

ニフティクラウドmobiel backend
UnitySDKハンズオン

ニフティ株式会社

目次

- 1 ハンズオン前の準備
- 2 各アセットのインポート
- 3 ゲームオブジェクトの設定準備
- 4 NCMBの設定
- 5 NCMBデータストアの活用

目次

- 1 ハンズオン前の準備
- 2 各アセットのインポート
- 3 ゲームオブジェクトの設定準備
- 4 NCMBの設定
- 5 NCMBデータストアの活用

ハンズオン前の準備 タスクリスト

- 1 mobile backendの登録
- 2 本日のサンプルのダウンロード
- 3 Unityちゃん、Japanese Otaku CityのDL

mobile backendの登録

お申込みページ

ID登録後、プッシュ通知配信画面や会員管理画面など、コントロールパネルの中をご覧いただけます。

ID登録（即時・無料）

ID登録・無料でお申し込みいただけます。

サインアップ

個人会員として登録ご希望の方

STEP1

ニフティクラウド mobile backendをご利用いただくには、@niftyのメールアドレスを取得する必要があります。

[@nifty会員の登録（無料）](#)

STEP2

@nifty会員への登録完了後「ニフティクラウド mobile backend」にログインし、新規お申し込み手続きを行ってください。新規お申し込み後、引き続きサービス利用ができます。

[ニフティクラウドmobile backendログイン](#)

[> 利用規約](#)

法人会員として登録ご希望の方

ニフティの法人会員としてご契約いただく、ニフティクラウド mobile backend利用料金を企業・団体にまとめてお支払いいただけます。Webまたは紙にて@nifty法人ID取得手続きを行ってください。なお、法人としてご契約いただけるのは、企業/各種法人/各種団体・グループ単位となります。

STEP1 @nifty法人ID取得

「@nifty法人ID取得フォーム」から、法人ID取得手続きを行ってください。なお、申込書（紙版）でのお申し込みも可能です。

[@nifty法人ID取得フォーム（無料）](#)

すでに@nifty法人IDを取得されている方で、mobile

@nifty

@nifty会員登録

[step1] 登録内容の入力

[登録の流れ]

[step1]
登録内容の入力

[step2]
登録内容の確認

[step3]
登録完了

@nifty会員へご登録いただきますと、@niftyのサービスがご利用できます。登録料や会費はもともと無料です！ご利用料金は有料サービスのご利用時のみ必要になります。

基本情報の入力

※必須項目

※@niftyユーザー名

※@niftyのサービスへログインする時に使用します。

90文字以内（半角英数字）

※ユーザー名が使用できるか調べる

パスワード

※必須項目

※@niftyのサービスへログインする時に使用します。

[半角英数字]

パスワードの安全性

確認のため、もう一度入力してください。

連絡先メールアドレス

※いずれか必須項目

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

※@niftyのサービスへログインする時に使用します。

niftyID取得画面

本日のサンプルのDL

1

- GithubよりDL
- https://github.com/hounenhounen/NCMB_handson_hacosco

2

• 内容物の確認

本日使用するUnity
プロジェクト



本資料のPDF版と
パワポ版



Unityちゃん、Japanese Otaku CityのDL

UnityChan "Unity-chan!" model



<https://www.assetstore.unity3d.com/jp/#!/content/18705>

Japanese Otaku City

Japanese Otaku City



<https://www.assetstore.unity3d.com/jp/#!/content/20359>

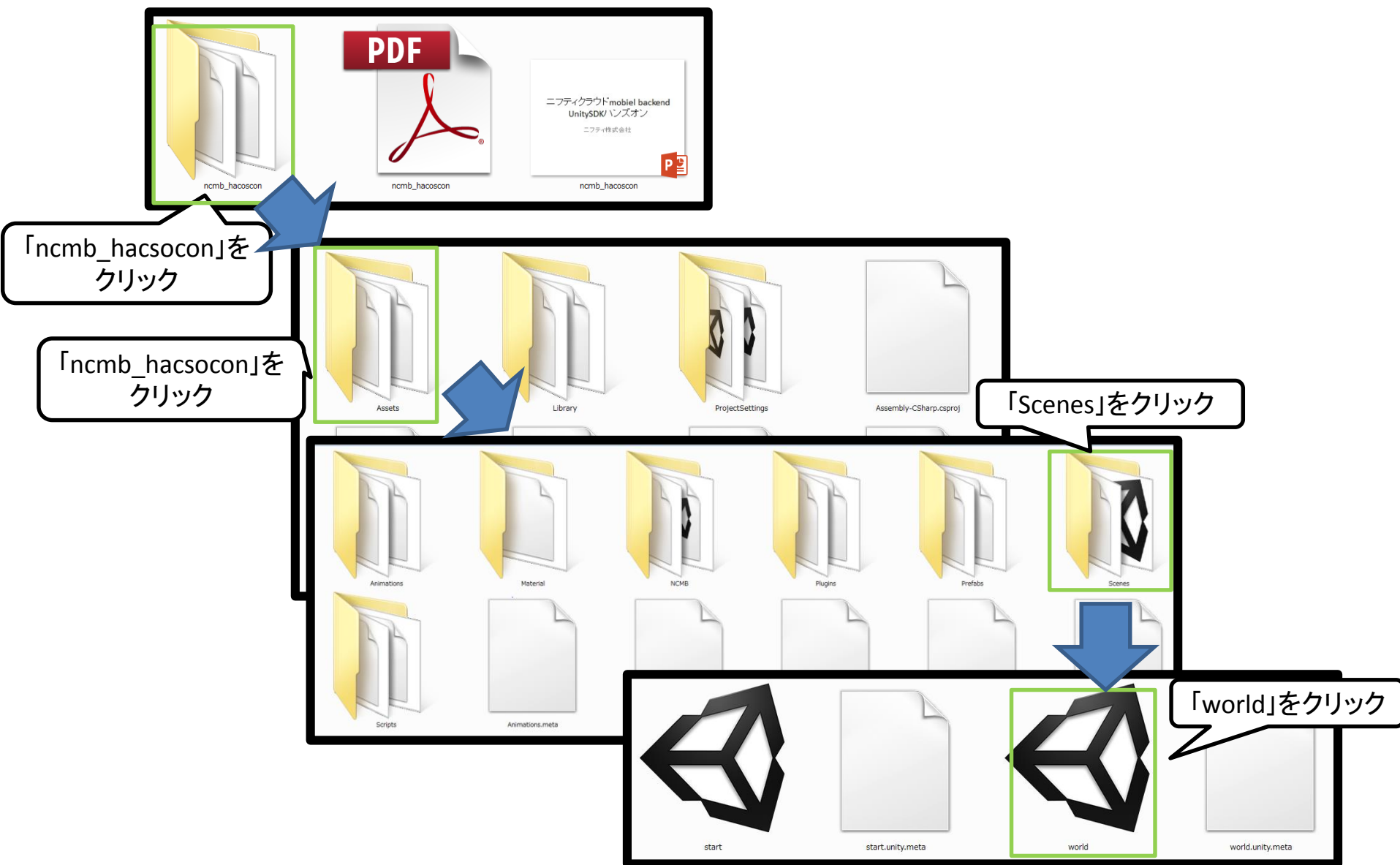
目次

- 1 ハンズオン前の準備
- 2 各アセットのインポート
- 3 ゲームオブジェクトの設定準備
- 4 NCMBの設定
- 5 NCMBデータストアの活用

各アセットのインポート タスクリスト

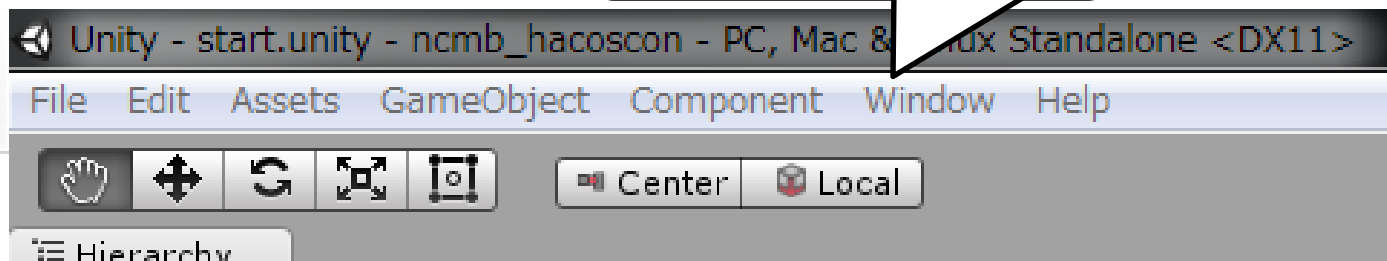
- 1 配布資料プロジェクトの開封
- 2 各アセットをプロジェクトインポート
- 3 Otaku City・Unityちゃんの設置

