender는 다운로드 페이지에서 설치파일을 다운로드 할 수 있으며 설치과정은 다른 DMG 패키지 설치와 동일하기 때문에 생략합니다.~~
iv. Blender 사용부분 제거

b. iOS 빌드

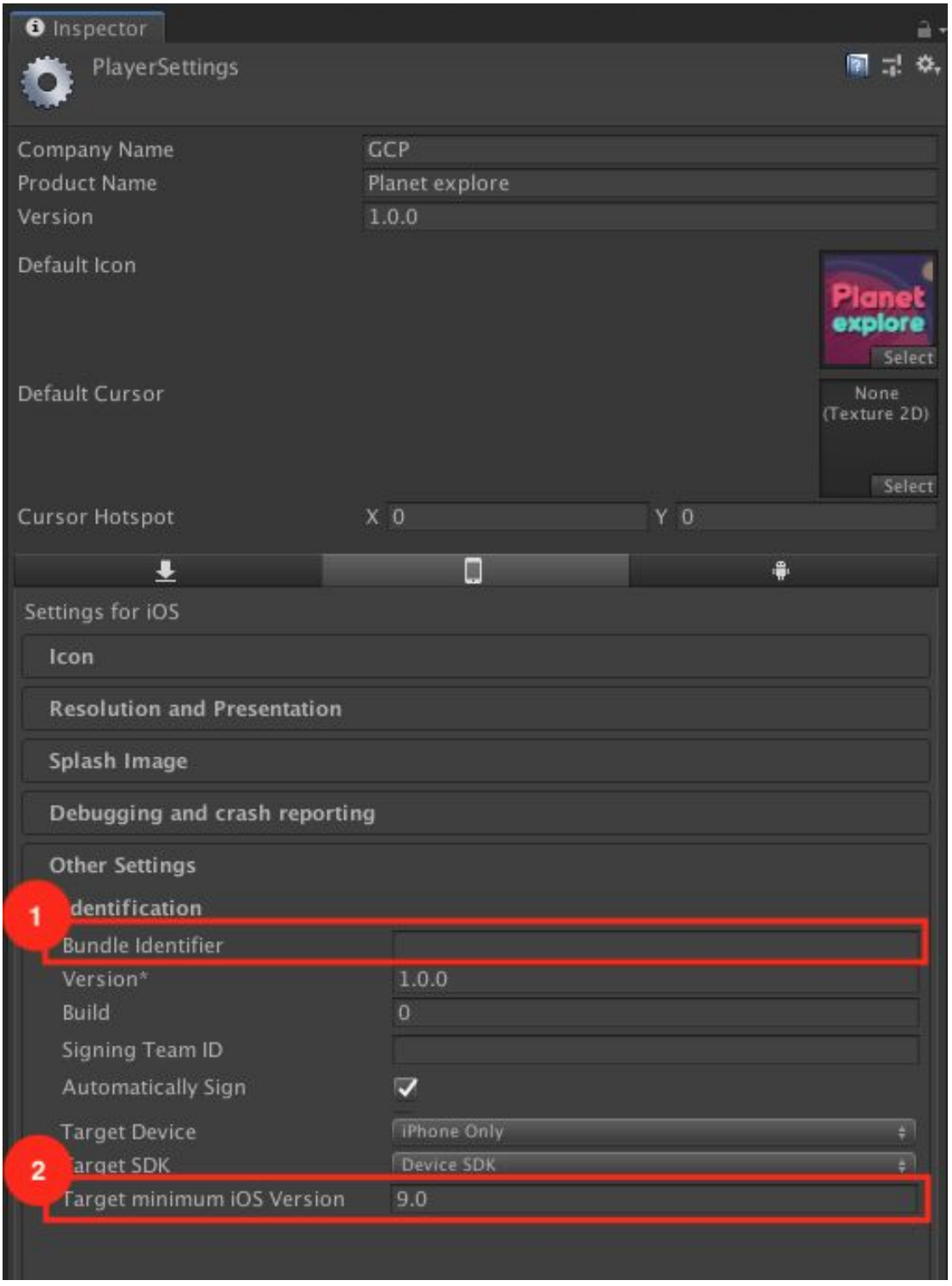
i. Import 기능으로 프로젝트를 불러온 후 **File > Build Settings(⇧⌘B)** 메뉴를 선택합니다. Build Settings 내 Platform 항목에서 **iOS**를 선택하고 **Switch Platform** 버튼을 클릭해 프로젝트 플랫폼을 iOS로 변경합니다. (그림 1 설명 1)
ii. 변경 후 **Player Settings** 버튼을 클릭해 Inspector 창을 표시합니다. (그림 1 설명 2)
iii. Inspector 창 Other Settings 항목에 발급 받은 Bundle Identifier와 HIVE가 지원하는 최소 iOS 버전을 입력합니다. 게임에 적용된 HIVE SDK 4.15.0.2 KS은 iOS 10.0부터 지원합니다. (그림 2 설명 1, 설명 2)
iv. HIVE SDK를 설정하기 위해 **Hive > Edit Config** 메뉴를 선택, HIVE Inspector를 띄우고 Android와 iOS 중 **iOS** 탭을 선택합니다. (그림 3 설명 1)
v. HIVE App ID 항목에 이전에 입력한 Bundle Identifier를 입력합니다. (그림 3 설명 2)
vi. Planet explore는 Facebook 연동을 지원하므로 Facebook Settings 항목에 Facebook App ID를 입력합니다. (그림 3 설명 3)
vii. 입력된 값으로 iOS와 HIVE 설정 파일을 생성하기 위해 **Regenerate iOS Plist & Hive Config** 버튼을 클릭합니다. (그림 3 설명 4)

viii. Xcode 프로젝트를 빌드하기 위해 Build Settings 창에서 **Build** 버튼을 클릭합니다. (그림 1 설명 3)



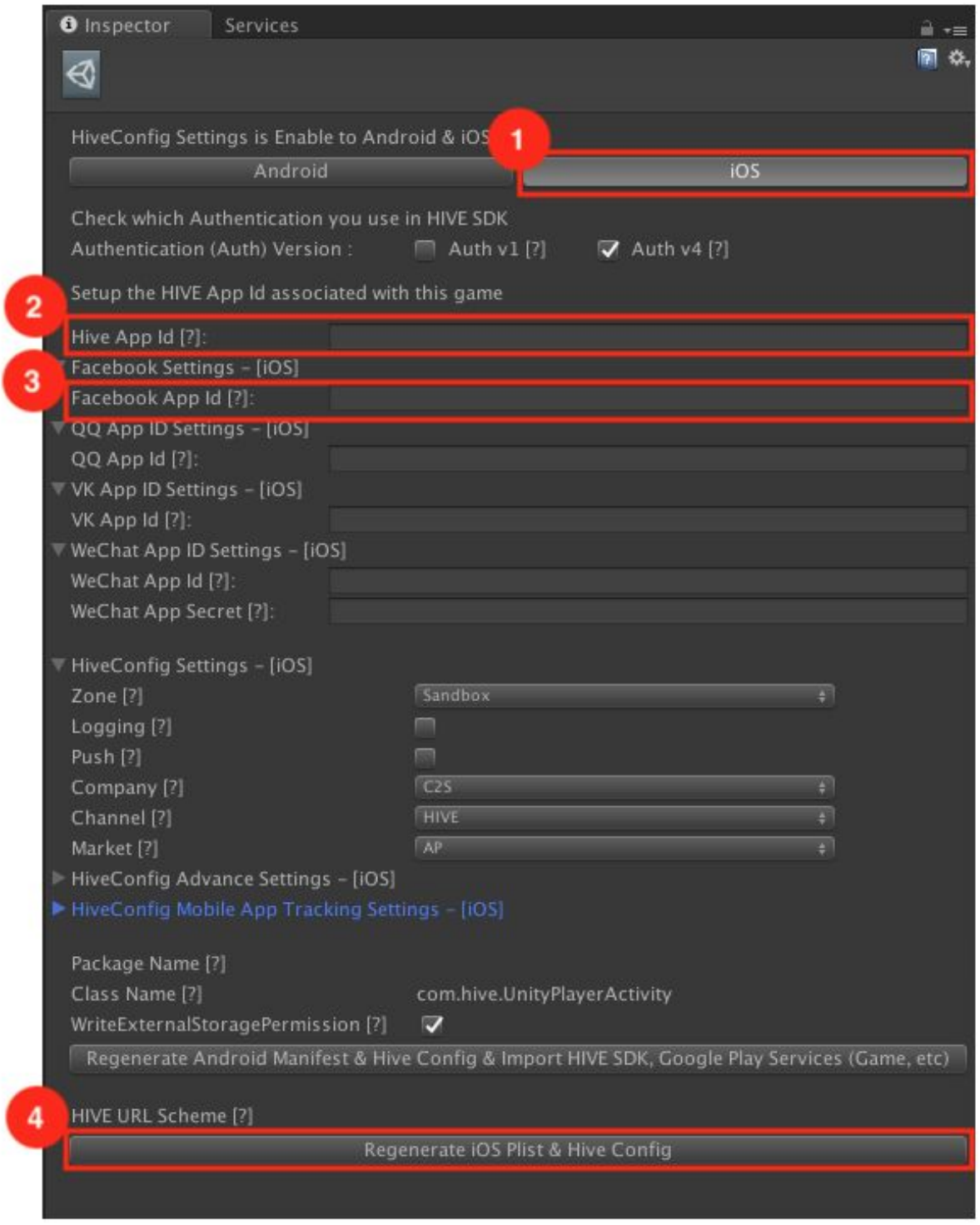
[그림 1] Unity Build Setting

1) 선택한 플랫폼으로 프로젝트를 전환한다.
2) 해당 플랫폼에 필요한 설정을 할 수 있는 Inspector 창을 표시한다.
3) iOS 프로젝트로 빌드한다.



[그림 2] Unity Inspector

1) 발급 받은 Bundle Identifier를 입력한다.
2) 지원하는 최소 iOS 버전을 입력한다.

