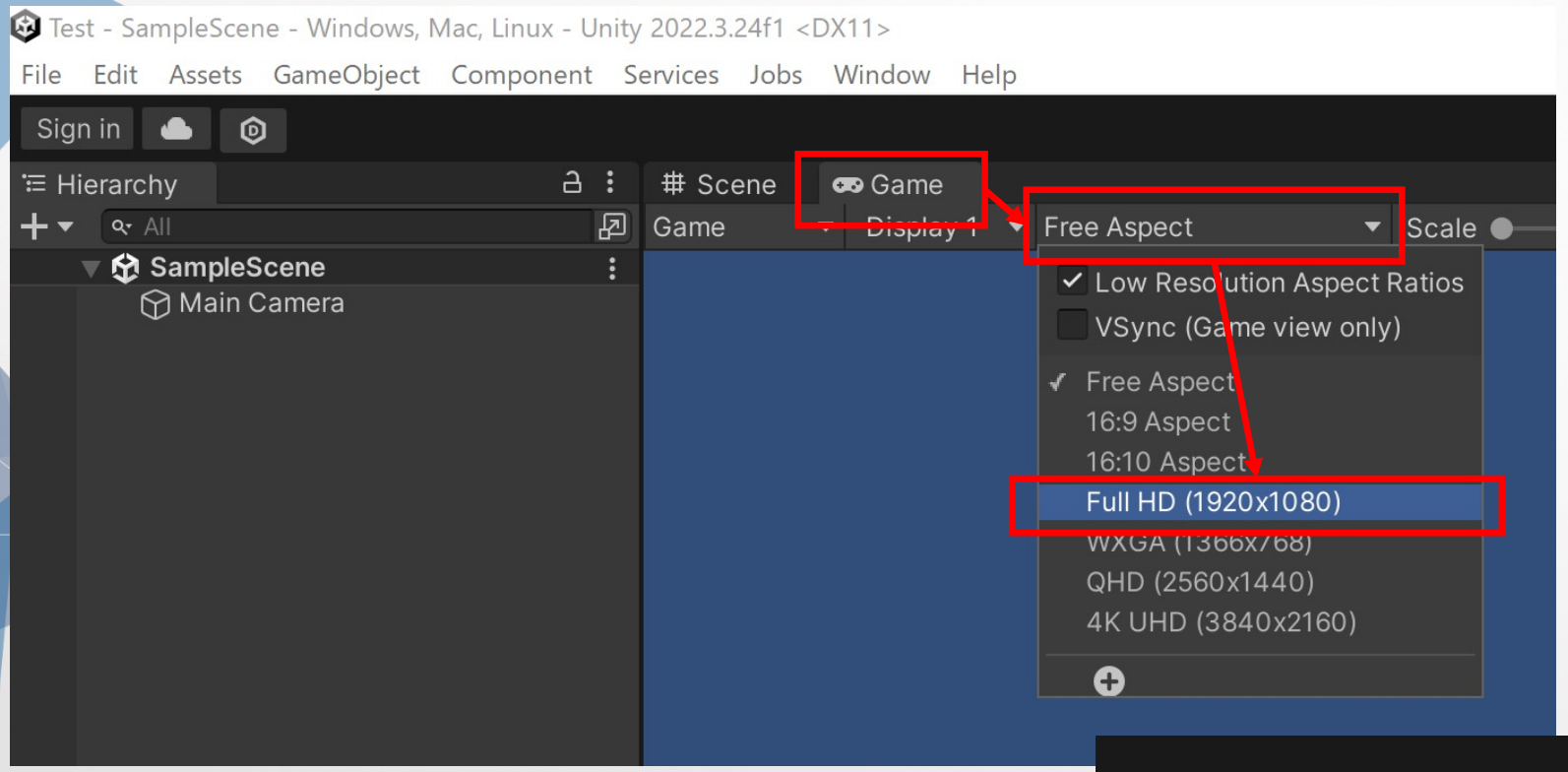
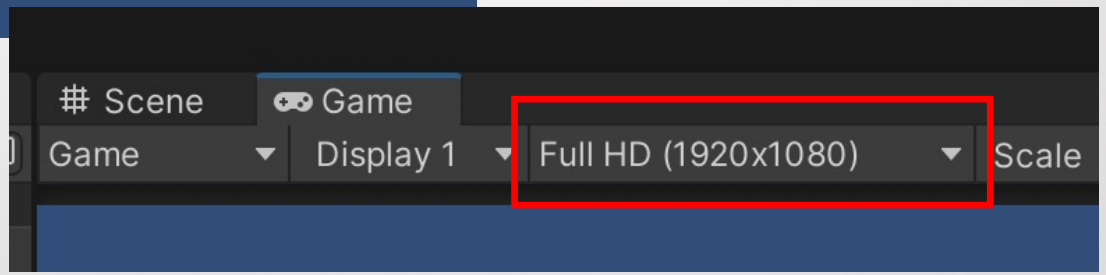
外部サービスとの連携