配布資料プロジェクトの開封

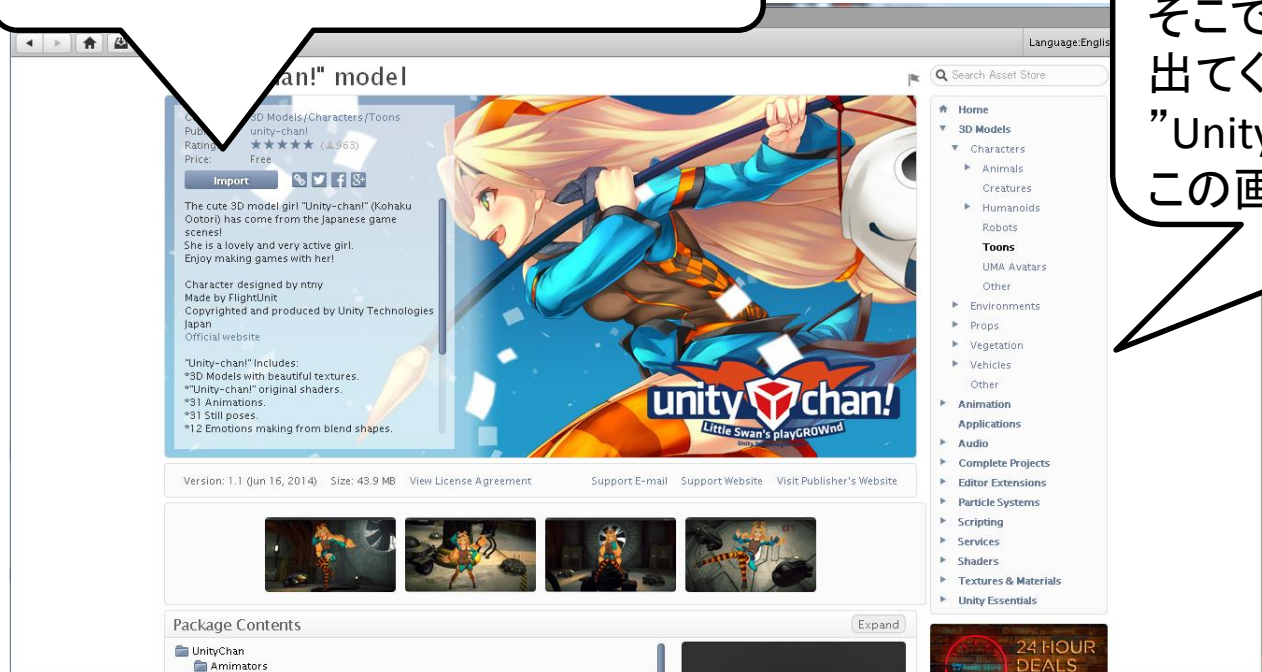


あらかじめアセットをDLLしていた場合

Window>AssetStoreを
クリックすると



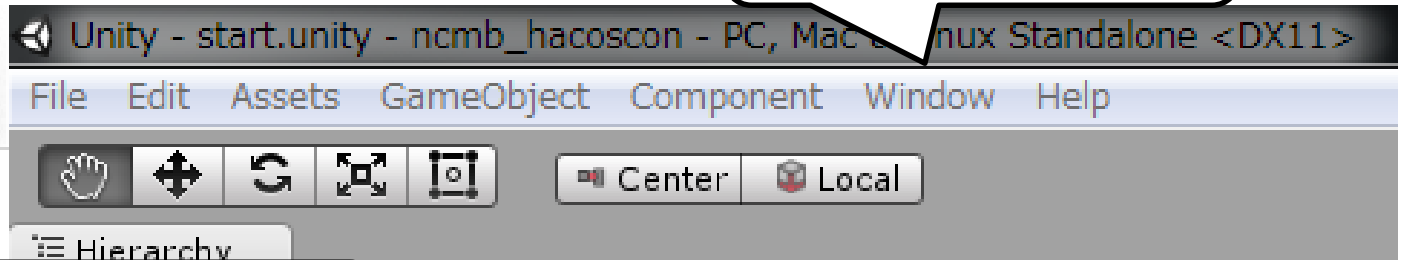
「Import」を押すとImportできます



Unityのブラウザが立ち上がるので
そこで、「Unitychan」と検索すると
出てくるので
”Unity-chan” modelを選ぶと
この画面が出てきます

本ハンズオンで始めてDL・インポートする場合

Window>AssetStoreをクリックすると



「Download」を押すとDLが始まり、
そのご「Import」を押すことができます



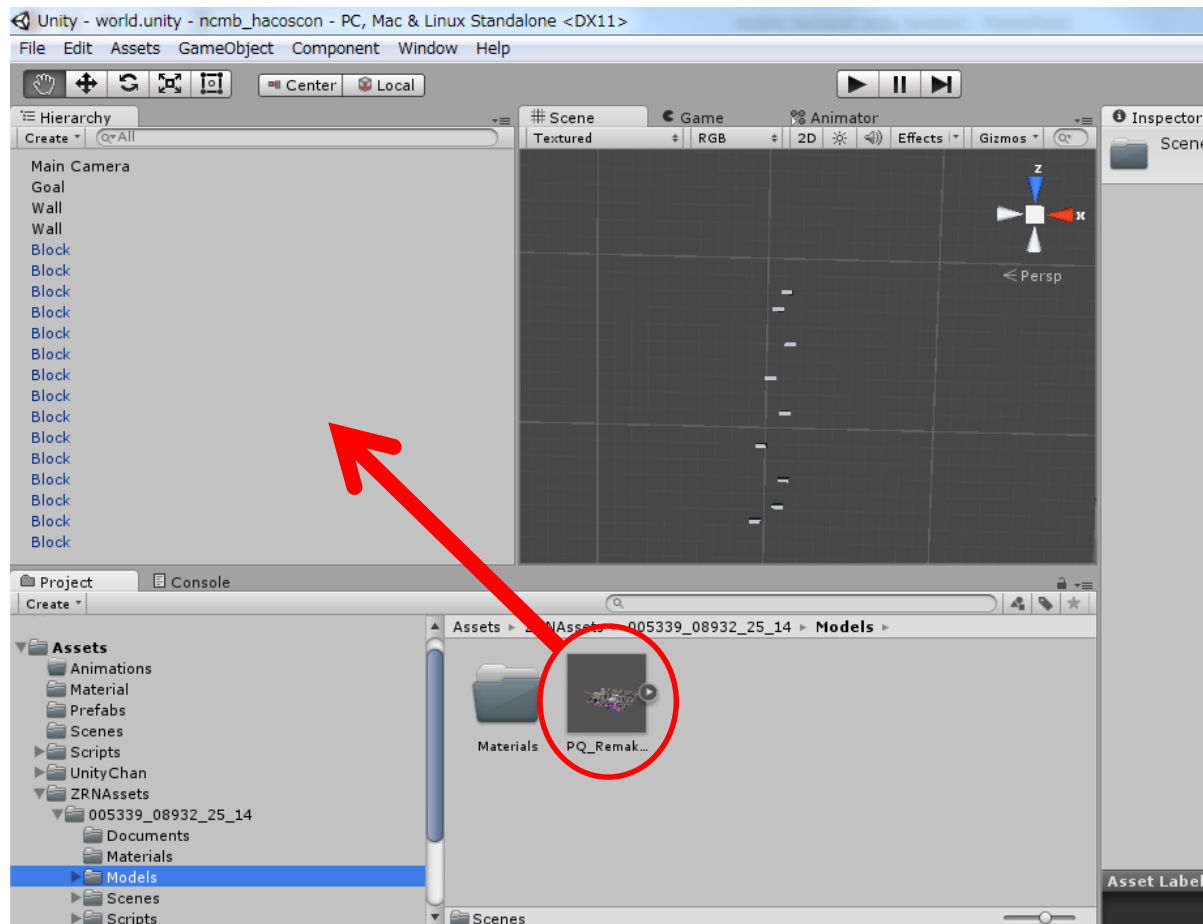
Unityのブラウザが立ち上がるので
そこで、「Japanese Otaku City」と検
索するとこの画面が出てきます