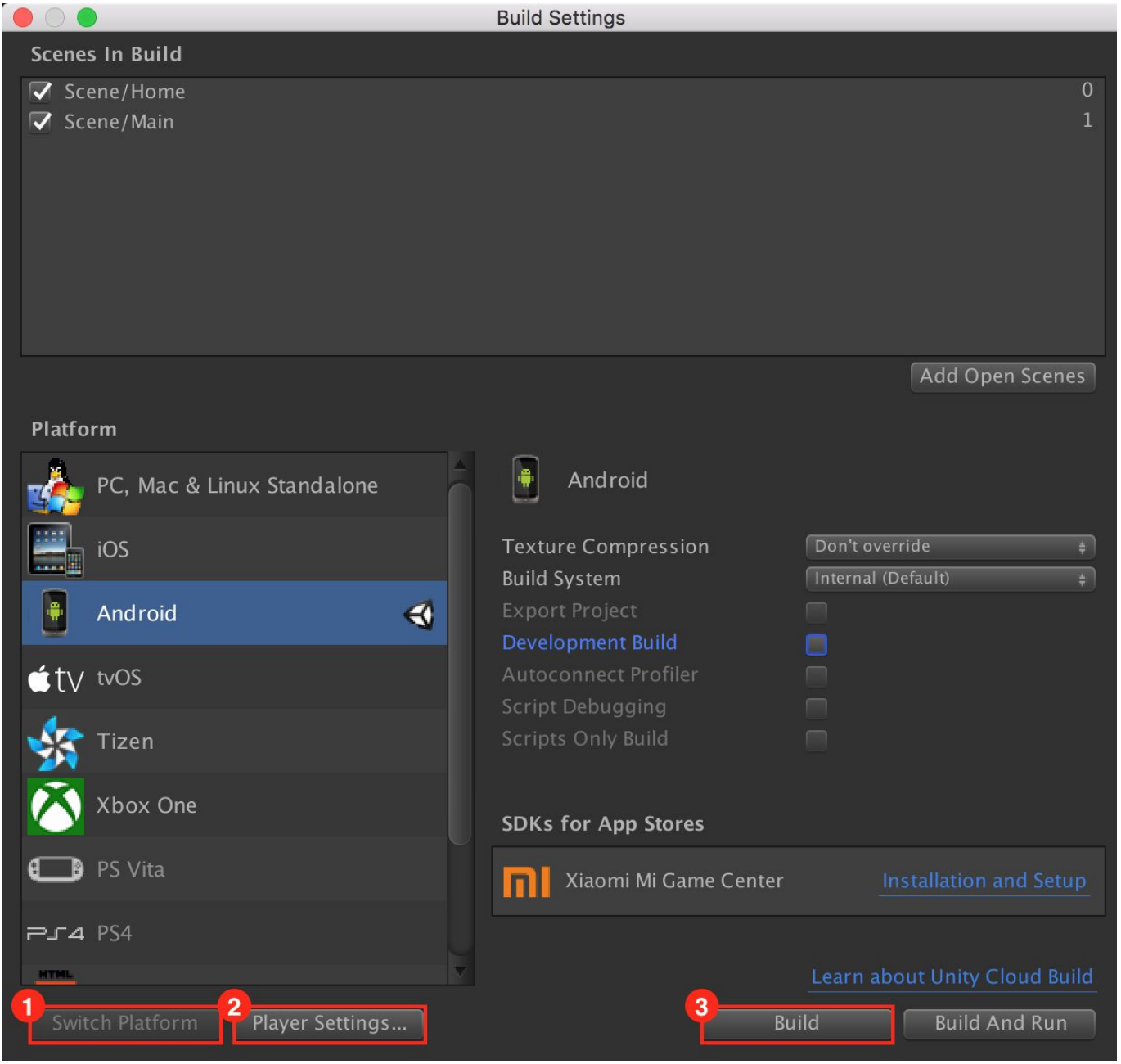


[그림 3] HIVE Inspector

1) iOS 플랫폼 설정 시 iOS로 선택한다.
2) Unity Inspector에서 기재한 App ID를 기재한다.
3) Facebook App ID를 기재한다.
4) 입력한 설정 값이 적용된 iOS Plist와 HIVE 설정 파일을 생성한다.

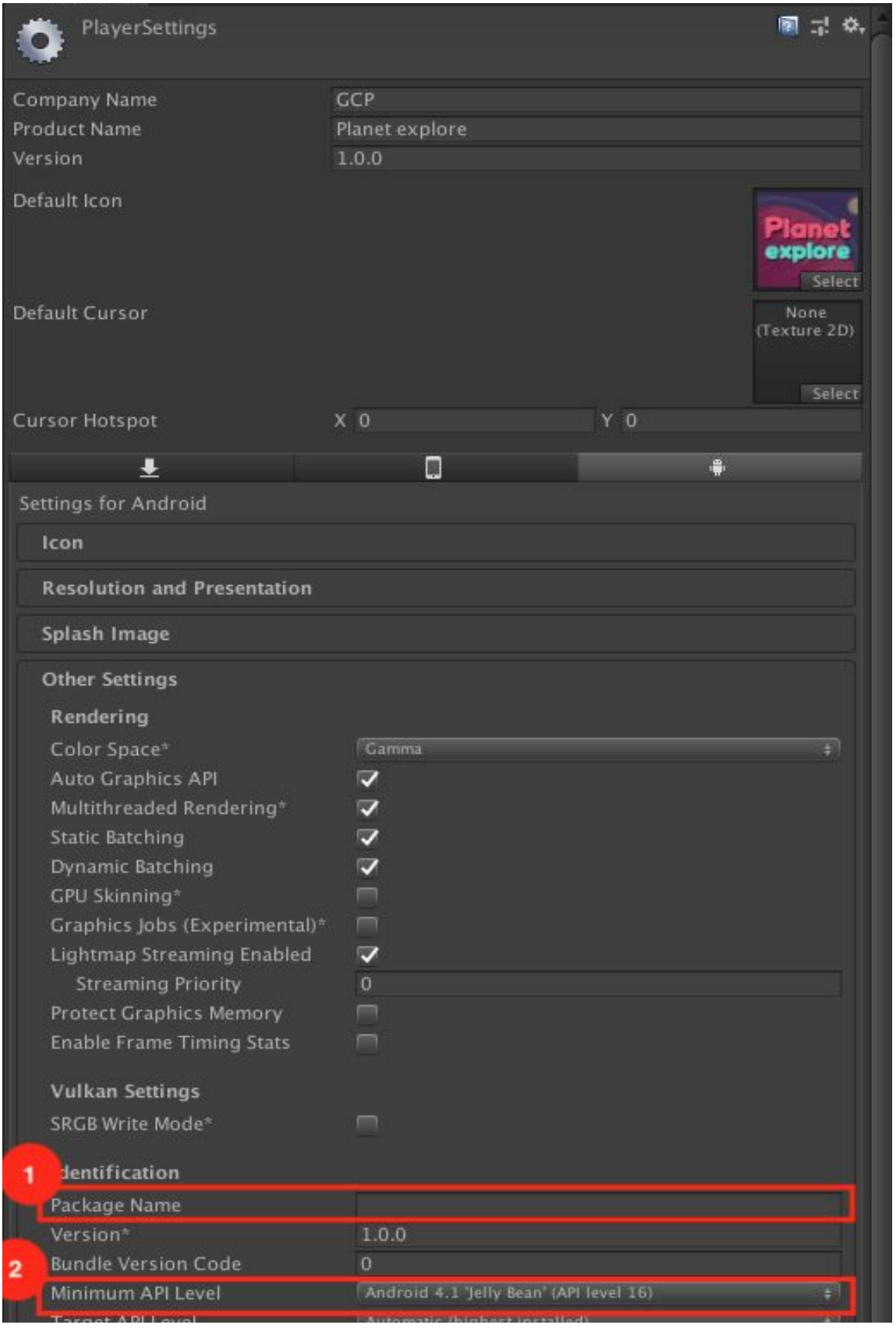
c. Android 빌드

i. Import 기능으로 프로젝트를 불러온 후 **File > Build Settings(⇧⌘B)** 메뉴를 선택합니다. Build Settings 내 Platform 항목에서 **Android**를 선택하고 **Switch Platform** 버튼을 클릭해 프로젝트 플랫폼을 Android로 변경합니다. (그림 4 설명 1)
ii. 변경 후 **Player Settings** 버튼을 클릭해 Inspector 창을 표시합니다. (그림 4 설명 2)
iii. Inspector 창 Other Settings 항목에 발급 받은 Package Name과 HIVE가 지원하는 최소 iOS 버전을 입력합니다. 게임에 적용된 HIVE SDK 4.15.0.2 KS은 Android 4.1 (API Level 16)부터 지원합니다. (그림 5 설명 1, 설명 2)
iv. HIVE SDK를 설정하기 위해 **Hive > Edit Config** 메뉴를 선택, HIVE Inspector를 띄우고 Android와 iOS 중 **Android** 탭을 선택합니다. (그림 6 설명 1)
v. HIVE App ID 항목에 이전에 입력한 Package Name을 입력합니다. (그림 6 설명 2)
vi. Planet explore는 Facebook 연동을 지원하므로 Facebook Settings 항목에 Facebook App ID를 입력합니다. (그림 6 설명 3)
vii. 입력된 값으로 iOS와 HIVE 설정 파일을 생성하기 위해 **Regenerate Android Manifest & Hive Config & Import HIVE SDK, Google Play Services (Game, etc)** 버튼을 클릭합니다. (그림 6 설명 4)
viii. APK를 생성하기 위해 Build Settings 창에서 **Build** 버튼을 클릭합니다. 기기에 바로 실행하려면 **Build And Run** 버튼을 클릭합니다.(그림 4 설명3)