目次

UnityPackageを使用しての準備

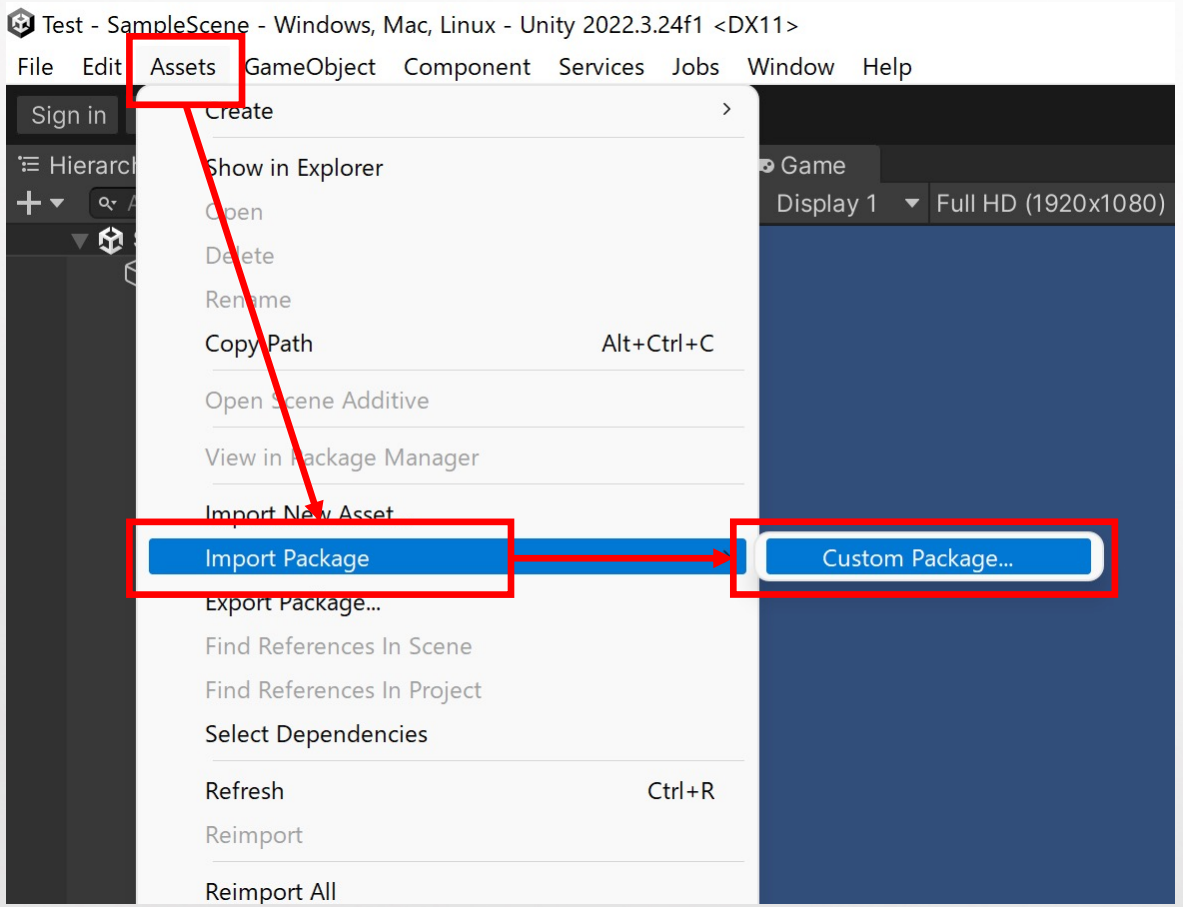
プロジェクト開いた人は、以下の操作をお願いします。(画面の解像度の設定)