Otaku Cityの設置

※worldシーンを開いてください

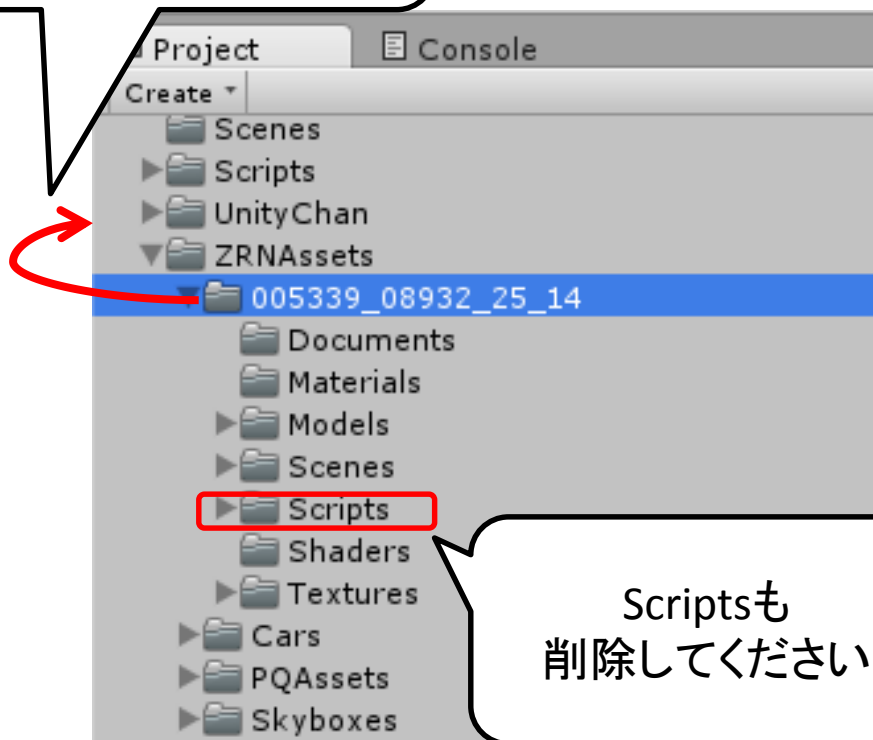
Assets>ZRNAssets>005339_08932_25_14>Models

PQ_Remake_AKIHABARA_Import SettingsをHierarchyにドラック&ドロップ



Otaku Cityの設定

ドラック&ドロップで
Assetの直配下へ



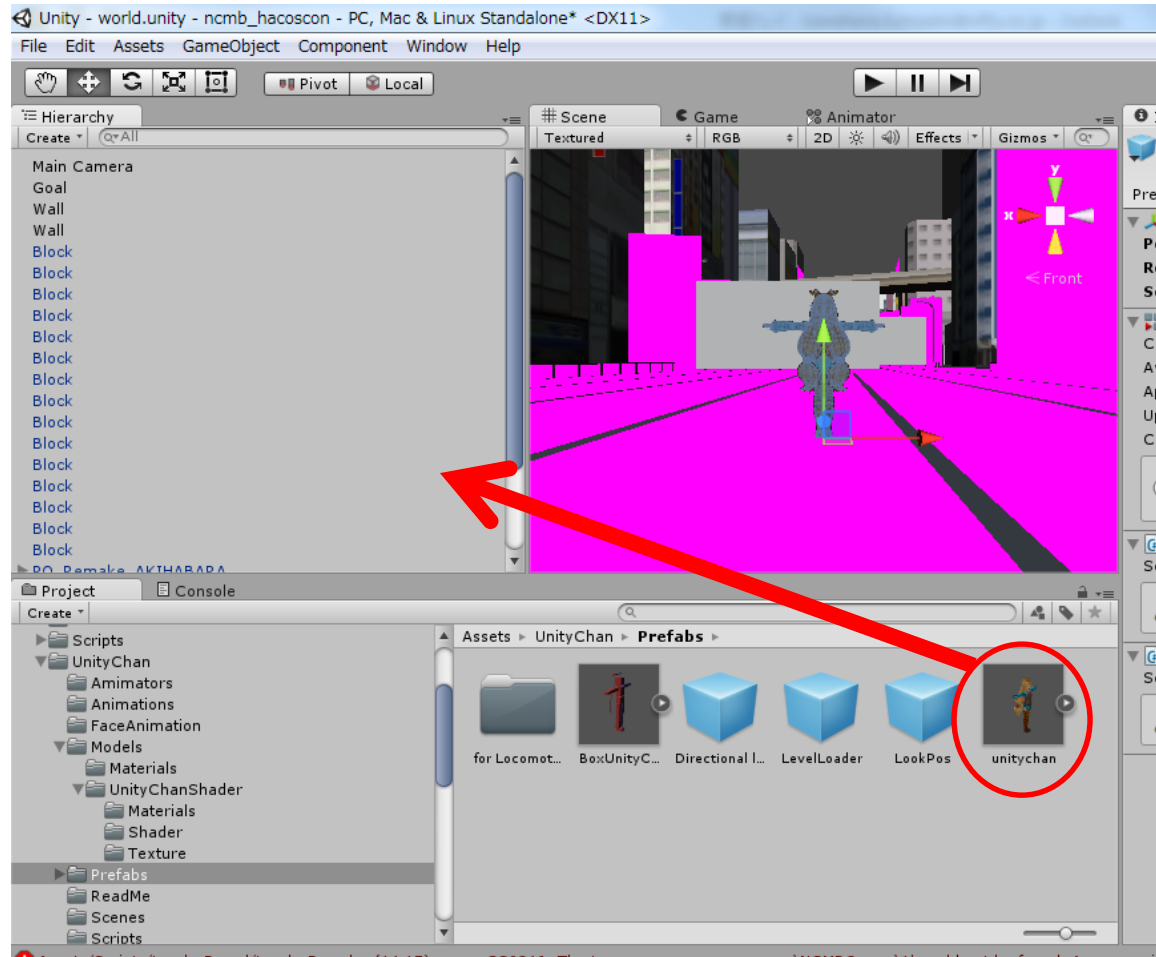
005339_08932_25_14 のフォルダを
ZRNAssetsフォルダの配下から
Assetの直配下に移動してください

その後ZRNAssetsを削除してください
また005339_08932_25_14
のScriptsフォルダも削除してください
※このScriptsが原因でエラーが出ている
端末もあるかと思います。

Scriptsも
削除してください

Unityちゃんの設置

Assets > UnityChan > Prefabs UnitychanをHierarchyにドラック & ドロップ



インスペクターでUnityChanの位置を(-9, 0.3, -21) ローテーションを(0, 180, 0)
スケールを(0.1, 0.1, 0.1)にあわせる