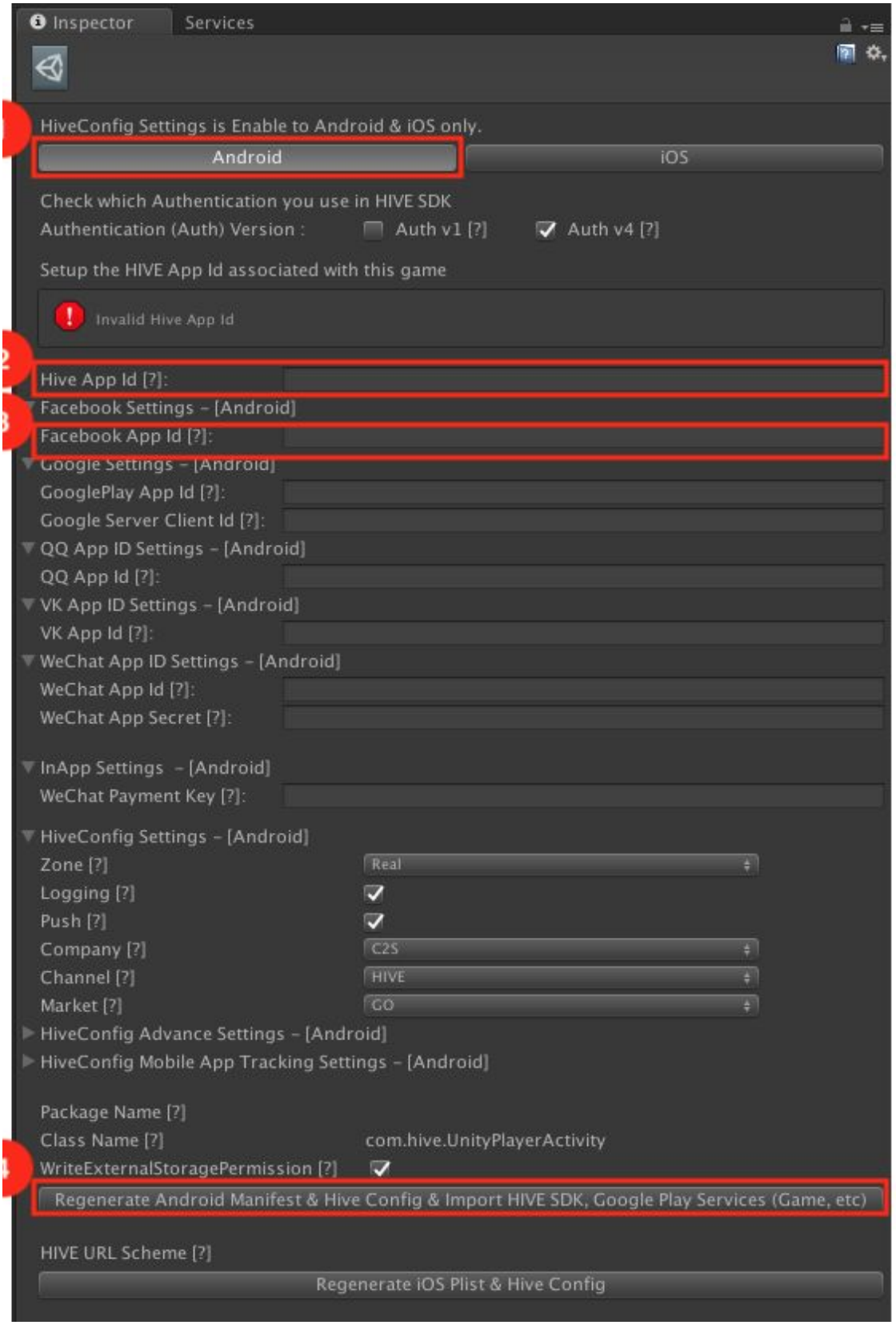
[그림 4] Unity Build Setting

1) 선택한 플랫폼으로 프로젝트를 전환한다.
2) 해당 플랫폼에 필요한 설정을 할 수 있는 Inspector 창을 표시한다.
3) Android 프로젝트로 빌드한다.



[그림 5] Unity Inspector

1) 발급 받은 Package Name을 입력한다.
2) 지원하는 최소 Android 버전을 입력한다.



[그림 6] HIVE Inspector

1) Android 플랫폼 설정시 Android로 선택해야 한다.
2) Unity Inspector에서 기재한 App ID를 기재한다.
3) Facebook App ID를 기재한다.
4) 입력된 설정값이 적용된 Android Manifest와 HIVE 설정 파일을 생성한다.

3. 기능

샘플 게임 홈 화면(Home Scene)에는 게임 타이틀이나 인증 같은 HIVE 연동 항목이, 메인 화면(Main Scene)에는 게임 로직이 구현되었습니다.

샘플 게임의 **HIVEManager.cs**에는 HIVE SDK API를 직접 호출하는 부분이 구현되어 있습니다.

예시 코드에서 녹색 테두리는 HIVE API 를 호출하는 게임 로직을 의미하며, 파란 상자는 API 콜백 처리를 의미합니다. **굵은 글씨**는 HIVE API를 호출하는 라인입니다.

자세한 정보는 참고자료 > c. HIVE 개발자 사이트에서 확인할 수 있습니다.

a. 권한 고지 커스터마이징

HIVE SDK 4.11.2 에 새로 추가된 권한 고지 커스터마이징 API를 활용하여 게임에 어울리는 디자인의 권한 고지 알림을 노출 할 수 있습니다. HIVE SDK 초기화 이전에 해당 API를 호출하여 권한 고지에 필요한 정보를 얻어옵니다.

i. 구현 파일

1. HIVEManager.cs
2. Permission.cs

ii. HIVE API

API	설명
AuthV4.requestPErmissionViewData()	권한 고지 UI 노출 시 필요한 데이터를 요청합니다.

iii. 예시 코드

```
// Permission.cs: 권한 고지 팝업 데이터 요청

public void gerPermissionData()
{
        HIVEManager.shared.requestPermissionViewData(onHIVEManagerRequestPermissionEvent);
}
```

```
// HIVEManager.cs: 권한 고지 팝업 데이터 요청

public void requestPermissionViewData(onHIVEManagerRequestPermissionEvent listener)
{
        onHIVEManagerRequestPermissionViewData = listener;

        AuthV4.requestPermissionViewData(onPermissionViewData);
}
```

```
// HIVEManager.cs:  권한 고지 팝업 노출

void onPermissionViewData(ResultAPI result, PermissionViewData data)
{
        Debug.Log ("onPermissionViewData : " + result.toString());
```

```
        if (result.isSuccess() == true) {
                // 상태 변경 없음.
                if (result.code == ResultAPI.Code.AuthV4SkipPermissionView) {
                        // 권한 고지 팝업 노출 없음
                } else {
                        // 권한 고지 팝업 노출
                }
        } else {
                // no operation.
        }
}
```

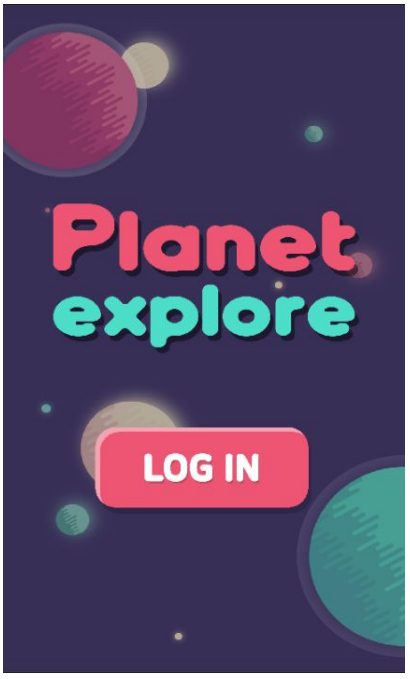
b. 초기화, 로그인

게임을 처음 실행하면 [그림 7] 화면이 표시됩니다. 여기에서 **Login** 버튼을 클릭하면 앱을 초기화합니다. 앱 초기화는 HIVE Set-up, HIVE Sign-In, Game Sign-In 순서로 구현되어 있습니다. 만약 초기화 진행 도중 실패하면 다음 클릭에서 실패한 단계부터 다시 진행됩니다.

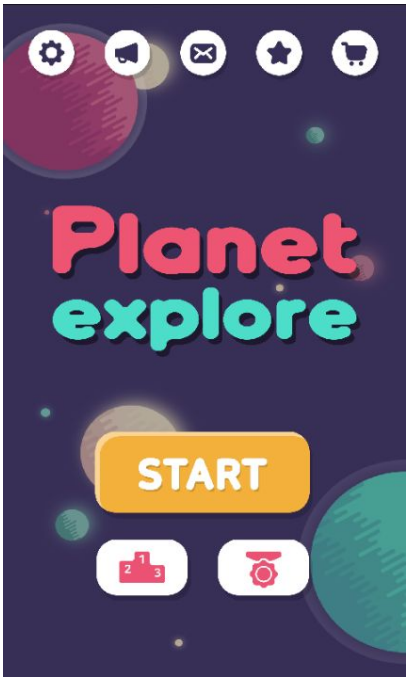
로그인 기능은 HIVE SDK 4.7.0에 새로 추가된 AuthV4Helper를 사용해 구현되었습니다. 최초 로그인 요청 시 묵시적 로그인(Apple Game Center, Google Play Game Services)을 시도하며 로그인 실패나 로그아웃 후 로그인 시에는 명시적 로그인(묵시적 로그인 + Facebook, HIVE Membership) 창을 표시합니다.

로그인에 성공하면 이후 로그인 요청 시 기존에 로그인한 세션으로 자동 로그인되는데 이 때 계정간 충돌이 발생할 수 있습니다. 계정 충돌이란 게임에 로그인한 Game Center 또는 Play Game Services 계정과 현재 사용하는 기기에 로그인 된 계정이 다른 경우에 발생합니다. 계정이 충돌했을 때는 계정 선택 팝업을 표시해 해결할 수 있습니다.

앱 초기화가 정상적으로 끝나면 [그림 8] 화면으로 넘어가게 되며, 이후 게임을 진행하거나 다른 기능을 사용할 수 있습니다.



[그림 7] 로그인 화면



[그림 8] 로그인 성공 화면

i. 구현 파일

1. HIVEManager.cs
2. HomeManager.cs

ii. HIVE API

API	설명
AuthV4.setup()	HIVE SDK를 설정합니다
AuthV4.Helper.signIn()	로그인을 요청합니다
AuthV4.Helper.syncAccount()	계정 충돌 여부를 검사합니다
AuthV4.Helper.showconflict()	계정 충돌 처리 팝업을 표시합니다

iii. 예시 코드

```
// HomeManager.cs: SDK 초기화

public void onStartClick()
{
	if (HIVEManager.shared.isSetup == false) {
		Debug.Log ("HIVE SDK 초기화 요청");
		// HIVE SDK 초기화 요청
		HIVEManager.shared.setup(onHIVEManagerSetup);

	} else if (HIVEManager.shared.isSignIn == false) {
		Debug.Log ("HIVE SDK SIGN IN");
		HIVEManager.shared.signIn(onHIVEManagerSignIn);

	} else if (!IsGameSignIn) {
		Debug.Log ("GAME SIGN IN");
		GameSignIn();

	} else {
		Debug.Log ("LoadScene \"Main\"");

		SceneManager.LoadScene("Main");
	}
}
```

```
// HIVEManager.cs: SDK에 인증 요청

public void signIn(onHIVEManagerEvent listener)
{
	Debug.Log("HIVEManager - signIn");

	if (_isSignIn == true) {
		Debug.Log("HIVEManager - Already did signin.");

		listener(true, "Success");
		return;
	}

	onHIVEManagerSignIn += listener;

	AuthV4.Helper.signIn(onAuthV4HelperSignIn);
}
```

```
// HIVEManager.cs: 계정 충돌 검사

void OnApplicationFocus(bool hasFocus)
{
	_isPaused = !hasFocus;

	if (_isPaused == false) {
		if (_isResumeProcess == false) {
			_isResumeProcess = true;

			AuthV4.Helper.syncAccount(AuthV4.ProviderType.AUTO,
onAuthV4HelperSyncAccount);
		}
	}
}
```

```
// HIVEManager.cs: 계정 충돌 검사 결과 처리

void onAuthV4HelperSyncAccount(ResultAPI result, AuthV4.PlayerInfo playerInfo)
{
	Debug.Log ("onAuthV4HelperSyncAccount : " + result.toString());

	if (result.isSuccess() == true) {
		// 상태 변경 없음.
	} else if (result.errorCode == ResultAPI.ErrorCode.CONFLICT_PLAYER) {
		// 다른 계정으로 로그인함
		AuthV4.Helper.showConflict(onAuthV4HelperShowConflict);
	} else {
		// no operation.
	}

	_isResumeProcess = false;
}
```

c. 설정

설정 화면(그림 6)에서는 로그인한 유저 정보(Player ID, 프로필 이미지, 플레이어 이름, 게임 서버)를 표시하고 게임에서 지원하는 Provider에 계정을 연동하는 기능을 제공합니다. 또한 계정 변경에 필요한 로그아웃, 1:1 문의, HIVE 카페, 약관 보기 기능이 구현되어 있습니다.