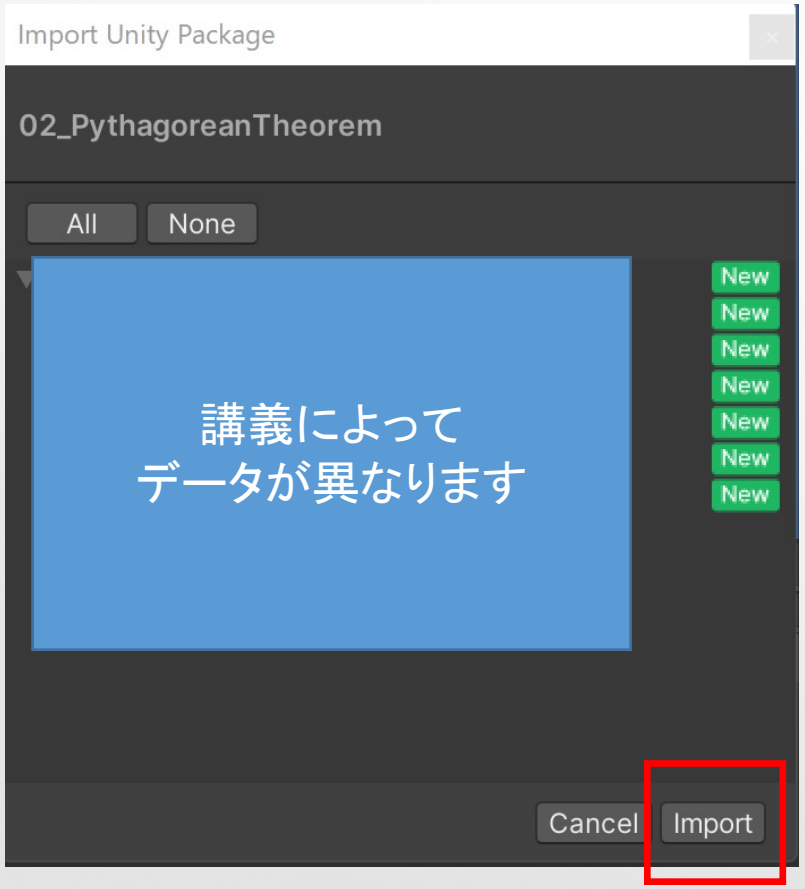
UnityPackageを使用しての準備

次に、unitypackageというパッケージファイル(データが詰め込まれたファイル)を読み込みます。



UnityPackageを使用しての準備

ファイルを指定し、以下の画面が出たらImportを押すと、ファイルが展開されます。



０．作成するゲーム

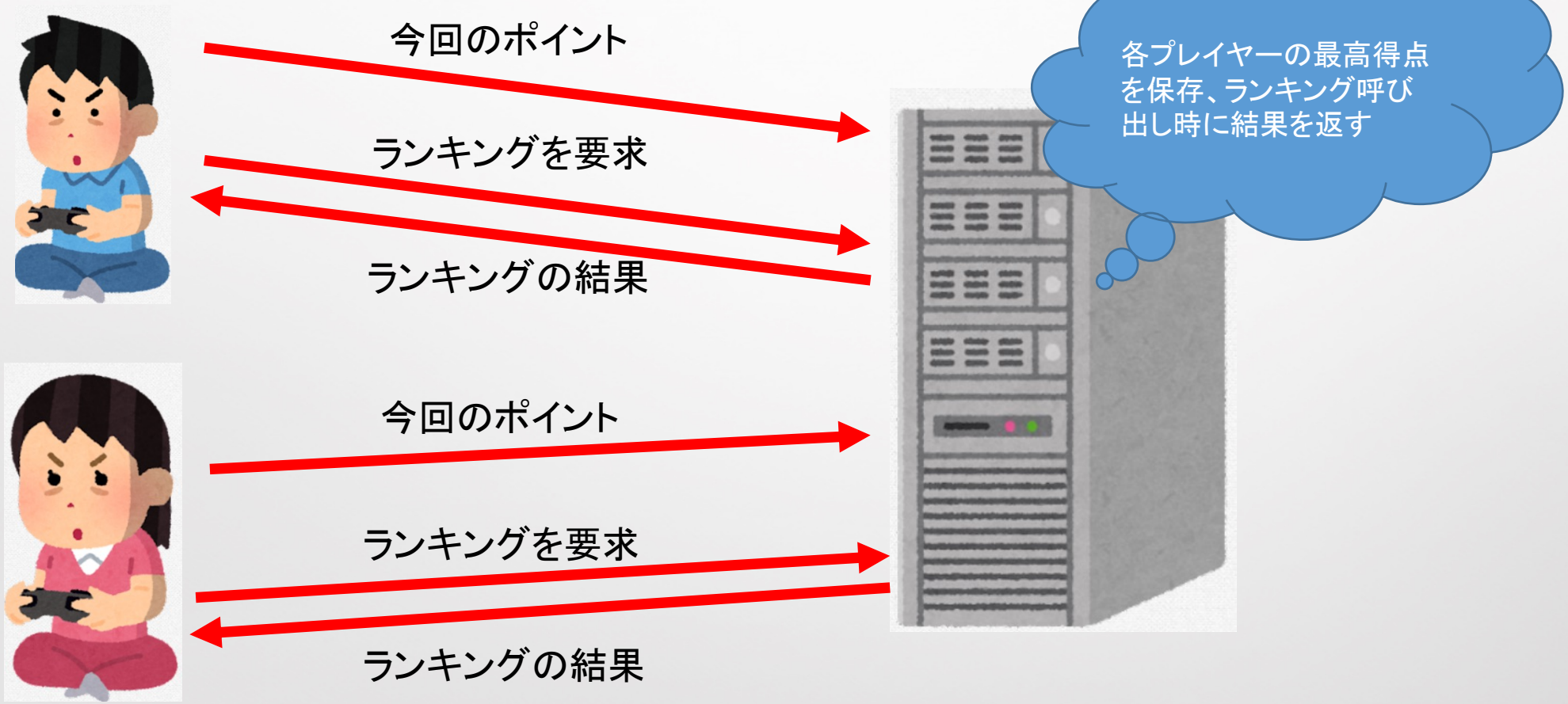
実際にプレイしてみよう！
(1文字のタイピングゲーム)

どんな仕組みで動いている？

１．外部サービスについて(BaaS)

サーバー側にポイント保存。

ランキング要求時は、ソートをして結果を返している。



１．外部サービスについて(BaaS)