目次

- 1 ハンズオン前の準備
- 2 各アセットのインポート
- 3 ゲームオブジェクトの設定準備
- 4 NCMBの設定
- 5 NCMBデータストアの活用

ゲームオブジェクトの設定準備 タスクリスト

- 1 ライトの設定
- 2 Unityちゃんの設定
- 3 Cameraの設定

ライトの設定

Assets > UnityChan > Prefabs Directional light for UnityChanをHierarchyにドラック＆ドロップ

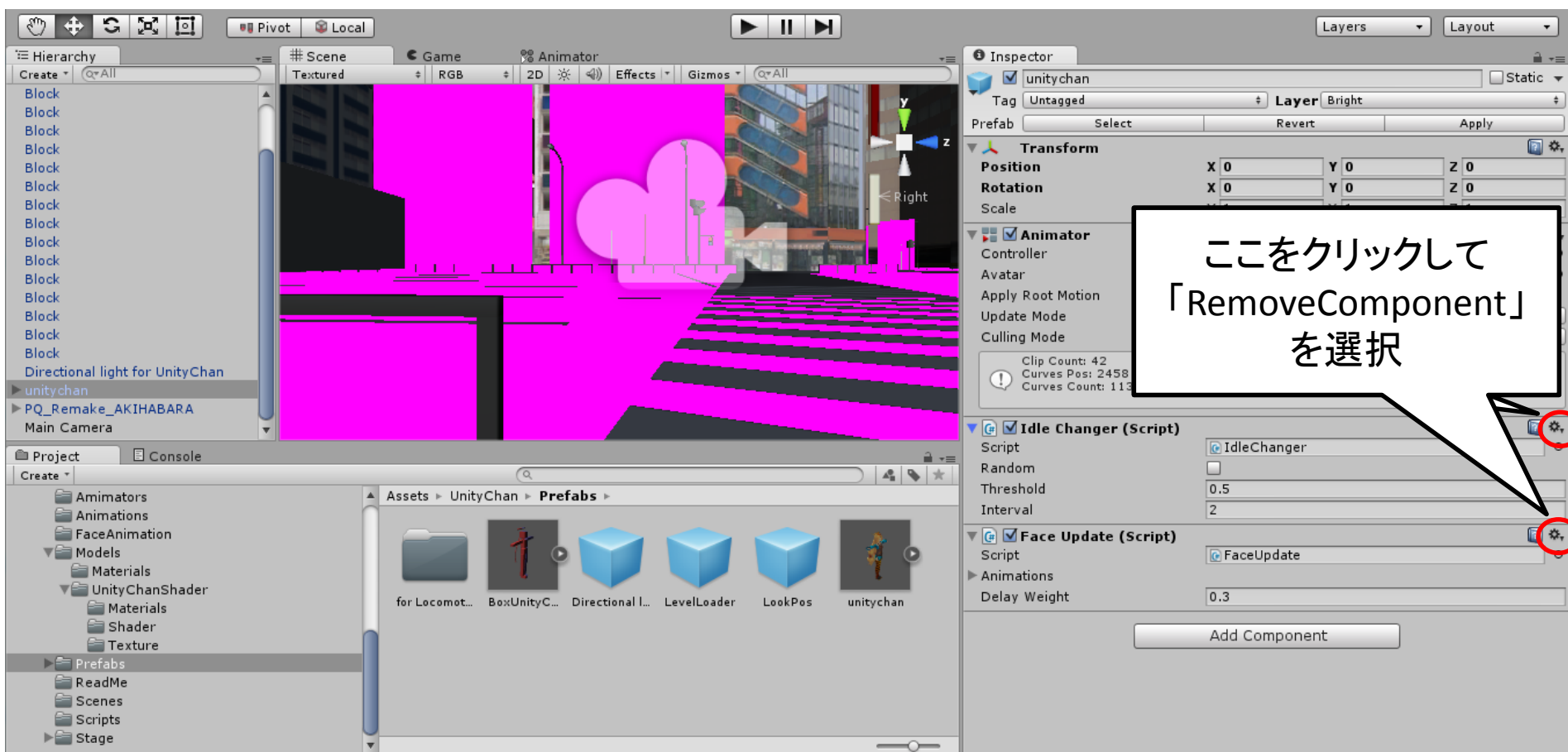


Directional lightの影響範囲をUnityちゃんに限定するため、BrightというLayerを追加し、UnityちゃんのLayerをBrightに、Directional lightのCulling MaskをBrightに設定します。

Unityちゃんの設定1

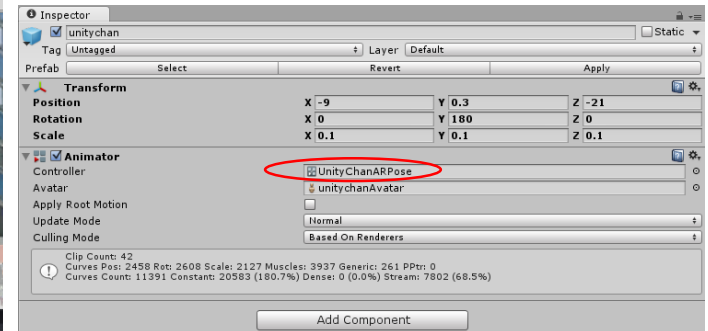
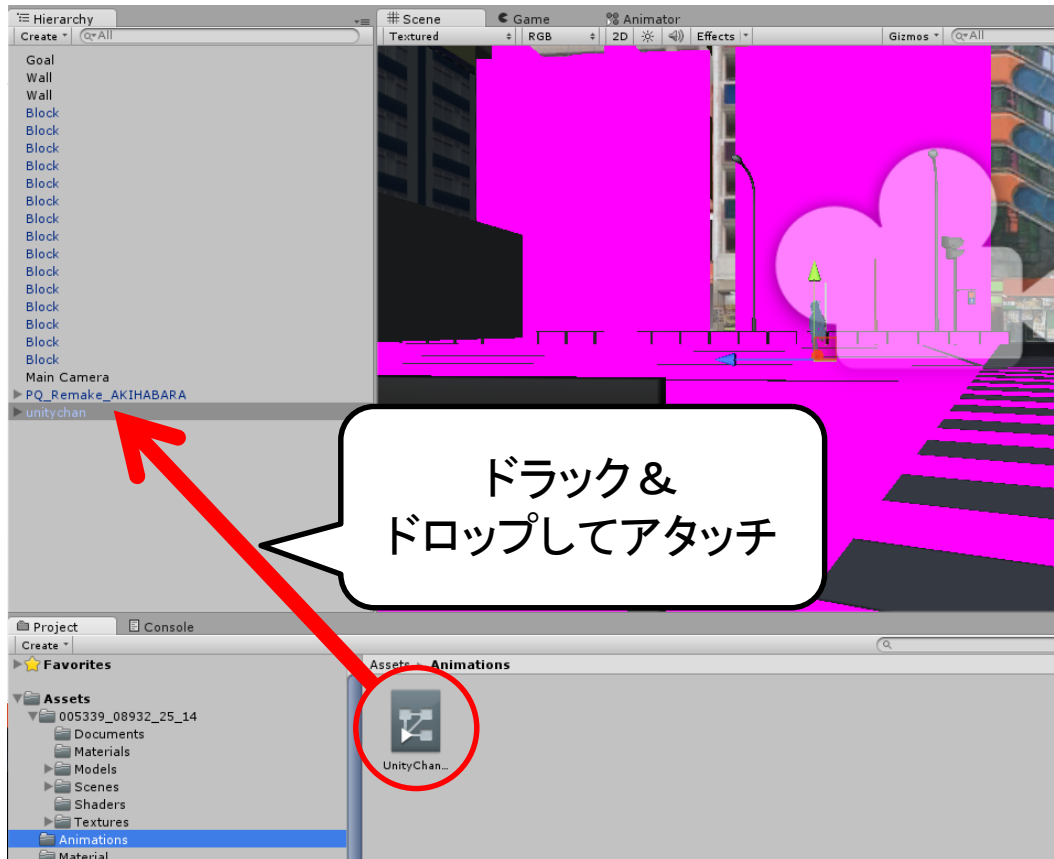
現状のUnityちゃんにアタッチされているスクリプト

「IdleChanger」「FaceUpdate」をRemoveComponentします

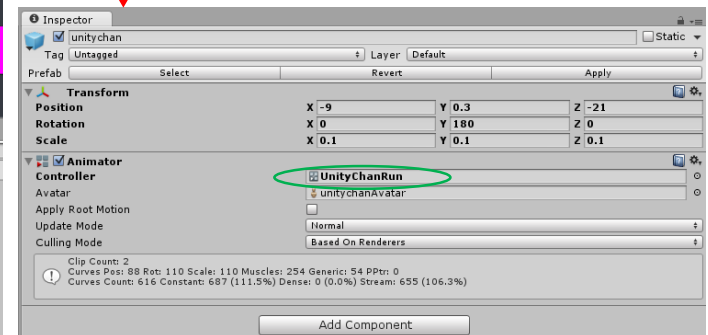


Unityちゃんの設定2

Assets>Animations内の UnityChanRunをUnityちゃんに
アタッチしアニメーションコントローラを変更します

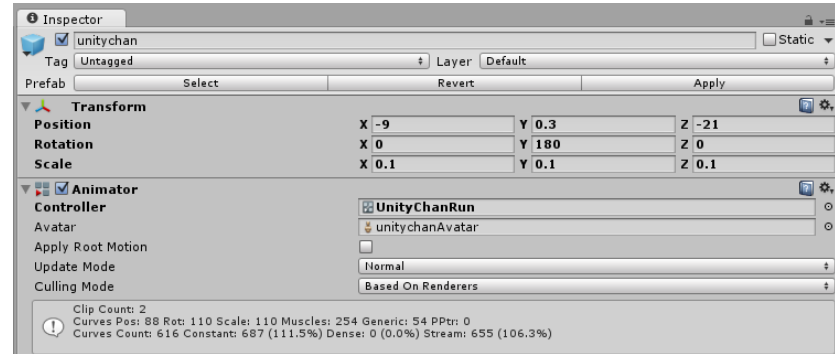
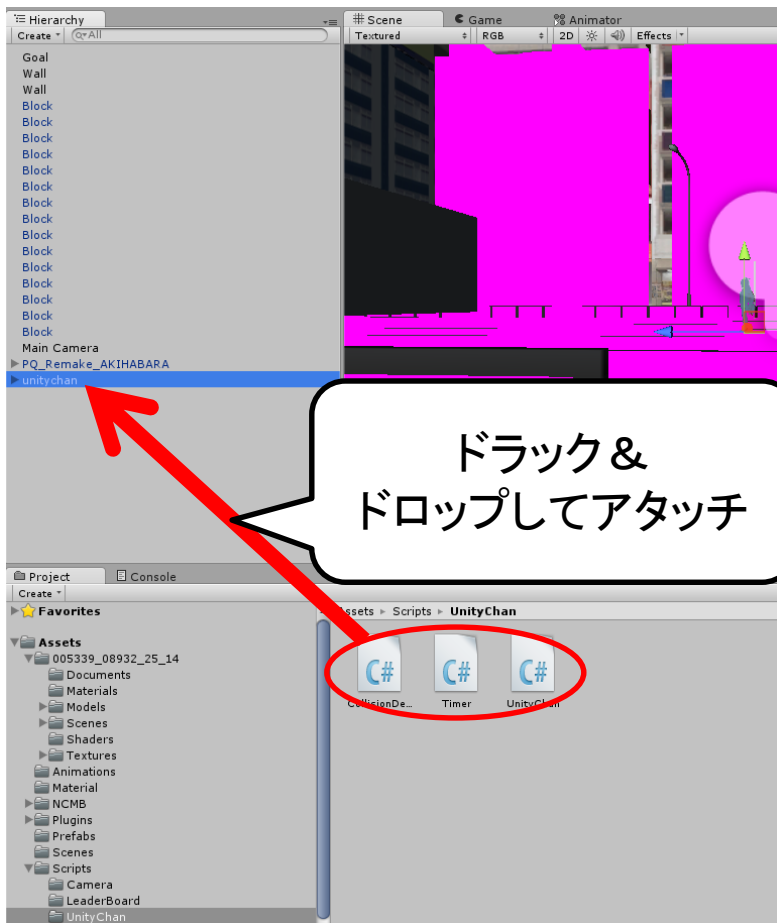