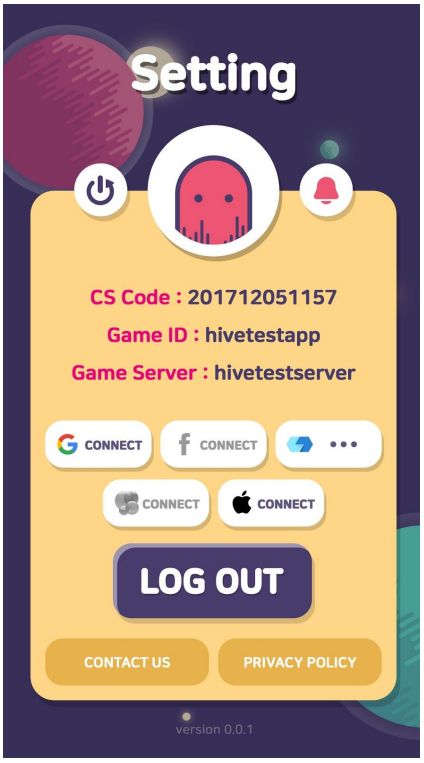
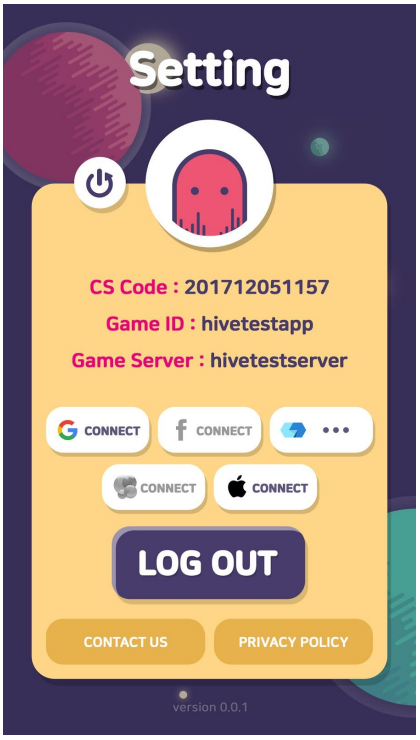
유저 정보는 프로필 이미지, CS Code, Nick Name, Language 항목을 보여줍니다. 프로필 이미지는 사용자 정보의 Player Image URL 값을 나타내며 CS Code에는 Player ID, Nick Name에는 Player Name를 나타냅니다.

CONNECT 버튼은 Provider의 연결 상태를 나타냅니다. 비활성화 상태에는 지원하는 Provider에 연결할 수 있습니다. 현재 게임에서 지원하는 Provider는 Game Center/Play Game Services, Facebook, HIVE, LINE, Apple 로그인입니다. Provider에 연동할 때 이미 다른 Player ID에 연결되어 있어 계정 충돌이 발생하는 경우, 로그인할 때 발생하는 계정 충돌 상황과 마찬가지로 HIVE UI를 활용해 해결할 수 있습니다. Provider에 연결된 상태에서 **DISCONNECT** 버튼을 클릭하면 연결이 해제됩니다.

COPPA 법률에 따라 사용자가 13세 미만일 경우에는 알림 버튼을 숨기고, Provider 버튼을 모두 비활성화합니다. 이 부분은 게임에서 처리하며, AuthV4.setup() 이후 Configuration.getAgeGateU13() 값을 이용합니다.

LOG OUT 버튼을 클릭하면 게임 유저 정보를 초기화하고 설정창을 닫은 후 로그인 화면으로 돌아갑니다. 이 때 로그인을 요청하면 Provider 목록을 보여주는 명시적 로그인이 진행됩니다.

CONTACT US 버튼은 1:1 문의, **PRIVACY POLICY** 버튼은 약관 다시보기 기능과 연결됩니다.

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>[그림 9] 기본 설정 화면</td>
<td>[그림 10] 유저가 13세 미만일 때 설정 화면</td>
</tr>
</table>

i. 구현 파일

1. HIVEManager.cs
2. Settings.cs

ii. HIVE API

API	설명
AuthV4.getPlayerInfo()	현재 유저 정보를 불러옵니다
AuthV4.Helper.connect()	새로운 Provider에 연결합니다
AuthV4.Helper.disconnect()	연결된 Provider를 해제합니다
AuthV4.Helper.signOut()	로그아웃합니다

AuthV4.showInquiry()	1:1 문의 화면을 출력합니다
AuthV4.showTerms()	약관을 출력합니다
Configuration.getHiveCountry()	유저의 접속 국가를 조회합니다
AuthV4.getAgeGateU13()	13세 미만 유저 제한 여부를 조회합니다.

iii. 예시 코드

```
// Settings.cs: 유저 정보 표시

StartCoroutine(downloadTexture(AuthV4.getPlayerInfo().playerImageUrl));

textCSCode.text = "" + AuthV4.getPlayerInfo ().playerId;
textGameID.text = AuthV4.getPlayerInfo ().playerName;
textGameServer.text = Configuration.getHiveCountry();
```

```
// Settings.cs: Provider 연결 상태 갱신

Dictionary<AuthV4.ProviderType, AuthV4.ProviderInfo> providerInfo =
AuthV4.getPlayerInfo().providerInfoData;

foreach (AuthV4.ProviderType type in providerInfo.Keys) {
        switch (type) {

        case AuthV4.ProviderType.GOOGLE:
                enableGoogle = true;
                break;

        case AuthV4.ProviderType.APPLE:
                enableGameCenter = true;
                break;

        case AuthV4.ProviderType.FACEBOOK:
                enableFacebook = true;
                break;

        case AuthV4.ProviderType.HIVE:
                enableHive = true;
                break;

        case AuthV4.ProviderType.SIGNIN_APPLE:
                enableAppleSignIn = true;
                break;

        default:
                // nothing to do
                break;
        }
}

#if UNITY_ANDROID
buttonGoogle.GetComponent<Image>().sprite = enableGoogle ? spritePGSOn : spritePGSOff;

#elif UNITY_IOS
buttonGoogle.GetComponent<Image>().sprite = enableGoogle ? spritePGOn : spritePGOff;
```

```
buttonGameCenter.GetComponent<Image>().sprite = enableGameCenter ? spriteGameCenterOn :
spriteGameCenterOff;

buttonAppleSignIn.GetComponent<Image>().sprite = enableAppleSignIn ? spriteAppleSignInOn :
spriteAppleSignInOff;

#endif

buttonFacebook.GetComponent<Image>().sprite = enableFacebook ? spriteFBOn : spriteFBOff;
buttonHIVE.GetComponent<Image>().sprite = enableHive ? spriteHIVEOn : spriteHIVEOff;
```

```
// HIVEManager.cs: HIVE SDK Provider 연동API 호출

public void connect(AuthV4.ProviderType type, onHIVEManagerRedirectEvent listener)
{
        Debug.Log("HIVEManager - connect");

        onHIVEManagerConnect += listener;

        AuthV4.Helper.connect(type, onAuthV4HelperConnect);
}
```

```
// HIVEManager.cs: HIVE SDK IDP 연동 결과 콜백 핸들러

void onAuthV4HelperConnect(ResultAPI result, AuthV4.PlayerInfo playerInfo)
{
        Debug.Log ("onAuthV4HelperConnect : " + result.toString());

        bool success = false;
        bool moveToTitle = false;  // 유저 정보 변경으로 인하여 타이틀 화면으로 이동하여 게임 세션
재인증 여부
        string message = null;

        message = result.message;

        if (result.isSuccess() == true) {
                success = true;
        } else if (result.errorCode == ResultAPI.ErrorCode.PLAYER_CHANGE && playerInfo != null) {
                success = true;
                moveToTitle = true;
        } else {
                success = false;
        }

        onHIVEManagerConnect(success, message, moveToTitle);
        onHIVEManagerConnect -= new onHIVEManagerRedirectEvent(onHIVEManagerConnect);
}
```

```
// HIVEManager.cs

// HIVE SDK IDP 연동 해제 함수를 호출한다
public void disconnect(AuthV4.ProviderType type, onHIVEManagerEvent listener)
{
        Debug.Log("HIVEManager - disconnect");
        onHIVEManagerDisconnect += listener;

        AuthV4.Helper.disconnect(type, onAuthV4HelperDisconnect);
```

```
}
```

```
// HIVEManager.cs

// HIVE SDK IDP 연동 해제 결과 콜백 핸들러
void onAuthV4HelperDisconnect(ResultAPI result, AuthV4.PlayerInfo playerInfo)
{
  Debug.Log ("onAuthV4HelperDisconnect : " + result.toString());

  onHIVEManagerDisconnect(result.isSuccess(), result.message);
  onHIVEManagerDisconnect -= new onHIVEManagerEvent(onHIVEManagerDisconnect);
}
```

```
// HIVEManager.cs: HIVE SDK 인증 해제

public void signOut(onHIVEManagerEvent listener)
{
        Debug.Log("HIVEManager - signOut");

        onHIVEManagerSignOut += listener;
        AuthV4.Helper.signOut(onAuthV4HelperSignOut);
}
```

```
// HIVEManager.cs: HIVE SDK 인증 해제 결과 콜백 핸들러

void onAuthV4HelperSignOut(ResultAPI result, AuthV4.PlayerInfo playerInfo)
{
        Debug.Log ("onAuthV4HelperSignOut : " + result.toString());

        isSignIn = false; // signIn flag 변경
        Promotion.setEngagementReady(false); // engagement 설정을 변경. false

        onHIVEManagerSignOut(true, null);
        onHIVEManagerSignOut -= new onHIVEManagerEvent(onHIVEManagerSignOut);
}
```

```
// Settings.cs: 1:1 문의

void onHIVEManagerShowInquiry(bool success, string message)
{
        Debug.Log ("onHIVEManagerShowInquiry : " + message);
}

public void onInquiryClick()
{
        Debug.Log ("onInquiryClick");
        HIVEManager.shared.showInquiry(onHIVEManagerShowInquiry);
}
```

```
// Settings.cs: 약관

void onHIVEManagerShowTerms(bool success, string message)
{
        Debug.Log ("onHIVEManagerShowTerms : " + message);
}

public void onTermsClick()
{
        Debug.Log ("onTermsClick");
```

```
        HIVEManager.shared.showTerms(onHIVEManagerShowTerms);
}
```

d. 캠페인

캠페인은 유저 애퀴지션(User Acquisition, UA)이라는 기능으로, 게임 유저를 통해 신규 유저를 확보하는 방식입니다. 게임에서는 이 기능을 공유하기, 초대링크, QR 코드 방식으로 제공하고 있습니다.

