

【ハンズオンおまけコンテンツ】 VRゲームにゴースト機能を 追加しよう！

※このコンテンツを行う前に、mBaaSデータストアのTimeクラスを削除してください

- ゴースト機能とは？
- ゴースト機能追加の流れ
- ゴーストデータの作成・保存方法
 - プレイヤーの走行ログ生成
 - 走行ログのサーバーへの保存
- ゴーストデータの再生
 - 走行ログの引き出し
 - ゴーストの読み込み
 - 走行ログによるゴーストの操作



これがゴースト

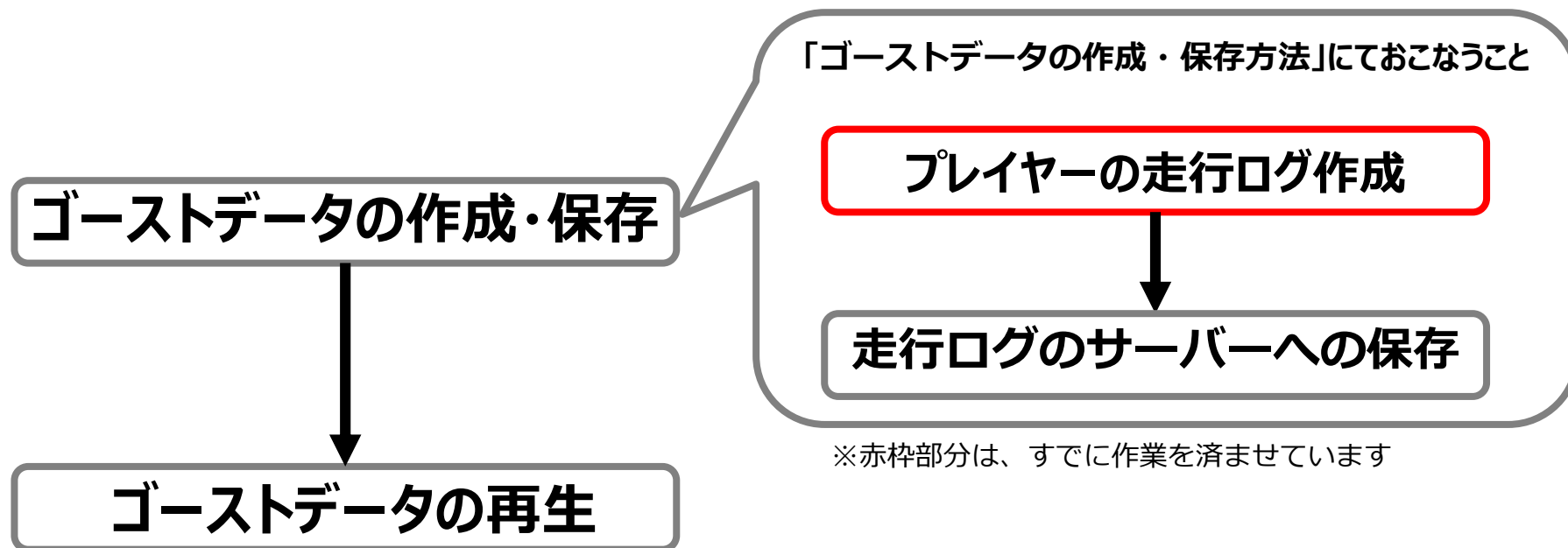


ゴースト機能とは？

レーシングゲームや
シューティングゲームで
上手な人のプレイログを
見ながら自分もプレイできる
機能です。

ユーザーは自分の
プレイ技術を向上させるために
この機能を利用します。

これを先ほど作成したVRゲーム
に導入してしまいましょう。



※今回は本ページの作業を行う必要はありません。

Asset>Scripts>Runnerの「Runner.cs」の下記のコードで走行ログを作成しています

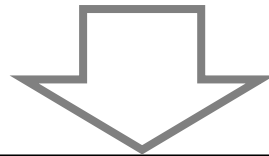
```
//ゴーストデータ生成のため、プレイヤーの現在地のログデータの作成  
float[] postion = new float[2];  
postion [0] = transform.position.x;  
postion [1] = transform.position.z;  
posList.Add(postion);
```