アタッチ後はインスペクターの赤丸部
が下記の緑丸のように変わります

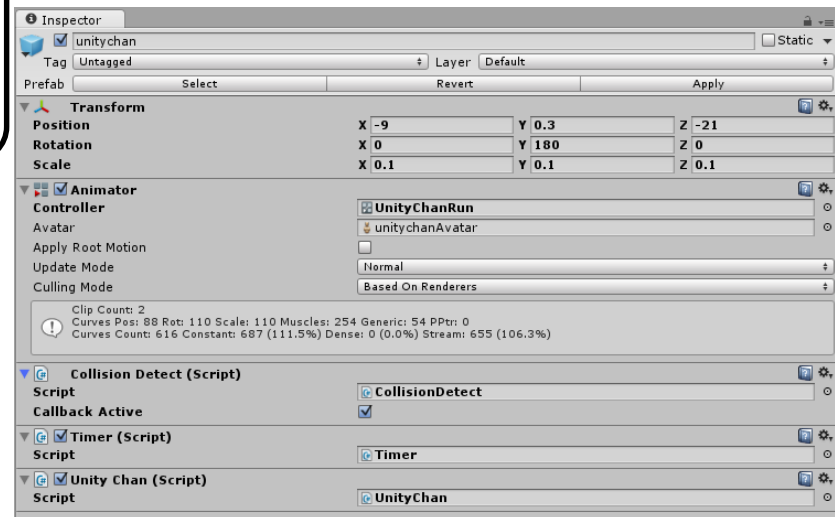


Unityちゃんの設定3

Assets>Scripts>UnityChan内にあるスクリプト(CollisionDetect、Timer、UnityChan)をそれぞれUnityちゃんにアタッチします