캠페인 화면에서는 캠페인별 보상을 표시하고 있으며, 달성한 캠페인은 흑백으로 처리하여 상태를 구분합니다. 화면 하단에 위치한 세 버튼은 왼쪽부터 각각 공유하기, 초대링크 URL, QR 코드입니다.

[그림 11] 캠페인 화면

i. 구현 파일

1. CampaignManager.cs

ii. HIVE API

API	설명

Promotion.getAppInvitationData()	앱 초대 목적으로 데이터를 조회합니다
Promotion.showUAShare()	“소셜 미디어 공유”가 가능한 앱 목록을 노출합니다

iii. 예시 코드

굵은 글씨는 HIVE API를 사용하는 부분입니다.

```
// CampaignManager.cs: Campaign UI 내 항목을 재설정

void reloadData ()
{
	Debug.Log ("reloadData");

	ClearOldItems ();

	indicator.ShowIndicator();

	Promotion.getAppInvitationData(PromotionAppInvitationHandler);
}
```

```
// CampaignManager.cs: HIVE 로부터 응답받은 Promotion 데이터 응답 처리

@param result 응답 결과
@param appInvitationData Promotion data
*/
void PromotionAppInvitationHandler (ResultAPI result, AppInvitationData appInvitationData)
{
	Debug.Log ("PromotionAppInvitationHandler:\nresult - " + result + "\nappInvitationData - " +
appInvitationData);

	if (result.isSuccess () == false) {
		popup.show ("Notice", result.errorMessage, "OK", null);
	} else {
		_qrcode = appInvitationData.qrcode;
		_inviteCommonLink = appInvitationData.inviteCommonLink;
		_inviteMessage = appInvitationData.inviteMessage;

		if ((_inviteMssage == null) || (_inviteMessage == “null”)) {
		   _inviteMessage = “”;
		}

		foreach (AppInvitationCampaign campaign in appInvitationData.stageCampaignList)
{
			AddNewItem (campaign, true);
		}

		foreach (AppInvitationCampaign campaign in appInvitationData.eachCampaignList)
{
			AddNewItem (campaign, false);
		}
	}

	indicator.HideIndicator ();
}
```

```
// CampaignManager.cs: 링크 공유

/*
Link Share 버튼 동작
 */
public void onShareButton ()
{
	Debug.Log ("onShareButton - " + _inviteCommonLink);

	if (_inviteCommonLink == null) {
		popup.show ("Notice", "Error - Invite URL NOT found.", new string [] { "OK" }, true,
null);
	}
	else {
		popup.show ("Notice", _inviteCommonLink, new string[] { "Cancel", "Share" },
ShareButtonHandler);
	}
}

/*
Link Share 버튼 동작 handler
@param buttonIndex button index.
*/
void ShareButtonHandler (int buttonIndex)
{
	Debug.Log ("ShareButtonHandler - " + buttonIndex);

	if (buttonIndex == 0) {
		// no operation.
	} else if (buttonIndex == 1) {
		Promotion.showUAShare(_inviteMessage, _inviteCommonLink,
onHiveShareHandler);
	}
}

/*
Share HIVE handler
@param result Share HIVE 동작 결과
*/
void onHiveShareHandler (ResultAPI result)
{
	Debug.Log ("onHiveShareHandler - " + result);

	if (result.isSuccess()) {
		popup.show("Share Successful!", "You've successfully shared the invitation link.",
new string[] {"OK"}, true, null);
	} else {
		popup.show("Share Failed!", "Failed to share the inviatation link. Please try again.",
new string[] {"OK"}, true, null);
	}
}
```

```
// CampaignManager.cs: QR Code

/*
QR Code 버튼 동작
*/
public void onQRCodeButton ()
{
	Debug.Log ("onQRCodeButton");

	if (_qrcode == null) {
		popup.show ("Notice", "QR code image NOT found.", new string [] { "OK" }, null);
	}
	else {
		popup.show ("Notice", _qrcode, new string [] { "OK" }, null);
	}
```

```
}
```

e. 이벤트(프로모션)

HIVE에서는 유저에게 게임의 새로운 소식이나 이벤트를 효과적으로 제공하기 위해 프로모션 기능을 지원합니다. 샘플 게임에서 사용한 프로모션은 **전면 배너**와 **새소식 페이지 띄우기**로, 전면 배너는 게임에 로그인한 후 표시하고 새소식 페이지는 메인 화면에서 이벤트 버튼을 클릭했을 때 표시합니다.

딥링크 기능이 포함된 유저 인게이지먼트(User Engagement, UE) 모듈 또한 HIVE에서 지원합니다. 딥링크는 프로모션 목적에 따라 클릭만으로 유저를 게임 내 특정 위치로 이동하거나 이벤트 페이지로 초대할 수 있고, 바로 보상을 획득할 수도 있는 기능입니다. 본 게임에는 **interwork://hive/iappurchase?marketpid={MARKET_PID}**처럼 배너에서 상품을 구매할 수 있는 interwork URL Scheme을 적용했습니다. 게임에서 사용할 때는 **HIVE Console > 프로모션 > 캠페인 설정 > 스팟 배너** 메뉴에서 배너 이미지와 위 링크를 삽입한 후 상품을 노출합니다. 스팟 배너를 클릭하면 URL Scheme 내 MARKET_PID 값에 해당하는 상품을 구매할 수 있으며, 구매한 상품은 우편함에서 확인 가능합니다.

i. 구현 파일

1. GameConnect.cs

ii. HIVE API

API	설명
Promotion.setEngagementReady()	Engagement 이벤트 처리 가능 여부를 설정합니다
Promotion.showPromotion()	새로운 이벤트나 새로운 게임 소식 등 배너 화면을 표시합니다
Promotion.showCustomContents()	외부 컨텐츠를 사용하기 위해 커스텀 뷰를 노출합니다
IAPV4.transactionFinish()	상품 지급 완료 후 결제 트랜잭션 완료 처리를 요청합니다

iii. 예시 코드

```
// HomeManager.cs

//====================================
//          GameConnect 콜백 핸들러
//====================================
void onGameConnectSignIn(bool success, string message)
{
        IsGameSignIn = success;

        if (success == true) {
                HIVEManager.shared.didLogin();
        } else {
                popup.show("Game Sign-In Result", message, "OK", null);
        }
}
```

```
// HIVEManager.cs

// HIVE SDK 인증 및 GameServer 인증이 완료 후 호출
public void didLogin()
{
	Debug.Log("HIVEManager - didLogin");

	shopSetup(null);  // 미리 마켓 상점 연결하여 상품 정보를 얻어온다

	promotionShowBanner(); // HIVE SDK 대배너 프로모션 UI 노출

	Promotion.setEngagementReady(true); // engagement 설정을 변경. true
}
```

```
// HomeManager.cs

public void onEventClick()
{
	Debug.Log ("onEventClick");

	HIVEManager.shared.promotionShowNews();
}
```

```
// HIVEManager.cs

// HIVE SDK Promotion 배너 보기 함수를 호출한다
public void promotionShowNews()
{
	Debug.Log("HIVEManager - promotionShowNews");

	// forced 값이 true
	Promotion.showPromotion(PromotionType.NEWS, true, onPromotionView);
}
```

```
// HIVEManager.cs

// HIVE SDK Promotion Spot 배너 보기 함수를 호출한다
public void promotionShowInAppPromotion()
{
	Debug.Log("HIVEManager - promotionShowInAppPromotion");
	// Spot 배너를 이용하여 Game 종료 시 InApp 상품 노출
	string code = "";

	#if UNITY_ANDROID
	code ="10000";
	#elif UNITY_IOS
	code = "10001";
	#else
	return;
	#endif

	Promotion.showCustomContents(PromotionCustomType.SPOT, code, null);
}
```

```
// HIVEManager.cs

// HIVE SDK Promotion Engagement Listener
```

```
void onPromotionEngagement(ResultAPI result, EngagementEventType engagementEventType,
EngagementEventState engagementEventState, JSONObject param)
{
        Debug.Log ("onPromotionEngagement : " + result.toString() + "\nparam :" + param);

        if (engagementEventType == EngagementEventType.IAP_PURCHASE) {
                if (engagementEventState == EngagementEventState.END) {
                        string bypassInfo = "";
                        param.GetField("iapV4Receipt").GetField(ref bypassInfo, "bypassInfo");

                        if (bypassInfo != null) {
                                // 게임 서버로 영수증을 전달한다
                                GameConnect.shared.VerifyReceipt(bypassInfo,
onGameConnectVerifyReceipt);
                        }
                                Promotion.setEngagementReady(true);
                }
        }
}
```

```
// HIVEManager.cs

void onGameConnectVerifyReceipt(bool success, string message, string pid)
{
        if (success == true) {
                HIVEManager.shared.shopTransactionFinish(pid,
onInternalHIVEManagerTransactionFinish);
        } else {
                // TODO: error handle.
        }
}
```

```
// HIVEManager.cs

// HIVE SDK IAP V4 지급이 완료된 상품에 대한 transaction finish 처리 함수를 호출한다
public void shopTransactionFinish(string marketPid, onHIVEManagerEvent listener)
{
        Debug.Log("HIVEManager - shopTransactionFinish");
                if (_isMarkConnected == true && _isGetProductInfo == true) {
                // transaction-finish 시도
                IAPV4.transactionFinish(marketPid, onIAPV4TransactionFinish);
        } else {
                // 상품 구매 전 마켓 연동 및 상품 목록을 얻어와야 한다
                listener(false, "Need setup.");
        }
}
```

f. 빌링

구매는 5단계로 진행됩니다. 아래 순서로 구매를 진행하다 특정 단계에서 트랜잭션이 마무리되지 않을 경우에는 **복구**를 시도합니다.