フロント側(ゲーム)の他にサーバー側のプログラムを組む必要がある・・・？
でも、それを全部自分で実装するのはすごく大変。
なので、BaaS(Backend as a Service)というサービスを使ってみよう！
今回は有名なBaaSである、PlayFabを使ってどういったものか体験していく。

このように、UnityはUnity以外のサービスと連携して作成することが可能。
その場合は外部サービスの仕様書を読み込み、どのように設定していくか
確認しながら進めていく必要がある。

2．PlayFabの紹介

＜BaaS(Backend as a Service) / mBaaS(Mobile ～)＞

バックエンド側(サーバーが行う処理)はクラウドにお任せできるサービス。

(自分で実装しなくてもよい)

PlayFabだとこんなことが出来る。

- ユーザー管理
- ショップ情報
- 通貨/アイテム
- ランキング
- マッチング

etc…

２．PlayFabの紹介

＜PlayFabの特徴＞

・MicrosoftのAzureのサービスの1つ。

・小規模なインディーゲームからAAAタイトルまで使われている。

・PC(Unity/UE)、モバイル(Xcode/AndroidStudio)、
コンシューマ(PS/Switch)、などあらゆるプラットフォームに対応。

＜使うと嬉しいこと＞

・開発工数の削減(認証、課金、ランキング、マッチング etc)
自分で開発せずにアプリに組み込み可能

・バックエンドの要となるDBを用意、管理する必要がない！

・10万ユーザーまでは無料で使用可能

２．PlayFabの紹介

＜代表的なBaaS/mBaaS＞

・Firebase(Google)←非ゲーム用途

・GameSparks(Amazon)

・Azure PlayFab(Microsoft)

・ニフクラ(富士通)

非ゲーム用途での人気はFirebase一強。

ゲーム用途に関してもPlayFabが一強だが、GameSparksも追い上げてきている？？

２．PlayFabの紹介

＜PlayFabでは出来ないこと＞

・マルチプレイなどのリアルタイム通信

⇒PhotonやNetCodeなどを併用する必要がある

・サーバープログラムのホスティングやオーケストレーション

⇒別のAzureのサービスを使用することで可能

・マスタやユーザーデータをDBで管理

⇒他のAzureのサービスを使用することで可能

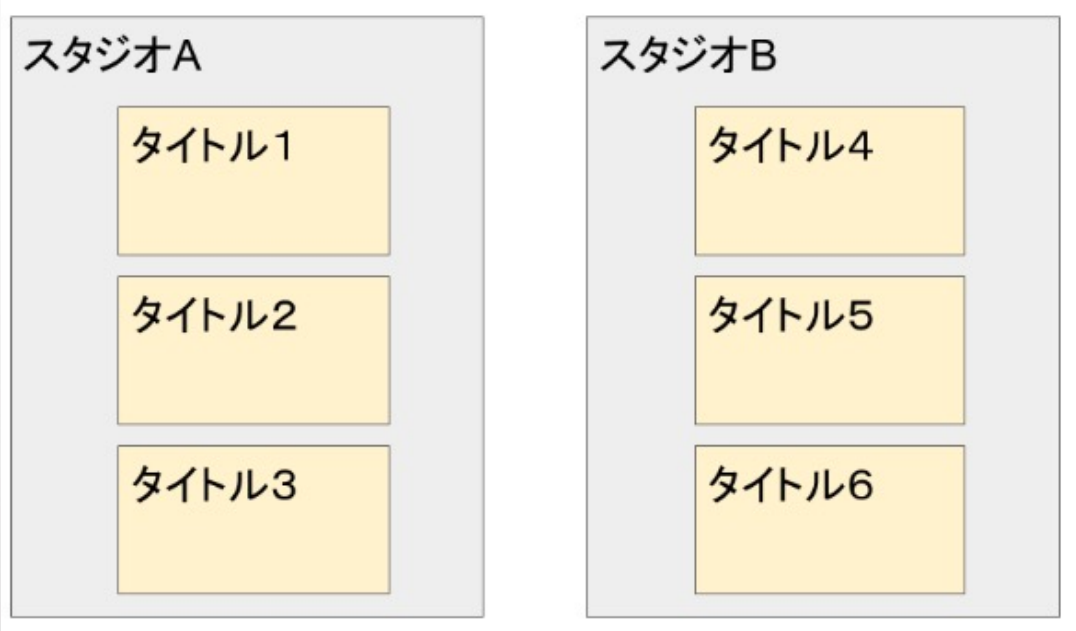
３．PlayFabの組み込み

＜PlayFabをUnityに導入する＞

・Playfab.comでアカウント作成

・スタジオとタイトルを作成

(タイトルを包括するものがスタジオ。スタジオA＝個人開発、
スタジオB ＝ チーム開発など)