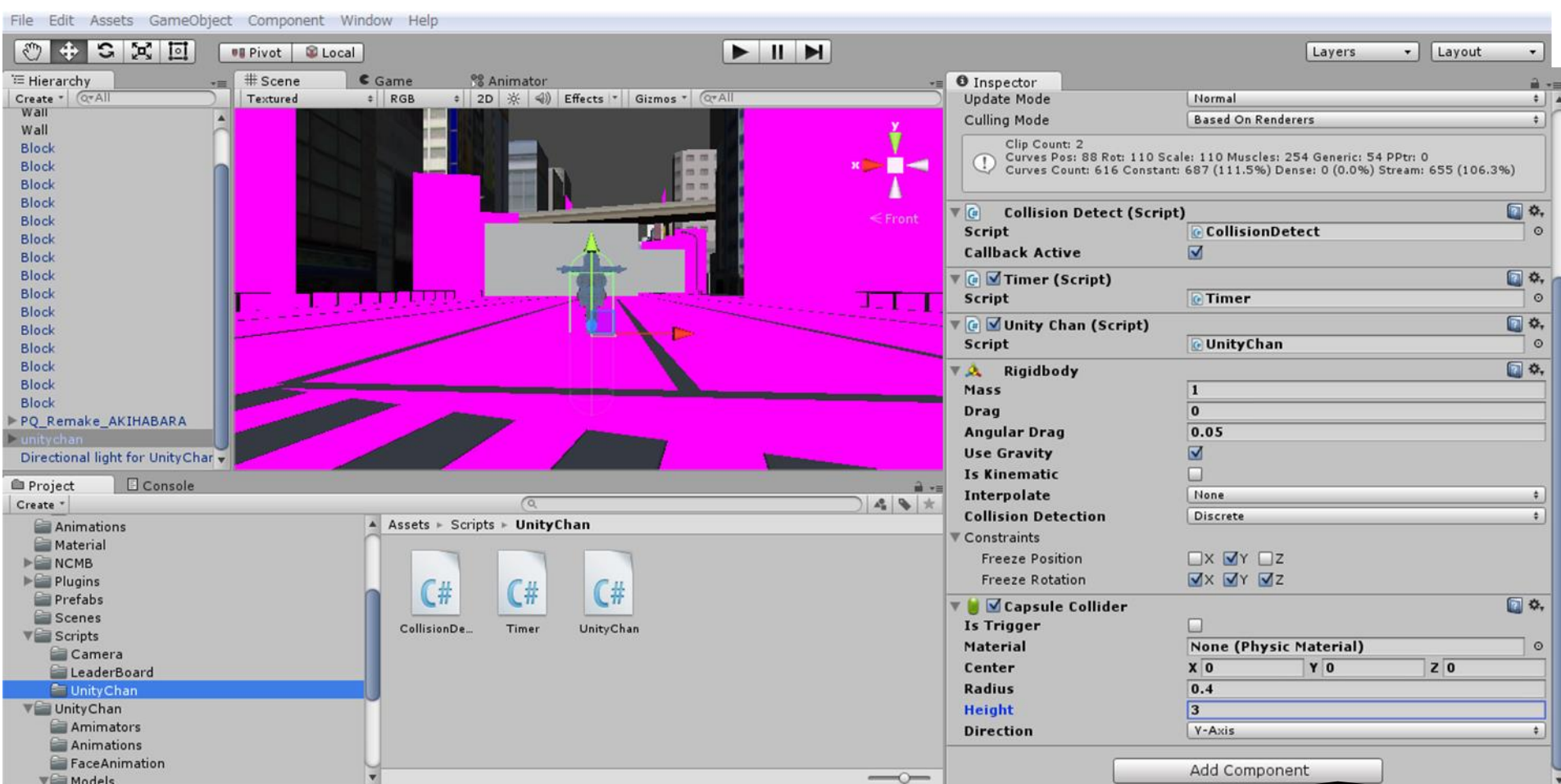


アタッチ後はインスペクターに
アタッチしたスクリプトが追加されます



Unityちゃんの設定4

AddComponentでRigidBody Capsule Colliderを加えて下記のように設定します



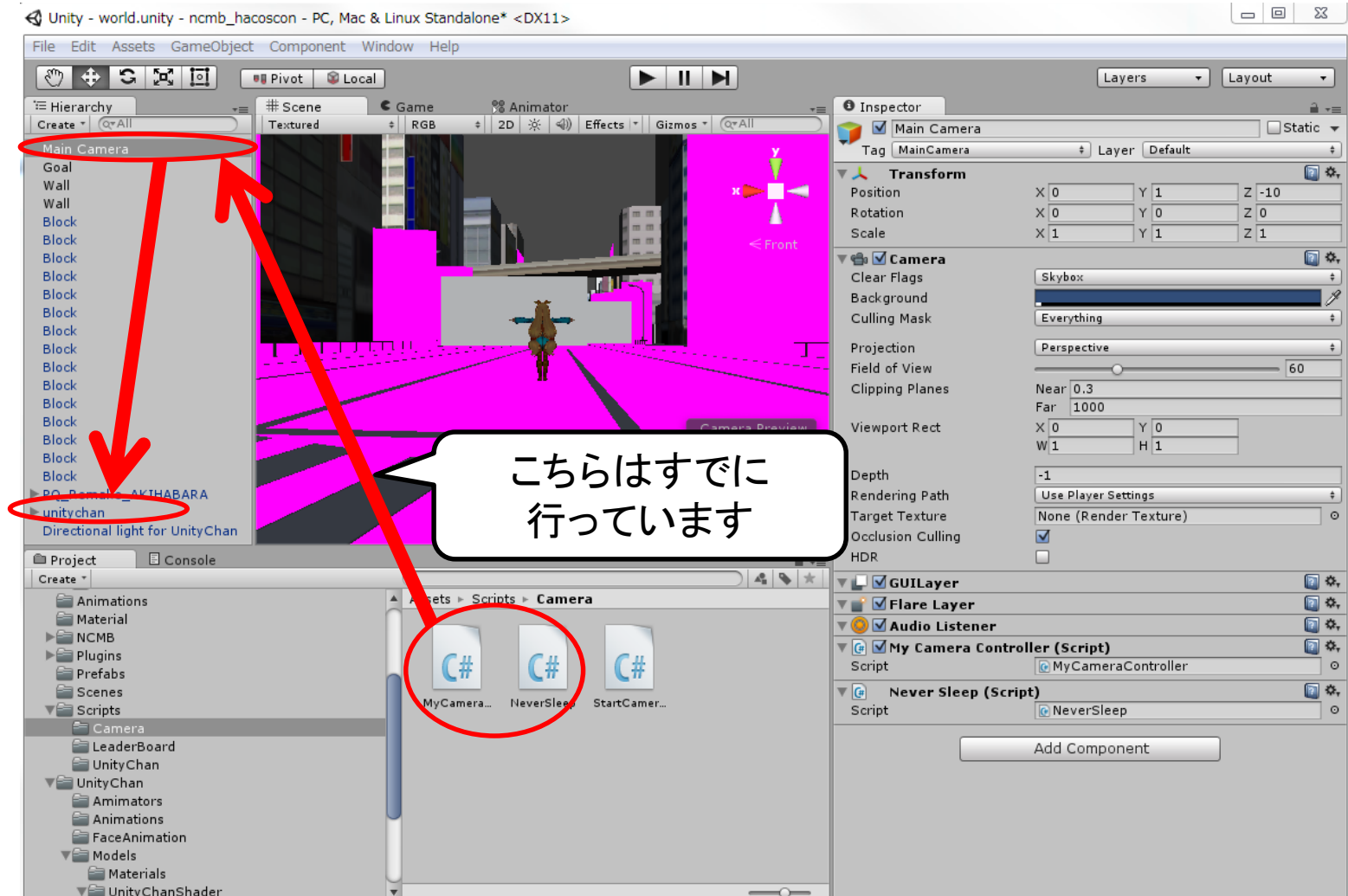
ここをクリックしたら
「RigidBody」、「Capsule Collider」などの
コンポーネントが追加できます

Cameraの設定

Assets>Scripts>Camera内にあるスクリプト「MyCameraController」「NeverSleep」をそれぞれアタッチ

※上記の内容はすでに行っています

Main CameraをUnityChanにアタッチ



目次

- 1 ハンズオン前の準備
- 2 各アセットのインポート
- 3 ゲームオブジェクトの設定準備
- 4 NCMBの設定
- 5 NCMBデータストアの活用

NCMBの設定タスクリスト

- 1 NCMBにアプリを作る
- 2 SDKをDL
- 3 SDKの組み込み

mobile backendでの新規アプリ作成

mobile backendへログインをお願いします！



アプリの新規作成

アプリ名

半角英数字もしくはアンダースコア

戻る 新規作成

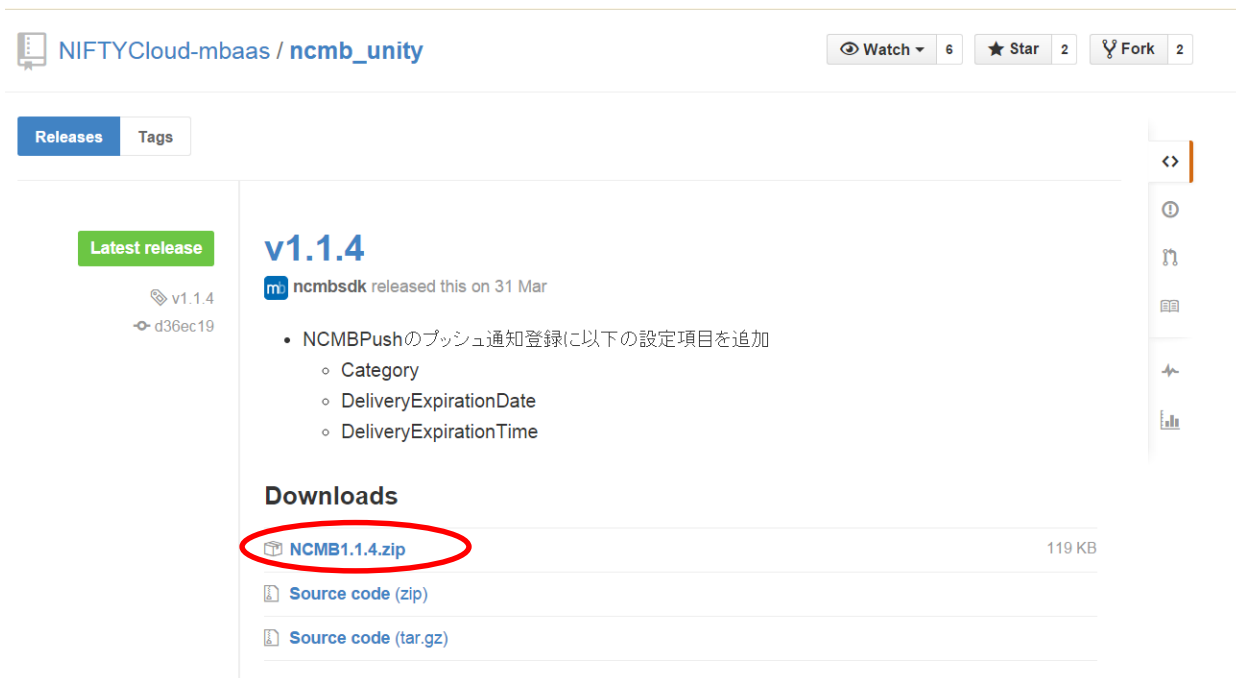
このような画面が表示されます

akibarunと入力して「新規作成」をクリック
本アプリ用のサーバー環境ができます

SDKをDL※こちらはすでにやっております

https://github.com/NIFTYCloud-mbaas/ncmb_unity/releases

のページを開くと下記のページが開かれます赤丸の部分をクリックしてSDKをDLします



The screenshot shows the GitHub repository page for `NIFTYCloud-mbaas / ncmb_unity`. The page has a header with the repository name and navigation links for Watch (6), Star (2), and Fork (2). Below the header, there are tabs for Releases and Tags. The Releases tab is active, showing the latest release, v1.1.4, which was released on 31 Mar. The release details include a list of items to add to the NCMBPush push notification registration: Category, DeliveryExpirationDate, and DeliveryExpirationTime. Under the Downloads section, the file `NCMB1.1.4.zip` is highlighted with a red circle, indicating it is the file to be downloaded. The file size is 119 KB. Other download options include Source code (zip) and Source code (tar.gz).

NIFTYCloud-mbaas / ncmb_unity

Watch 6 Star 2 Fork 2

Releases Tags

Latest release

v1.1.4

ncmb sdk released this on 31 Mar

- NCMBPushのプッシュ通知登録に以下の設定項目を追加
 - Category
 - DeliveryExpirationDate
 - DeliveryExpirationTime

Downloads

NCMB1.1.4.zip 119 KB

Source code (zip)