上記のコードで、プレイヤーの現在地の座標を取得配列をリスト化しています。X軸、z軸しか値をとっていないのは、今回のゲームはy軸方向には移動しないからです。

経過時間をサーバーに保存するコードを変更し、このposListも一緒に保存するようにします。

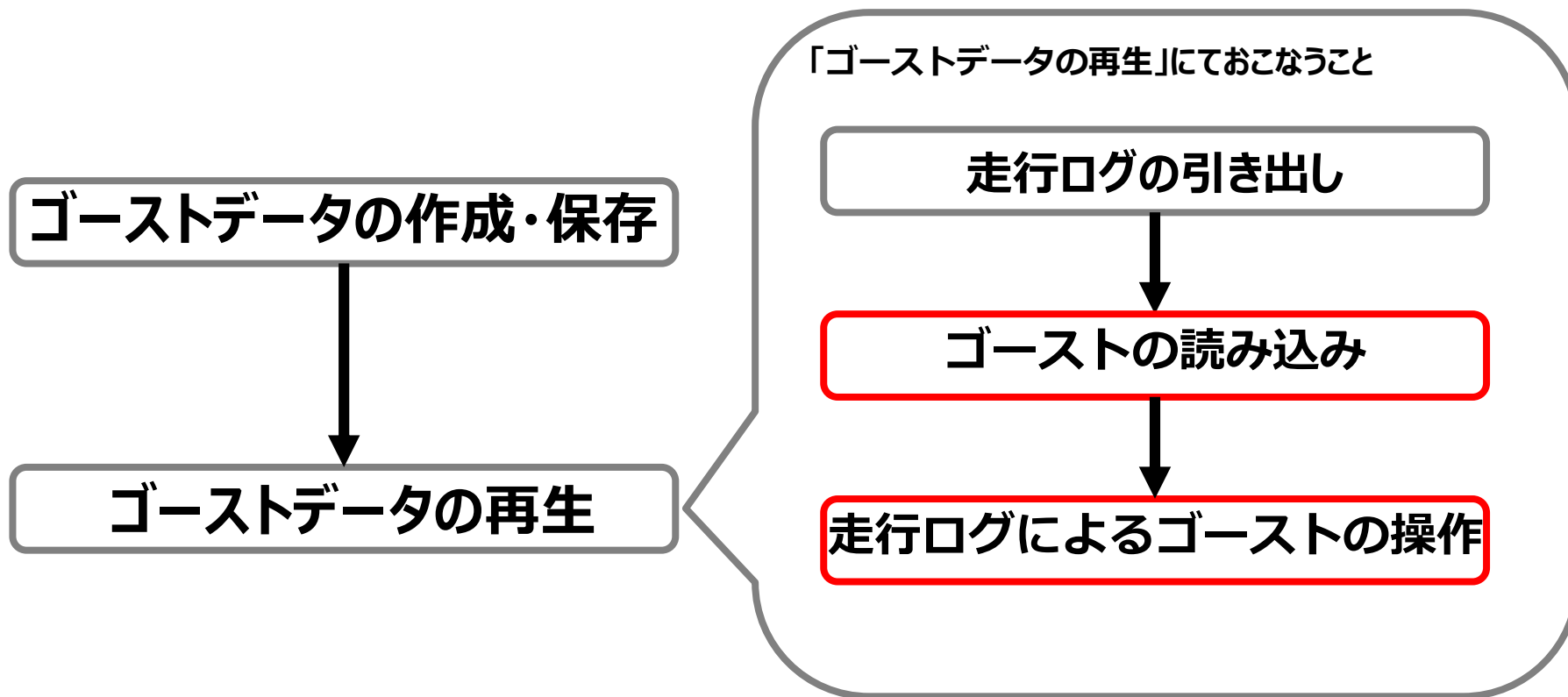
Asset>Scripts>Runnerの「Timer.cs」の下記の部分を変更し
走行ログもサーバーに保存できるようにします。

```
void OnGoal(){  
    goal = true;  
    //クラスの指定  
    NCMBObject timeClass = new NCMBObject("Time");  
    timeClass["time"] = lapTime; //非同期でのアップロード  
    timeClass.SaveAsync();  
}
```



```
void OnGoal(){  
    goal = true;  
    //クラスの指定  
    NCMBObject timeClass = new NCMBObject("Time");  
    timeClass["time"] = lapTime; //非同期でのアップロード  
    timeClass["Log"] = Runner.posList;  
    timeClass.SaveAsync();  
}
```

それでは一回ゲームをプレイしてみましょう！



※赤枠部分は、すでに作業を済ませています

Asset>Scripts>Ghostの「Bg_ghost.cs」のvoid Start(){}の中に下記のコードを実装してください

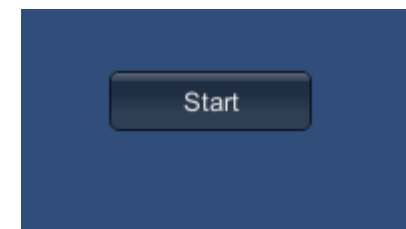
```
NCMBQuery<NCMBObject> query = new NCMBQuery<NCMBObject> ("Time");
query.OrderByAscending ("time");
query.Limit = 1;
query.FindAsync ((List<NCMBObject> objList ,NCMBException e) => {

    if (e != null) {
        //検索失敗時の処理
    } else {
        //検索成功時の処理
        // 取得したレコードをscoreクラスとして保存
        foreach (NCMBObject obj in objList) {
            posObj = obj;
        }
        readyGhost = true;
    }

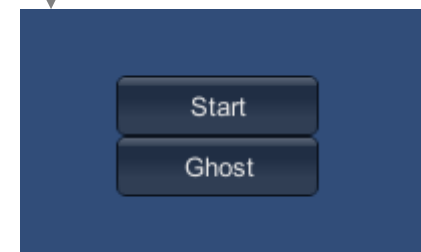
});
```