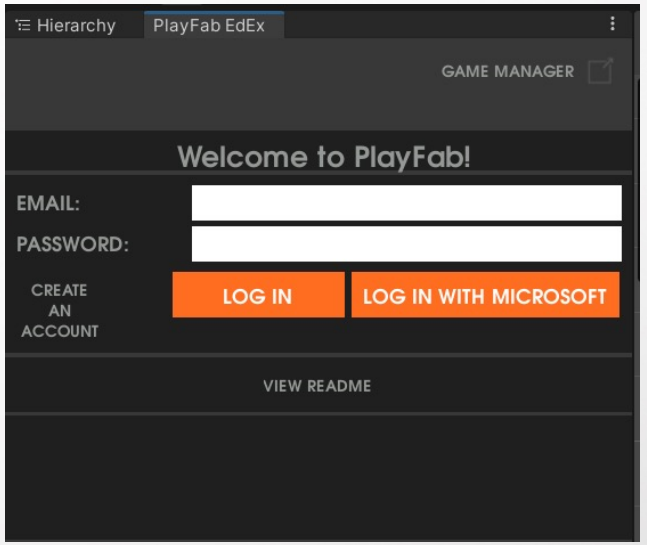


3．PlayFabの組み込み

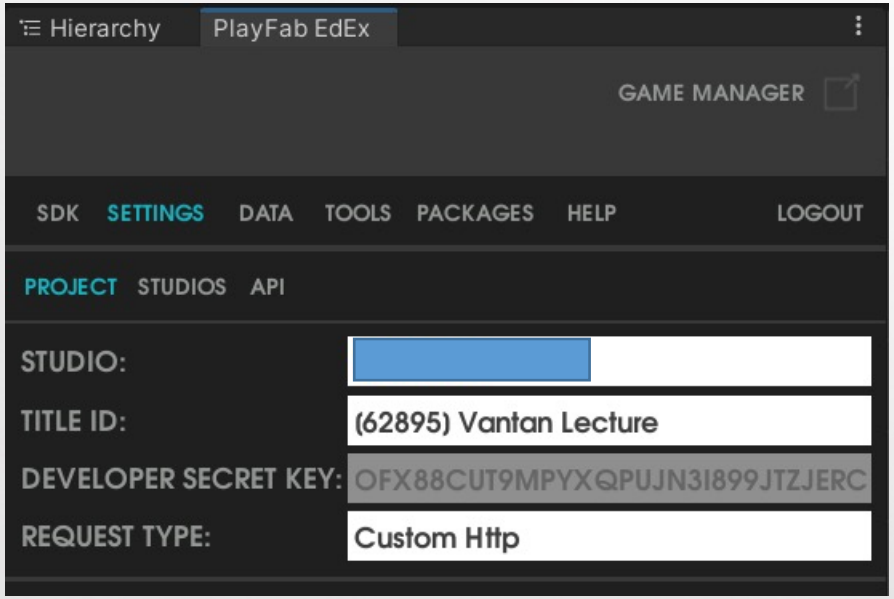
＜PlayFabをUnityに導入する＞

・SDK(Playfabサービスを使うためにパッケージ化したもの)をインポート

・UnityにてPlayFabにログイン、紐付けを実施する。



・スタジオとタイトルを設定する。



４．タイピングゲームの作成

＜シーン作成＞

・Title、Gameという名前のシーンを作成

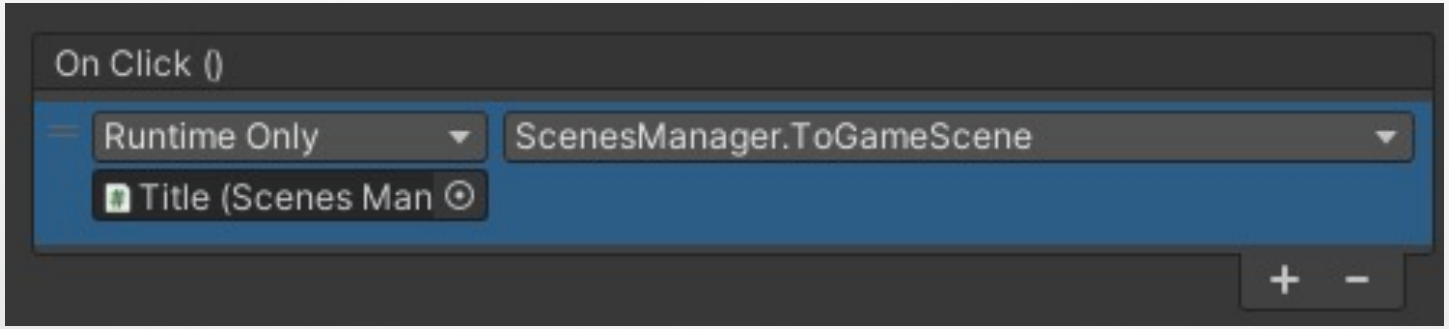
＜タイトル画面のUI設定＞

・ボタンを2つ作成してください。(ゲーム開始ボタンと、終了ボタン)

４．タイピングゲームの作成

＜スクリプト作成＞

・ScenesManager.csのスクリプトを作成し、空のGameObjectにアタッチ、
ゲーム開始ボタンを押したときに以下のメソッドを呼び出すように設定
⇒ゲーム開始ボタン