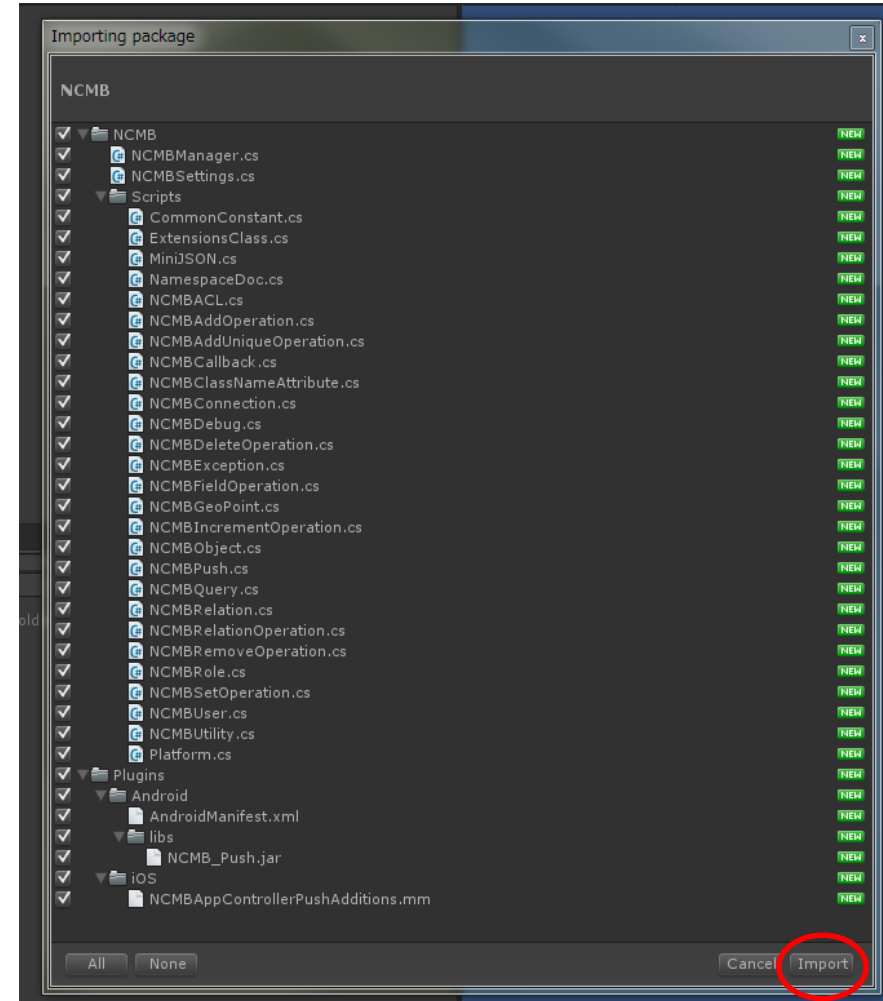
Source code (tar.gz)

SDKをインポート ※こちらはすでにやっております

先ほどDLしたサンプルゲーム
を開いた状態で

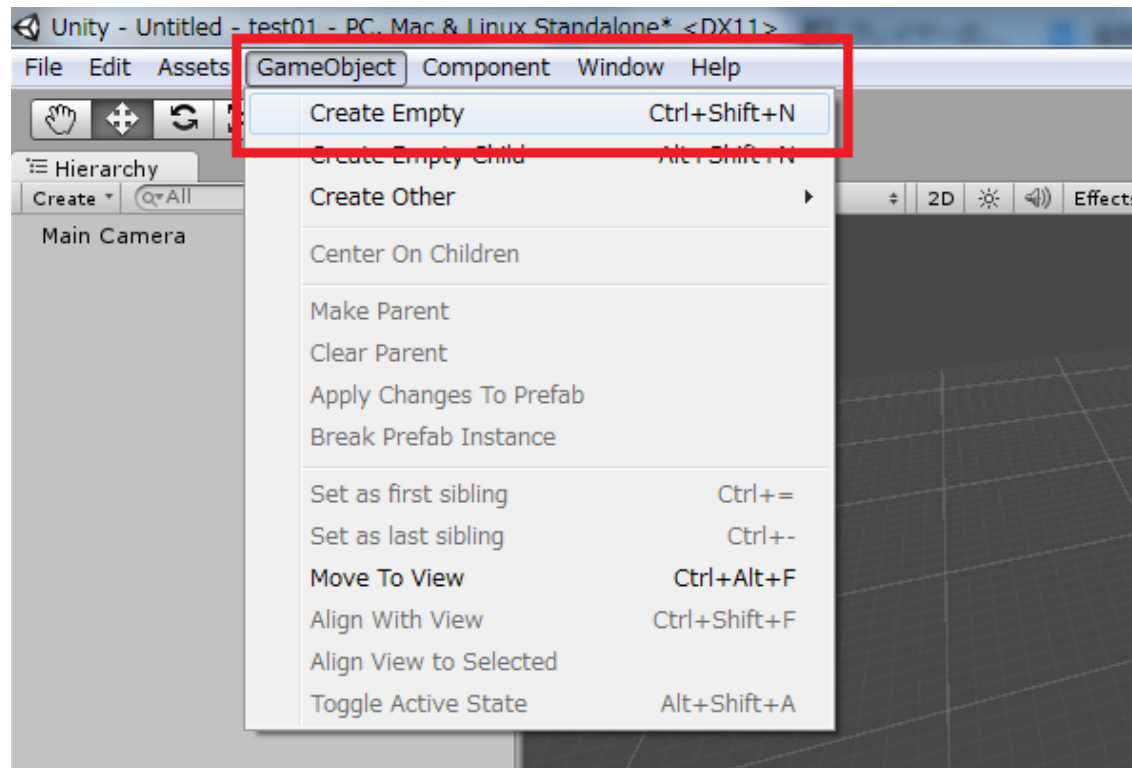
NCMB_latestのUnity package
file「NCMB」をダブルクリック

表示された右の画面の
「Import」をクリック



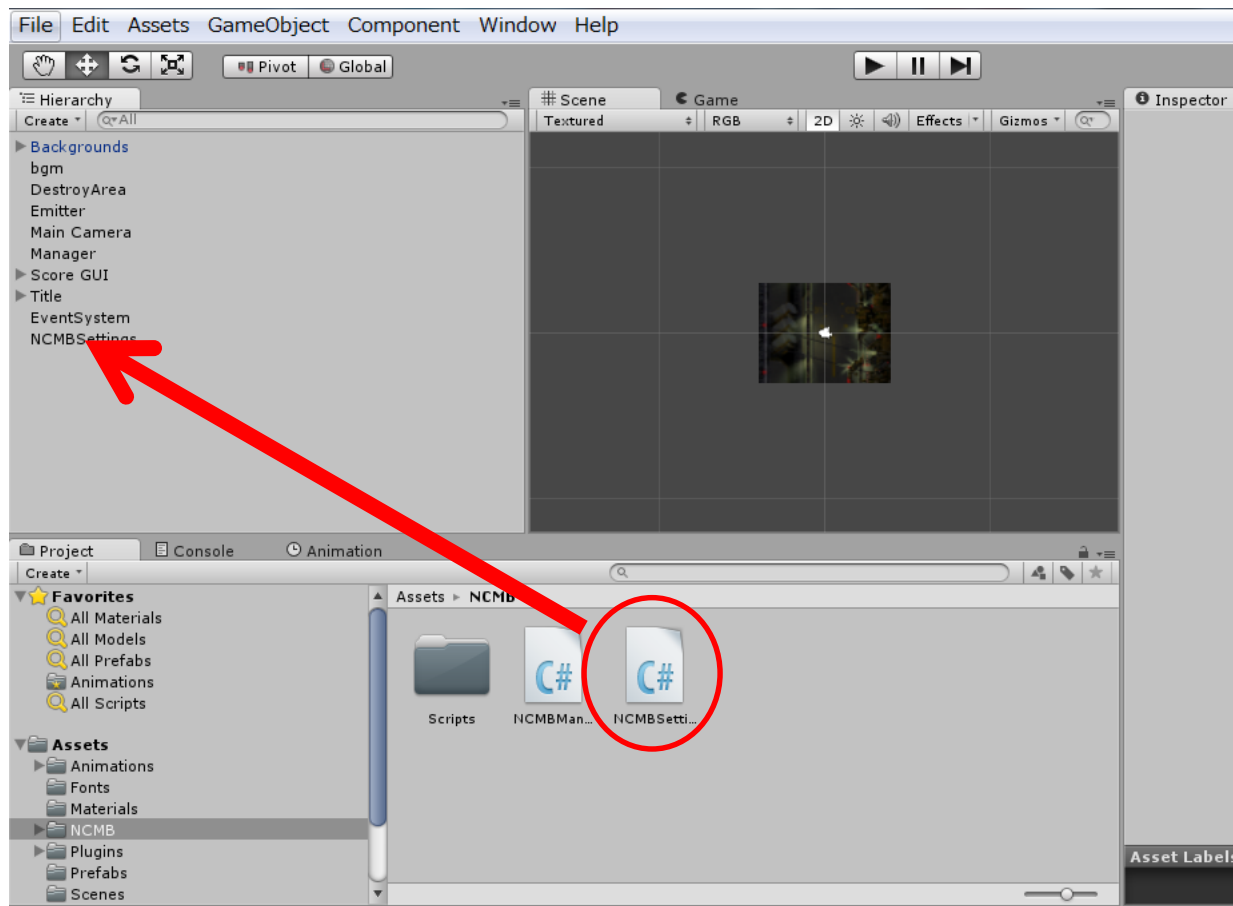
SDKの組み込み 1※こちらはすでにやっております

「Start」シーンを開いて空のGame Objectを作成し
「NCMBSettings」にリネームします



SDKの組み込み 2 ※こちらはすでにやっております

Asset>NCMBの「NCMBSettings.cs」を
「NCMBSettings」にアタッチします



SDKの組み込み 3

アプリケーションキー、クライアントキーを設定します

The image shows a Unity Inspector window on the left and an NCM Settings interface on the right. A red arrow points from the 'Application Key' field in the Unity Inspector to the 'アプリケーションキー' field in the NCM Settings. The 'Application Key' and 'Client Key' fields in the Unity Inspector are circled in red. The 'アプリケーションキー' and 'クライアントキー' fields in the NCM Settings are also circled in red.