1. 마켓 연결하기
2. 상품 정보 가져오기
3. 구매하기
4. 검증(중복체크) 후 상품 지급하기
5. 구매 종료하기

HIVE SDK IAP 모듈은 IAP와 IAP V4로 나뉘는데, 이 게임은 IAP V4를 활용해 구현했습니다.

마켓 연결하기는 로그인 직후 한 번 시도하며, 실패한 경우에는 마켓을 열 때마다 다시 시도하도록 구현되어 있습니다. 마켓이 열리지 않으면 이후 진행이 불가하므로 항상 선행되어야 하는 단계입니다.

상품 정보 가져오기에서는 연결된 마켓에서 받아온 정보를 HIVE IAP 서버를 거쳐 부가세가 계산된 값을 받습니다. 샘플 게임에서는 IAP 서버에서 제공하는 고정 값으로 구현하였으나 실제 게임 서버까지 거치는 UI를 구성하면 서버에서 동적인 상점 UI를 제공할 수 있습니다. 상품 정보를 가져오다 실패했을 경우에는 다시 시도하도록 샘플 코드에 구현했습니다.

구매하기 단계에서는 상품 정보 가져오기로 받은 정보 중 Market PID(마켓에 등록된 상품 고유 ID)를 이용해 구매를 요청합니다.

검증 후 상품 지급하기는 상품을 지급하기 전에 영수증 검증과 중복 체크를 하는 단계로 대부분 게임 서버에서 처리합니다.
처리 순서는 첫 번째, 플랫폼 구분 없이 구매하기 결과로 받은 영수증 값을 검증합니다. 영수증 객체의 String bypassInfo 값을 게임 서버로 보내면 게임 서버는 HIVE IAP 서버에 그대로 전달한 후 검증 결과를 돌려받습니다. HIVE IAP 서버에서 보내준 결과 중 hiveiap_market_pid 값으로 상품을 지급합니다. 게임 클라이언트에서 직접 검증하면 보안이 취약하기 때문에 서버에서 검증하고(Server to Server) 게임 서버에서 상품을 지급하도록 구현합니다.
두 번째, 영수증 전달이나 검증 과정에서 구매 트랜잭션이 완전히 종료되지 않아 동일한 영수증을 다시 받을 수 있습니다. 이 때문에 영수증이 한 번만 지급되도록 중복으로 체크해야 합니다. HIVE IAP 서버에서 검증하면 hiveiap_transaction_id 값을 내려주는데, 이 값은 검증 완료한 영수증별로 생성되는 Transaction ID입니다. 이 값을 게임 서버에 저장해서 게임이 영수증 중복 체크를 수행하도록 합니다.

구매 종료하기는 검증과 중복 체크 후 상품 지급까지 완료되면 진행하는 절차입니다. 구매 트랜잭션이 종료되기 전까지는 복구 요청을 할 때마다 같은 영수증을 전달하기 때문에 해당 ID 상품의 구매가 진행되지 않습니다.
같은 영수증을 받을 때는 영수증 검증과 중복 체크한 후 상품을 지급하지 않고 트랜잭션을 종료합니다. HIVE IAP 서버 검증 결과가 1000503일 경우에도 트랜잭션을 종료합니다.

* 1000503: 검증 실패. 해킹이나 영수증 변조 같이 명백한 검증 에러로, 상품을 지급하지 않고 마켓 트랜잭션 종료를 권장합니다.

구매 트랜잭션 복구는 유저가 구매 과정을 거쳐 실제 상품을 지급받기까지의 과정 중 네트워크 단절 등의 사유로 지급이 중단되었을 때 활용하는 기능입니다. 이 경우 구매 트랜잭션 복구를 요청한 후 영수증을 다시 받아 검증 단계부터 진행할 수 있습니다. 샘플 게임에서는 유저가

직접 호출하도록 설정되었지만 상품 지급이 가능할 때나 상점에 진입 시와 같이 호출 시점을 지정하는 방식이 가능합니다. 단, 복구 콜백을 받기 전에 구매가 진행되거나 상점에 진입할 경우 유저 모르게 자동으로 복구하니 실패 팝업을 띄우는 방법으로 구현하지 않도록 주의해야 합니다.

[그림 12] 빌링 화면

i. 구현 파일

1. Shop.cs

ii. HIVE API

API	설명
IAPV4.marketConnect()	결제 API 초기화를 요청합니다
IAPV4.getProductInfo() IAPV4.getSubscriptionProductInfo()	상품 구성을 위한 정보를 조회합니다
IAPV4.purchase() IAPV4.purchaseSubscriptionUpdate()	마켓(Apple App Store, Google Play Store)에 상품 구매를 요청합니다
IAPV4.restore() IAPV4.restoreSubscription()	미지급된 상품 지급을 요청합니다
IAPV4.transactionFinish()	지급 완료된 상품의 결제 트랜잭션을 완료합니다

iii. 예시 코드

```
// HIVEManager.cs

// HIVE SDK IAP V4 상점 연동 및 상품 정보를 얻어온다
// marketConnect()가 성공한 이후에 getProductInfo()를 호출
public void shopSetup(onHIVEManagerEvent listener)
{
	Debug.Log("HIVEManager - show setup");

	if (_isMarkConnected == false) {
		// 마켓 연동 시도
		Debug.Log("HIVEManager - marketConnect");

		onHIVEManagerShopSetup += listener;
		IAPV4.marketConnect(onIAPV4MarketConnect);

	} else if (_isGetProductInfo == false) {
		// 상품 정보 얻어오기 시도
		Debug.Log("HIVEManager - productInfo");

		onHIVEManagerShopSetup += listener;
		IAPV4.getProductInfo(onIAPV4ProductInfo);

	} else {
		// 이미 마켓 연동 및 상품 목록 얻어오기 완료
		Debug.Log("HIVEManager - Already setup.");
		listener(true, null);
	}
}
```

```
// HIVE SDK IAP V4 마켓연동(marketConnect) 결과 콜백 핸들러
void onIAPV4MarketConnect(ResultAPI result, List<IAPV4.IAPV4Type> iapV4TypList)
{
	Debug.Log ("onIAPV4MarketConnect : " + result.toString());

	if (result.isSuccess() == true) {
		_isMarkConnected = true;
		IAPV4.getProductInfo(onIAPV4ProductInfo);

	} else {
		_isMarkConnected = false;
	}
}
```

```
// HIVE SDK IAP V4 상품 목록 요청 결과 콜백 핸들러
void onIAPV4ProductInfo(ResultAPI result, List<IAPV4.IAPV4Product> iapV4ProductList, int balance)
{
	Debug.Log ("onIAPV4ProductInfo : " + result.toString());

	bool success = false;
	string message = null;

	if (result.isSuccess() == true) {
		_productList = iapV4ProductList;
		_isGetProductInfo = (_productList != null);

		string marketPid1000G = "";
		string marketPid4000G = "";
		string marketPid10000G = "";
```

```
                #if UNITY_ANDROID
                marketPid1000G =
"com.com2us.stepbystep.normal.freefull.google.global.android.common.1000g";
                marketPid4000G =
"com.com2us.stepbystep.normal.freefull.google.global.android.common.4000g";
                marketPid10000G =
"com.com2us.stepbystep.normal.freefull.google.global.android.common.10000g";
                #elif UNITY_IOS
                marketPid1000G = "com.com2us.stepbystep.1000g";
                marketPid4000G = "com.com2us.stepbystep.4000g";
                marketPid10000G = "com.com2us.stepbystep.10000g";
                #endif

                int i = 0;
                foreach (IAPV4.IAPV4Product product in _productList) {

                        if (string.Equals (product.marketPid, marketPid1000G))
                                item1Index = i++;
                        else if (string.Equals (product.marketPid, marketPid4000G))
                                item2Index = i++;
                        else if (string.Equals (product.marketPid, marketPid10000G))
                                item3Index = i++;
                        else
                                i++;
                }
                success = true;
                message = result.message;

        } else {
                _isGetProductInfo = false;
                success = false;
                message = result.message;
        }

        onHIVEManagerShopSetup(success, message);
        onHIVEManagerShopSetup -= new onHIVEManagerEvent(onHIVEManagerShopSetup);
}
```

```
// HIVEManager.cs

// HIVE SDK IAP V4 상품 구매 함수를 호출한다
public void shopPurchase(string marketPid, onHIVEManagerPurchaseEvent listener)
{
        if (_isMarkConnected == true && _isGetProductInfo == true) {
                // 상품 구매를 시도한다
                onHIVEManagerPurchase += listener;

                IAPV4.purchase(marketPid, null, onIAPV4Purchase);

        } else {
                // 상품 구매 전 마켓 연동 및 상품 목록을 얻어와야 한다
                listener(false, "Need setup.", null);
        }
}
```

```
// HIVE SDK IAP V4 구매 결과 콜백 핸들러
void onIAPV4Purchase(ResultAPI result, IAPV4.IAPV4Receipt iapV4Receipt)
{
        Debug.Log ("onIAPV4Purchase : " + result.toString());

        if (result.isSuccess() == true) {
                onHIVEManagerPurchase(true, result.message, iapV4Receipt);
```

```
        } else {
                onHIVEManagerPurchase(false, result.message, null);
        }
        onHIVEManagerPurchase -= new
onHIVEManagerPurchaseEvent(onHIVEManagerPurchase);
}
```

```
void onHIVEManagerPurchase(bool success, string message, IAPV4.IAPV4Receipt iapV4Receipt)
{
        if (success) {
                GameConnect.shared.VerifyReceipt(iapV4Receipt, onGameConnectVerifyReceipt);
        } else {
                popup.show("Purchase", "Purchase Failed.\n" + message, "OK", null);
        }
}
```

```
// Game server로 IAPV4.IAPV4Receipt 데이터를 전달하여 검증 요청한다

public void VerifyReceipt(IAPV4.IAPV4Receipt iapV4Receipt, onGameConnectReceiptEvent listener)
{
    Debug.Log("GameConnect - VerifyReceipt");

    onGameConnectReceiptVerify += listener;

    RequestBillingVerify request = new RequestBillingVerify ();

    // require
    request.bypassInfo = iapV4Receipt.bypassInfo;

    // optional
    request.PlayerID = AuthV4.getPlayerInfo ().playerId;
    request.AppID = Configuration.getAppId ();
    int marketId = 0;
    #if UNITY_ANDROID
    marketId = 2;
    #elif UNITY_IOS
    marketId = 1;
    #endif
    request.MarketID = marketId;
    request.HIVEIAPReceipt = iapV4Receipt.hiveiapReceipt;
    request.Price = (float)iapV4Receipt.product.price;
    request.Currency = iapV4Receipt.product.currency;

    //request.AppVersion = Configuration.;
    request.DID = long.Parse (AuthV4.getPlayerInfo ().did);
    request.HIVECountry = Configuration.getHiveCountry();
    //request.Country = Configuration
    //request.Language;
    //request.GameLanguage = Configuration.get
    //request.UID;
    //request.DeviceModel =
    //request.OSVersion = Configuration.getv;
    //request.OSAPILevel;
    request.SDKVersion = Configuration.getHiveSDKVersion();
    //request.ReceiptLevel;

    request.Request (onGameReceiptVerify);
}
```

```
// HIVEManager.cs
```

```
void onGameConnectVerifyReceipt(bool success, string message, string pid)
{
	if (success == true) {
		// 계속 restore가 발생하지 않도록 게임서버에 전달이 된 상품은 HIVE SDK에
transactionFinish를 해 주어야 한다
		HIVEManager.shared.shopTransactionFinish(pid,
onHIVEManagerTransactionFinish);
	} else {
		popup.show("Shop", message, "OK", null);
	}

	UpdatePlayerHealthUpButton();
	UpdatePlayerSpeedUpButton();
}
```

```
// HIVE SDK IAP V4 미지급된 상품에 대한 지급 요청처리 함수를 호출한다
public void shopRestore(onHIVEManagerRestoreEvent listener)
{
	Debug.Log("HIVEManager - shopRestore");

	if (_isMarkConnected == true && _isGetProductInfo == true) {
		// 상품 구매 복원 시도
		onHIVEManagerRestore += listener;


		IAPV4.restore(onIAPV4Restore);

	} else {
		// 상품 구매 전 마켓 연동 및 상품 목록을 얻어와야 한다
		listener(false, "Need setup.", null);
	}
}
```

```
// HIVE SDK IAP V4 구매 복원 결과 콜백 핸들러
void onIAPV4Restore(ResultAPI result, List<IAPV4.IAPV4Receipt> iapV4ReceiptList)
{
	Debug.Log ("onIAPV4Restore : " + result.toString());

	bool success = false;
	string message = null;
	List<IAPV4.IAPV4Receipt> receiptList = null;

	// 적절한 시점에 자동으로 해도 상관은 없으나 마켓 연결 여부와, 상품 지급이 가능한 시점,
	// 결제 도중에 중복으로 호출되지 않도록 주의한다
	// NOT_OWNED 는 요청에는 성공했으나 복구할 상품이 없다는 뜻이다. 일반적으로 대부분 이
경우에 해당
	if (result.errorCode == ResultAPI.ErrorCode.NOT_OWNED) {
		success = true;
		message = "Restore Success.\nbut Not owned item.";
	} else if (result.isSuccess() == true) {
		success = true;
		message = result.message;
		receiptList = iapV4ReceiptList;
	} else {
		success = false;
		message = "Restore failed.\n" + result.message;
	}

	onHIVEManagerRestore(success, message, receiptList);
	onHIVEManagerRestore -= new onHIVEManagerRestoreEvent(onHIVEManagerRestore);
}
```