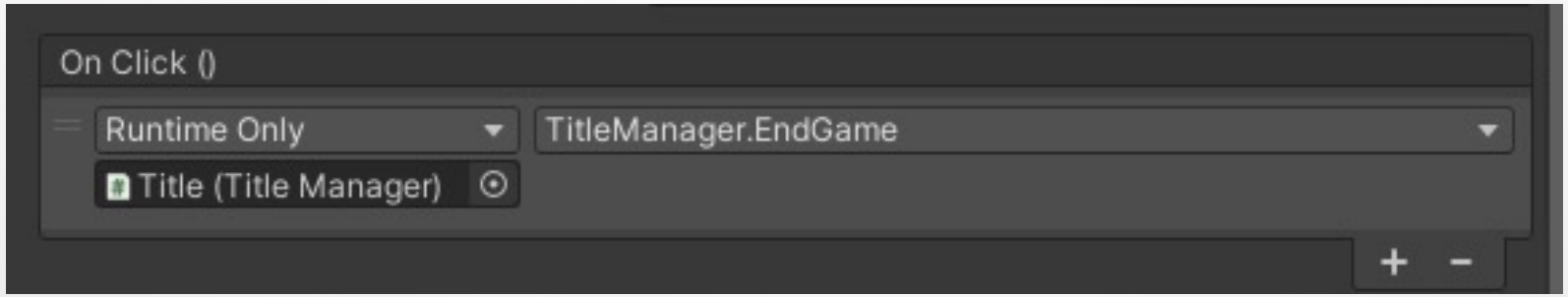


４．タイピングゲームの作成

＜スクリプト作成＞

・終了ボタンを押したときに以下のメソッドを呼び出すように設定

⇒終了ボタン

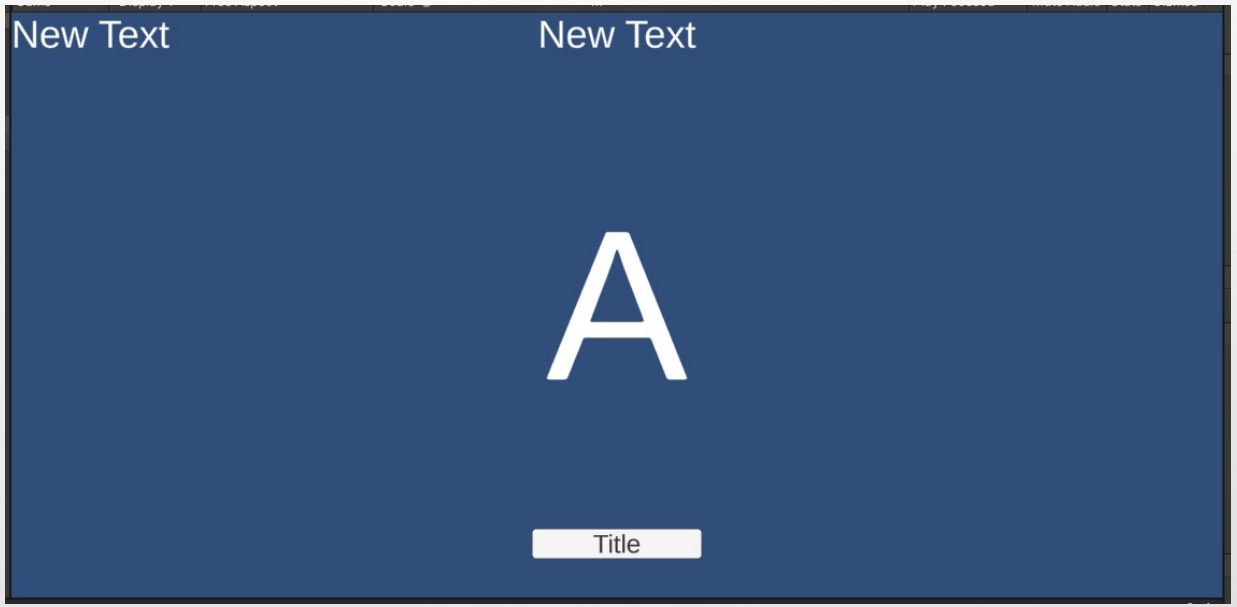


４．タイピングゲームの作成

＜ゲーム画面のUI設定＞

・TextMeshProのUIを3つ配置(左上のポイント、上中央の時間、中央のワード)