Inspector (Left):

- NCMBSettings (Script)
- Application Key
- Client Key
- Use Push
- Android Sender Id
- Response Validation

NCM Settings (Right):

- 2013年9月
- プッシュ通知 0回
- ストレージ 0.00MB
- 残り49,999,921画
- 残り9,999,998画
- 空き102,399.72MB
- アプリケーションキー ? ca57501b099fb9d8a6c2fb32fc41a79fa22858fb514102459a684; コピー
- クライアントキー ? 3863cbc8884b80f047bce327d92ecc5c11a8288c9e1b7f6b5e6b2 コピー

目次

- 1 ハンズオン前の準備
- 2 各アセットのインポート
- 3 ゲームオブジェクトの設定準備
- 4 NCMBの設定
- 5 NCMBデータストアの活用

NCMBデータストアの活用 タスクリスト

1 経過時間に保存するコードの実装

経過時間保存ロジックの実装

Assets > Scripts > UnityChan内にあるスクリプト「Timer.cs」を
参照ください

```
using UnityEngine;
using System.Collections;
using NCMB;
.
. (省略)
.
// Goal到着が検知されたとき
void OnGoal(){
    goal = true;
    //クラスの指定
    NCMBObject timeClass = new NCMBObject("Time");
    //カラムと挿入するデータの指定
    timeClass["time"] = lapTime;
    //非同期でのアップロード
    timeClass.SaveAsync();
}
```

動作確認

- ここまでの作業でサーバーに経過時間を保存できるようになっています。
- 一度ゲームを遊んでみて経過時間を保存してみましょう！

NCMBデータストアの活用 タスクリスト

- 1 経過時間に保存するコードの実装
- 2 保存されたデータをランキングで表示する

今回のハンズオンの発展内容
Assets > Scripts > LeaderBoard
をご参照ください

LeaderBoardフォルダの構成

- LeaderBoardフォルダには以下のスクリプトが含まれています
 - LeaderBoard.cs
 - Rankers.cs
- ランキングは以下のメソッドで行っております
 - Timer.csのOnGUI部分

各スクリプトの解説

- LeaderBoard.cs
 - mobile backendと接続して経過時間を引き出す
- Rankers.cs
 - 引き出した経過時間を一時的に保存するもの

LeaderBoard.csの解説1

```
public void fetchTopRankers(){
```

```
// データストアの「HighScore」クラスから検索
```

時間を保存しているTimeクラスを操作する
クエリの作成

```
NCMBQuery<NCMBObject> query = new NCMBQuery<NCMBObject> ("Time");
```

```
query.OrderByDescending ("time");
```

timeカラムを昇順に並び替えて
引き出すよう設定

```
query.Limit = 5;
```

5個だけ取り出すように設定

LeaderBoard.csの解説2

```
query.FindAsync ((List<NCMBObj> objList ,NCMBException e) => {
```

時間を引き出す操作

```
if (e != null) { //検索失敗時の処理
```

```
} else { //検索成功時の処理
```

```
    List<NCMB.Rankers> list = new List<NCMB.Rankers>();
```

```
    foreach (NCMBObj obj in objList) {
```

引き出したオブジェクトからtimeだけを取り出す

```
        string t = System.Convert.ToString(obj[" time "]);
```

```
        list.Add( new Rankers( t ) );
```

引き出した
時間をRankers
クラスに保存

どうやってLeaderBoard.csのメソッドを呼び出すか？

今回、LeaderBoard.csはUnityに依存しない形で実装しています。
開発状態によってはこのようにUnityに依存しない形で実装することもあるかと思うので
その場合のメソッドの呼び出し方についても記載しておきます

```
LeaderBoard lBoard;
```

```
lBoard = new LeaderBoard(); // LeaderBoard クラスのインスタンス生成
```

```
lBoard.fetchTopRankers(); //メソッドの呼び出し
```

おまけ

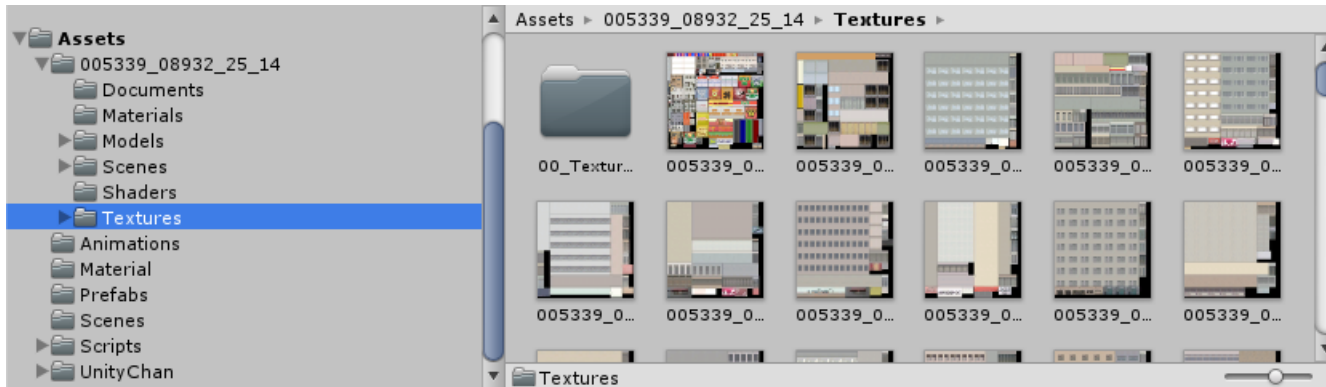
Otaku Cityの設定

現状こんなピンクですが...

005339_08932_25_14>Models内の PQ_Rmake_AKIHABARの
インスペクタ Materials Material Naming を

「From Model's Material」→「Model Name+Model's Material」に変更

その後 005339_08932_25_14>Texturesの00_TexturesPlusをクリックすると
「Unapplied import Settings」というアラートがでるのでApplyすると...



カラーになります

なぜこうなるのか？

まったくわからないので非公式な方法でご認識ください

スマホとMainCameraの傾きの同期

ハコスコで行うことを想定しているのでスマホの傾きとMainCameraの傾きを同期させています同期させているロジックが

Assets>Scripts>UnityChan内のMyCameraCotroller.cs です

MyCameraController.cs Update内

```
Input.gyro.enabled = true;
if (Input.gyro.enabled)
{
    Quaternion gyro = Input.gyro.attitude;
    this.transform.localRotation =
    Quaternion.Euler(90, 0, 0) * (new Quaternion(-gyro.x,-gyro.y, gyro.z, gyro.w));

    //最初に見ていた向きとゲームの進行方向を合わせる
    this.transform.localRotation = Quaternion.Euler(0, -start_gyro.y, 0);
}
```

ジャイロの値を取得する

ジャイロの値とカメラの傾きを同期させる

次のページにて解説

start_gyro.yについて

start_gyroは「startシーン」で「startボタン」をタップしたときに取得しているジャイロの値です。

MyCameraController.csこの値を利用し

```
this.transform.localRotation = Quaternion.Euler(0, -start_gyro.y, 0);
```

としているのは、

ゲームのユーザーが正面だと思いう向きと

ゲームの進行方向を合わせるためです、

もしこの一行がないと、ユーザーの向いている向きと関係なく、ジャイロのy軸の値が0° のときをゲームの進行方向としてしまいます。

ゴールしたことを検知する方法

今回、ゴールしたことを検知するロジックは

Assets>Scripts>UnityChan内のCollisionDetect.csで記述しています

CollisionDetect.cs

```
void OnTriggerStay (Collider col){  
    if(callbackActive == false){  
        return;  
    }  
    //UnityちゃんがGoalについたら各スクリプトのOnGoalを作動させる  
    if(col.name == "Goal"){  
        gameObject.SendMessage("OnGoal");  
        return;  
    }  
}
```

「Goal」objectにUnityChanが接触した際に“OnGoal”というメッセージを発行するようにし、UnityChanにアタッチされている各ロジックのOnGoalメソッドが作動するようにしています。

それにより経過時間のサーバーへの保存やアニメーションの切り替えが行えています

本日はお越しいただき
ありがとうございました！