g. 노티피케이션

HIVE SDK는 유저에게 다양한 방법으로 알림을 보낼 수 있는 푸시 노티피케이션 서비스를 제공합니다. 알림 종류는 모든 알림, 게임 알림, 공지 알림, 공지 야간 알림이 있습니다.

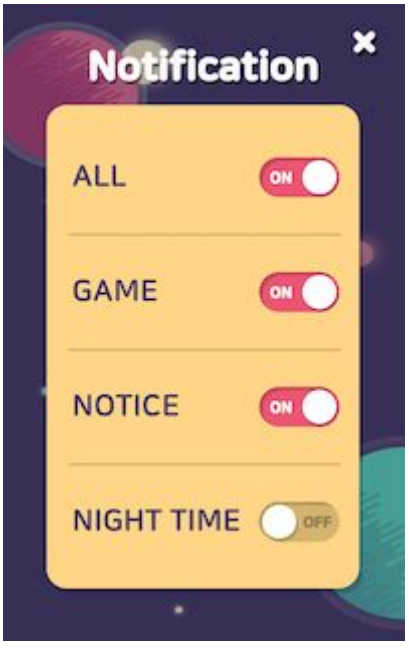
모든 알림은 Android 플랫폼에서만 사용 가능하기 때문에 UI 구성 시 주의가 필요합니다. 모든 알림을 On으로 활성화하면 게임 알림과 공지 알림도 On으로 설정됩니다. 모든 알림을 Off로 비활성화하면 게임 알림과 공지 알림 역시 Off가 됩니다. 공지 알림이 Off 되면 야간 알림 또한 Off 상태가 됩니다.

공지 알림과 야간 알림의 연계 정책은 아래와 같습니다.

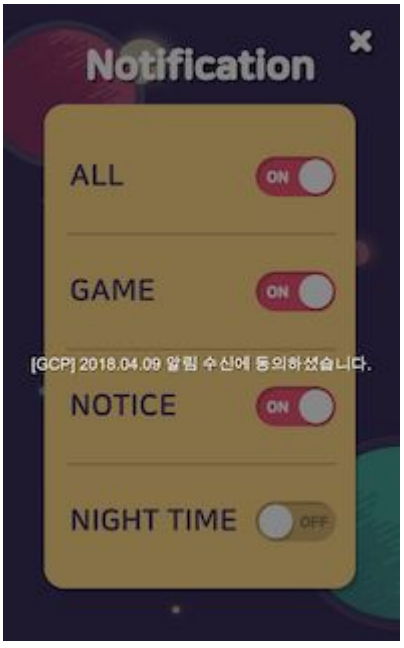
1. 공지 알림을 비활성화하면 야간 알림도 자동으로 비활성화해야 합니다.
2. 공지 알림이 비활성 상태일 때 야간 알림을 활성화할 수 없어야 합니다.
3. 공지 알림을 활성화해도 야간 알림을 자동으로 활성화하지 않아야 합니다.

알림 설정 변경 시 유저에게 변경 사항을 알려주기 위해 토스트 팝업을 사용합니다. 설정이 변경된 경우 토스트 팝업을 2초간 표시하며, 이는 모든 알림, 공지 알림, 야간 알림 설정이 변경되었을 때 표시해야 합니다. 게임에 구현된 문구 예시는 아래와 같습니다.

a. 모든 알림 동의 활성화: [게임빌] 2016.05.01 알림 수신에 동의하셨습니다.
b. 공지 알림 동의 비활성화: [컴투스] 2016.05.01 알림 수신에 거부하셨습니다.
c. 야간 알림 동의 활성화: [게임빌] 2016.05.01 야간 알림 수신에 동의하셨습니다.



[그림 13] 노티피케이션 화면



[그림 14] 노티피케이션 알림 문구

i. 구현 파일

1. Notification.csm
2. Toast.cs

ii. HIVE API

API	설명
Push.setRemotePush()	유저가 푸시를 수신하는 상태를 설정합니다
Push.getRemotePush()	유저가 푸시를 수신하는 상태를 조회합니다

iii. 예시 코드

```
// 버튼 정책 로직

private bool _isAll = true;
private bool _isGame = true;
private bool _isNotice = true;
private bool _isNightTime = false;


public bool isAll {   //모든 알림
        set {
                _isAll = value;

                _isGame = value;   // 게임 알림의 설정도 같이 변경합니다
                isNotice = value;    // 공지 알림의 설정도 같이 변경합니다. 공지 알림은 야간
알림에도 영향
        }
        get {
                return _isAll;
        }
}

public bool isGame {   // 게임 알림
        set {
                _isGame = value;
                if ( !(_isGame ^ _isNotice) )
                        _isAll = value;       // 게임, 공지 알림이 모두 On 이나 Off 일 경우
        }                                     // 모든 알림도 같이 변경합니다
        get {
                return _isGame;
        }
}

public bool isNotice {   // 공지 알림
        set {
                _isNotice = value;

                if (!_isNotice)
                        _isNightTime = false;

                if ( !(_isGame ^ _isNotice) )
                        _isAll = value;
        }
        get {
                return _isNotice;
        }
}
```

```
public bool isNightTime {   // 야간 알림
        set {
                _isNightTime = value;
                if (_isNightTime)
                        isNotice = true;   // 야간 알림이 On 되면 공지 알림도 On 합니다
        }                                  // 혹은, 공지 알림이 Off일 경우 야간 알림이 설정 되지
않도록 합니다
        get {
                return _isNightTime;
        }
}
```

iv. 푸시 버튼 클릭 시

```
public void onALLClick() {
        isAll = !isAll;

        if (isAll) {
                toast.show ("[GCP] " + getToday() + " 알림 수신에 동의하셨습니다.");
        } else {
                toast.show ("[GCP] " + getToday() + " 알림 수신에 거부하셨습니다.");
        }

        HIVEManager.shared.setRemotePush(new RemotePush(isNotice, isNightTime),
onHIVEManagerRemotePush);
}
```

```
string getToday()
{
        return System.DateTime.Now.ToString ("yyyy.MM.dd");
}
```

v. HIVE API

```
// HIVE 서버에 저장된 유저의 리모트 푸시 수신 여부 정보를 조회
public void getRemotePush(onHIVEManagerRemotePushEvent listener)
{
        onHIVEManagerGetRemotePush += listener;
        Push.getRemotePush (onGetRemotePushCB);
}

// 변경된 수신 정보를 HIVE 서버에 전달
public void setRemotePush(RemotePush remotePush, onHIVEManagerRemotePushEvent listener)
{
        onHIVEManagerSetRemotePush += listener;
        Push.setRemotePush (remotePush, onSetRemotePushCB);
}
```