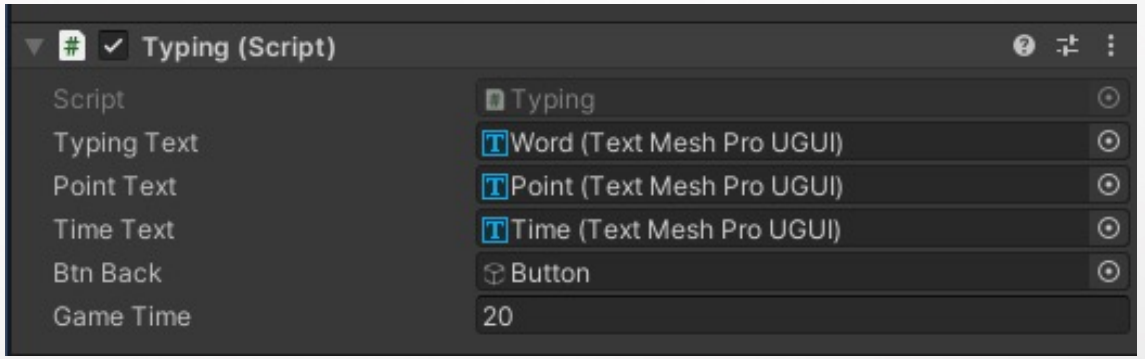
・下側にボタンを配置してください。



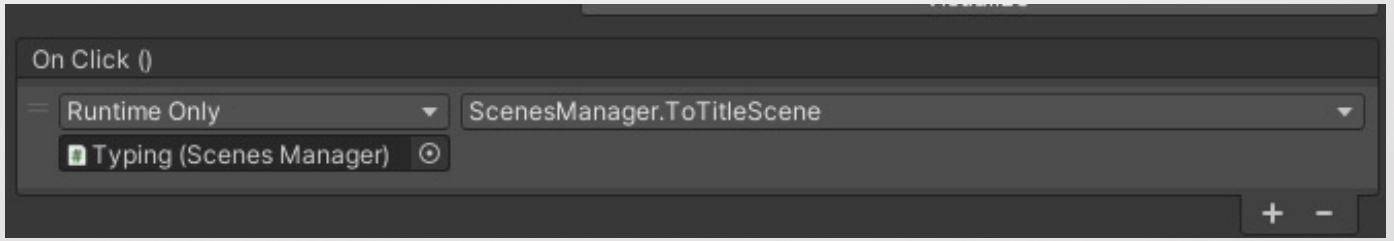
４．タイピングゲームの作成

＜スクリプト作成＞

・Typing.csのスクリプトを作成し、3つのTextMeshProをそれぞれアタッチ。
Game Timeは20秒に設定してください。
(赤枠はあとからでOK)



・ScenesManager.csをどこかのGameObjectにアタッチし、
ボタンが押されたときに以下のメソッドが呼ばれるように設定してください。



４．タイピングゲームの作成

＜プログラム解説＞

・Dictionary

⇒ Keyと呼ばれる名前と、Valueと呼ばれる値をセットで扱う連想配列。

例：) Key:0 Value:A / Key:1 Value:B

Keyの1番を指定したとき⇒Bの文字列が返ってくる。

KeyとValueの変数は指定可能

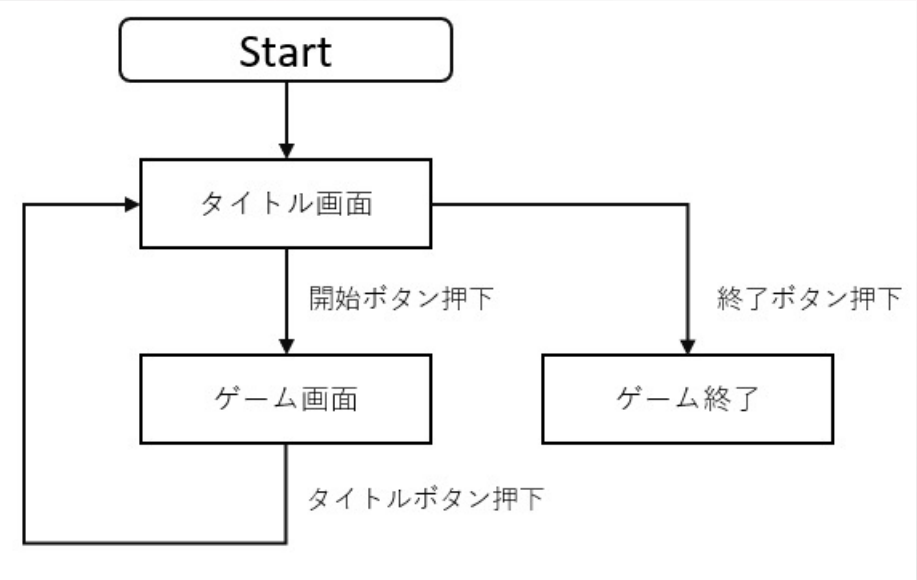
```
private Dictionary<int, string> dic;
```

５．PlayFabの要求処理の実装

＜概要＞

・４項で作成した中にPlayFabを実装していく。
この場合、今どういった遷移をするのか確認し、そこからどのように変化させていくのかを整理しながら進めていくと分かりやすい。

＜今の遷移方法＞



５．PlayFabの要求処理の実装

＜変更後のフローチャート＞

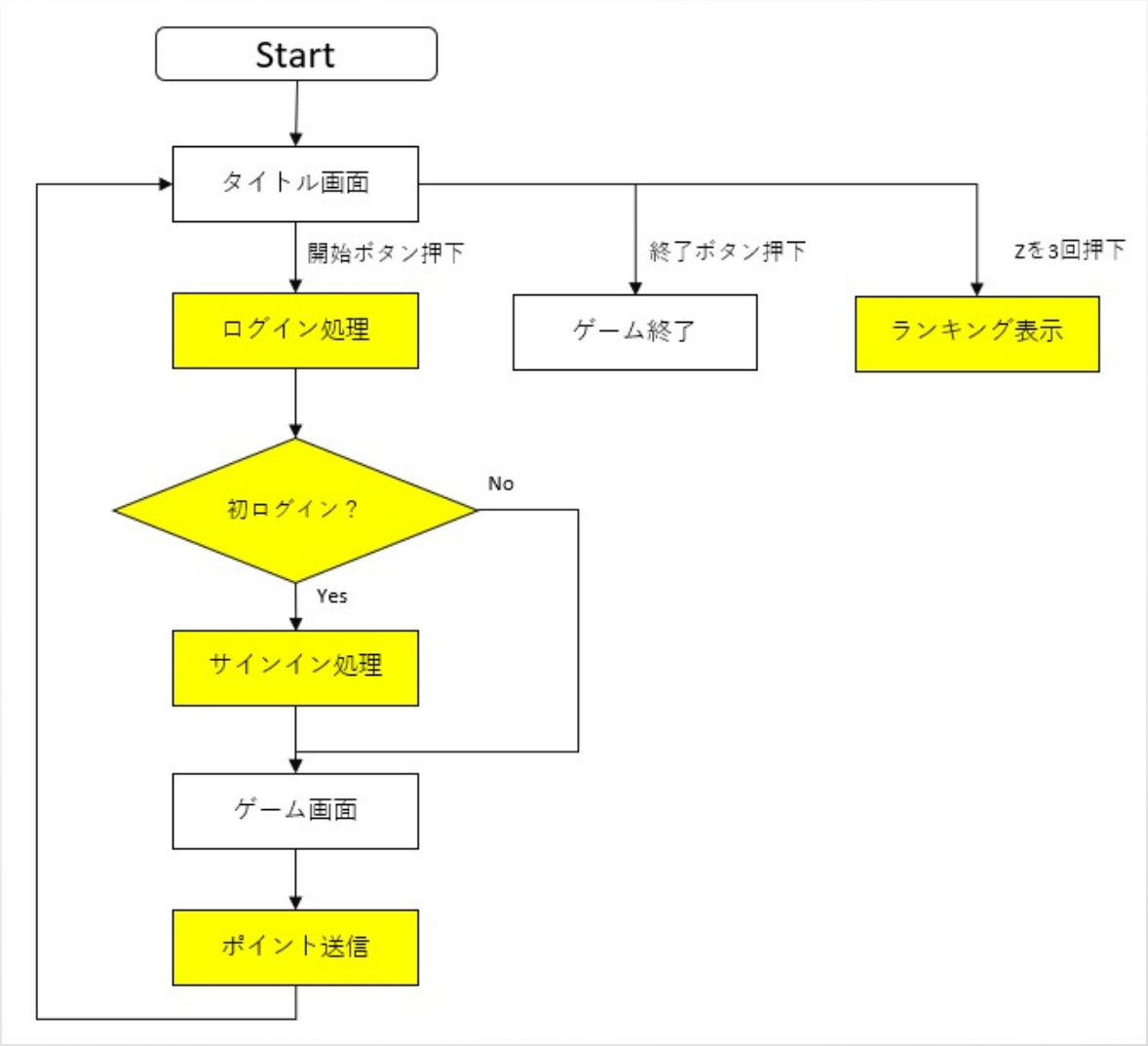
PlayFab実装に必要な機能

①ログイン機能の実装

②サインイン機能の実装

③ポイント送信機能の実装

④ランキング機能の実装



５．PlayFabの要求処理の実装

＜スクリプト作成＞

Typing.csを編集とRanking.csの作成もお願いします。

また、Unity上でも以下の変更をお願いします。

■Titleシーン

・ランキング表示用のPrefabのScroll Viewを設置

・空のGameObjectを作成、Login.cs/PlayfabData.cs/SignIn.csとRanking.csをアタッチ。

(RankingにはScroll View内のTextMeshProをアタッチ)

■Gameシーン

・Typing.csをアタッチしたGameObjectにPlayfabDataをアタッチ

5．PlayFabの要求処理の実装

＜ランキングの設定＞

・PlayFabログイン後、スタジオの横の歯車を押下、
API機能の以下の項目にチェックを入れる。
(フロントからランキングの更新を許可する設定)

５．PlayFabの要求処理の実装

＜スクリプト解説＞

・Login.cs

まずは、ローカルに保存しているcustomID情報があるかチェック。
あったら、すでにログインしている判断。無かったら16桁のID情報を
作成し、サインインする処理。
PlayFabClientAPI呼び出し時、CreateAccountがtrueなら
サインイン、falseならログインとなる。

・SingIn.cs

個々のプレイヤーには名前を付けることが出来る(DisplayName)
今回はLogin.csで作成したcustomIDの情報を名前としている。

５．PlayFabの要求処理の実装

＜スクリプト解説＞

・PlayfabData.cs

データ読み出しや書き込み、ランキング更新や読み込み処理の実装。
各プレイヤー個別にデータを設定することが可能。今回は毎回最後のポイントのデータを書き込みしている。(読み出しはしていない)
その他、ゲーム終了時、ランキング更新処理を実施している。