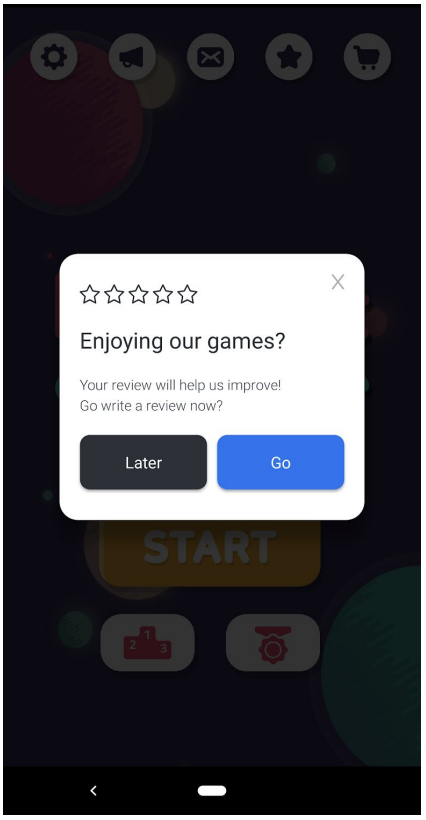
```
void onHIVEManagerRemotePush(bool isSuccess, string message, RemotePush remotePush)
{
        if (isSuccess) {
                isNotice = remotePush.isAgreeNotice;
                isNightTime = remotePush.isAgreeNight;
        }

        setButton ();
}

void setButton()
{
        buttonALL.GetComponent<Image>().sprite = isAll ? switchOn : switchOff;
        buttonGAME.GetComponent<Image>().sprite = isGame ? switchOn : switchOff;
        buttonNOTICE.GetComponent<Image>().sprite = isNotice ? switchOn : switchOff;
        buttonNIGHTTIME.GetComponent<Image>().sprite = isNightTime ? switchOn : switchOff;
}
```

h. 리뷰 팝업

i. 리뷰 팝업은 유저가 마켓(예. Google Play)에 게임에 대한 별점과 리뷰를 남기도록 유도하는 팝업입니다. HIVE의 리뷰 팝업 문구는 언어별로 고정되어 있으나, 여러분이 리뷰 유도 팝업을 직접 구현하면, 리뷰 팝업 문구와 팝업의 UI를 원하는 대로 구성할 수 있습니다.

ii. 구현 파일

1. GameManager.cs
2. HIVEManager.cs



[그림 15] 리뷰 팝업

iii. HIVE API

API	설명
Promotion.showNativeReview()	리뷰 팝업을 노출합니다

iv. 예시 코드

```
/*
Exit Button event.
*/
public void onGameExitButton ()
```

```
{
        Debug.Log ("on Exit button");
        SceneManager.LoadScene ("Home", LoadSceneMode.Single);
        HIVEManager.shared.showReview ();
}
```

```
// HIVEManager - 리뷰팝업
public void showReview()
{
        Debug.Log("HIVEManager - showReview");

        // onHIVEManagerShowReview += listener;
        #if !UNITY_EDITOR
        Promotion.showNativeReview ();
        #else
        Debug.Log("showReview");
        #endif // !UNITY_EDITOR
}
```

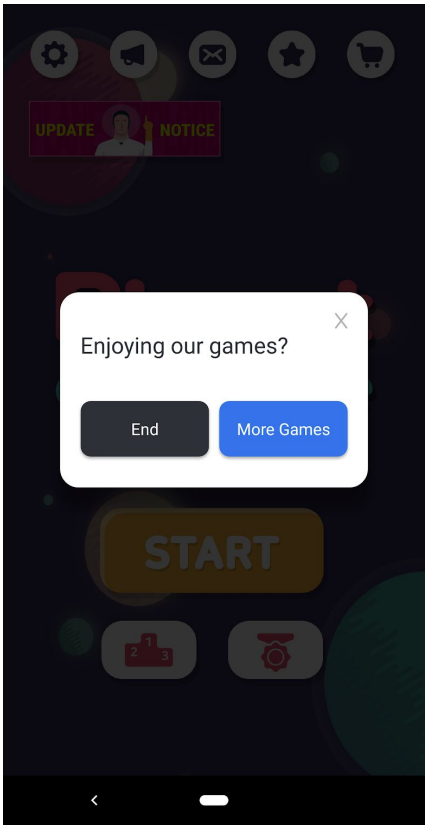
i. 종료 팝업

i. Android 플랫폼에서는 이용 중인 게임을 종료하려는 유저에게 새로운 HIVE 게임을 다운로드하도록 유도하기 위해 **더 많은 게임** 버튼을 노출할 수 있습니다.

ii. Android는 iOS와는 다르게 **되돌아가기(Back)** 키로 앱을 종료할 수 있어야 하며, 이 때 **종료 확인 팝업**을 띄울 수 있도록 구현합니다.

iii. 구현 파일

1. HomeManager.cs
2. HIVEManager.cs



[그림 16] 종료 팝업

iv. HIVE API

API	설명
Promotion.showExit()	종료 확인 팝업을 노출합니다

v. 예시 코드

```
// Update is called once per frame
void Update () {

#if UNITY_ANDROID
	if(Input.GetKeyDown(KeyCode.Escape))
	{
		HIVEManager.shared.showExit (onHIVEManagerShowExit);
	}
#endif
}
```

```
public void showExit(onHIVEManagerEvent listener)
{
	onHIVEManagerShowExit += listener;

	Promotion.showExit (onPromotionShowExitCB);
}
```

```
// HIVEManager - 종료팝업 UI 노출 결과 콜백 핸들러
void onPromotionShowExitCB(ResultAPI result, PromotionEventType promotionEventType)
{
	if (result.isSuccess () && promotionEventType == PromotionEventType.EXIT) {
		onHIVEManagerShowExit (true, result.message);
	} else {
		onHIVEManagerShowExit (false, result.message);
	}
	onHIVEManagerShowExit -= new onHIVEManagerEvent (onHIVEManagerShowExit);
}
```

```
public void onHIVEManagerShowExit(bool isSuccess, string message)
{
	if (isSuccess == true) {
		Application.Quit();
	}
}
```

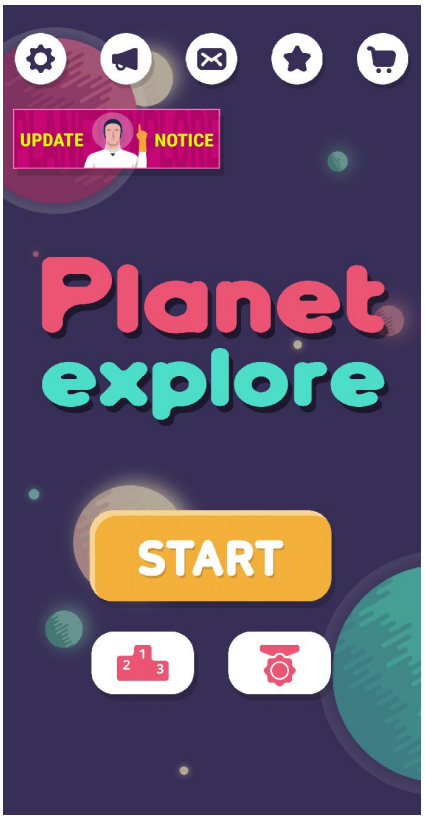
j. 롤링 배너

i. 롤링 배너란 HIVE SDK v4.13.0을 이용하는 게임에서 배너를 원하는 규격으로 자유롭게 노출하는 기능입니다.

ii. API를 통해 배너 데이터를 가져오면, 게임에서는 제공받은 데이터를 활용하여 특정 위치에 배너를 표시할 수 있습니다.

iii. 구현 파일

1. BannerManager.cs
2. HIVEManager.cs



[그림 17] 롤링 배너 노출

iv. HIVE API

API	설명
Promotion.getBannerInfo()	배너 데이터를 가져옵니다.
Promotion.showCustomContents()	배너 클릭시 Promotion 페이지를 노출합니다.

v. 예시 코드

```
void Start()
{
	HIVEManager.shared.promotionGetBannerInfo(onHIVEManagerGetBannerInfo);
}
```

```
public void promotionGetBannerInfo(onHIVEManagerPromotionBannerInfoEvent listener) {
	Debug.Log("HIVEManager - promotion get banner info");

	onHIVEManagerGetBannerInfo = listener;

	Promotion.getBannerInfo(PromotionCampaignType.NOTICE, PromotionBannerType.SMALL,
onGetBannerInfoCB);
}
```

```
public bool showBanner(GameObject bannerObject) {
	Image imageComponent = bannerObject.GetComponent<Image> ();
	Button buttonComponent = bannerObject.GetComponent<Button> ();

	imageComponent.sprite = sprite;
	buttonComponent.onClick.RemoveAllListeners();
	buttonComponent.onClick.AddListener( () => {
		Promotion.showCustomContents(PromotionCustomType.DIRECT, pid, (ResultAPI
ResultAPI, PromotionEventType promotionEventType) {
			// result
		});
	});
}
```

```
void onHIVEManagerGetBannerInfo(bool isSuccess, List<PromotionBannerInfo> bannerList) {
	Debug.Log("onHIVEManagerGetBannerInfo");

	foreach(PromotionBannerInfo bannerInfo in bannerList) {
		BannerData bannerData = new BannerData(bannerInfo);
		// string pid = bannerInfo.pid.ToString();
		// string imageUrl = bannerInfo.imageUrl;
		bannerDatas.Add(bannerData);
	}

}
```

```
void onGetBannerInfoCB(ResultAPI result, List<PromotionBannerInfo> bannerInfoList)
{
	Debug.Log("onGetBannerInfoCB : " + result.toString());

	if (onHIVEManagerGetBannerInfo != null) {
		if (result.isSuccess() ) {
			onHIVEManagerGetBannerInfo(true, bannerInfoList);
		} else {
			onHIVEManagerGetBannerInfo(false, null);
		}
	}

	onHIVEManagerGetBannerInfo = null;
}
```

4. 참고자료

a. Unity 다운로드: https://unity3d.com/kr/get-unity/download/archive
b. Blender 다운로드: https://www.blender.org/download/
c. HIVE 개발자 사이트: https://developers.withhive.com
d. HIVE Unity 설정: https://developers.withhive.com/dev4/porting-building-hive/unity/
e. 초기화, 로그인 설정: https://developers.withhive.com/dev4/authv4/#v4-helper
f. 캠페인: https://developers.withhive.com/dev4/promotion/useracquisition/
g. 이벤트: https://developers.withhive.com/dev4/promotion/userengagement/
h. 빌링: https://developers.withhive.com/dev4/billing/iapv4/
i. 노티피케이션: https://developers.withhive.com/dev4/notification/
j. 롤링 배너: http://developers.withhive.com/dev4/promotion/advanced/#rolling-banner