上記のコードはスタート画面にて駆動するように設定しています。走行ログの取得が完了したら、readyGhostというフラグを変更しスタート画面にて「Ghost」ボタンを表示するようにしています。「Ghost」ボタンを押下するとゴーストが表示されます、

左記コード実装前の
スタート画面



左記コードを
実装すると



左記コード実装後の
スタート画面
ログ取得後に
「Ghost」ボタンが
表示される

※参考資料※ゴーストの読み込み

※今回は本ページの作業を行う必要はありません。

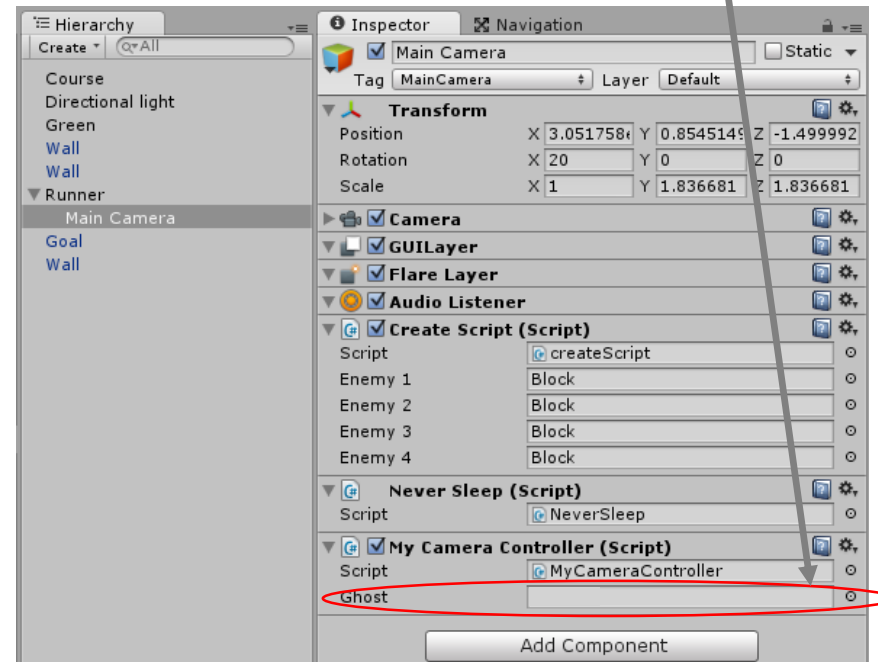
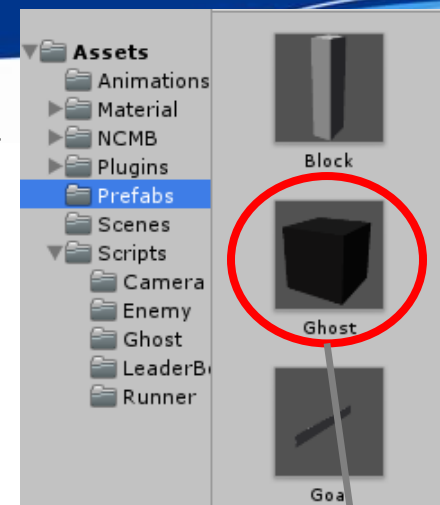
Asset>Prefabsに「Ghost」という、ゴーストとなるオブジェクトを作成しています(左図の赤丸内)。前頁の「Ghost」ボタン、押下時に「run」シーンでこのゴーストオブジェクトを読み込むようにします。

コードとしてはAsset>Scripts>Cameraの「MyCameraController.cs」のvoid Start(){}; 内の下記のコードにて行っています。

```
if (StartCameraController.withGhost == true) {  
    // ゴーストボタンを押下したらゴーストを表示する  
    Instantiate (ghost, ghost.transform.position,ghost.transform.rotation);  
}
```

また、読み込みを行うためにもインスペクターにて設定が必要です。

「MyCameraController.cs」をアタッチしているGameObject「MainCamera」の左図の場所にゴーストオブジェクトをドラック&ドロップしてください。



※参考資料※ 走行ログによるゴーストの操作

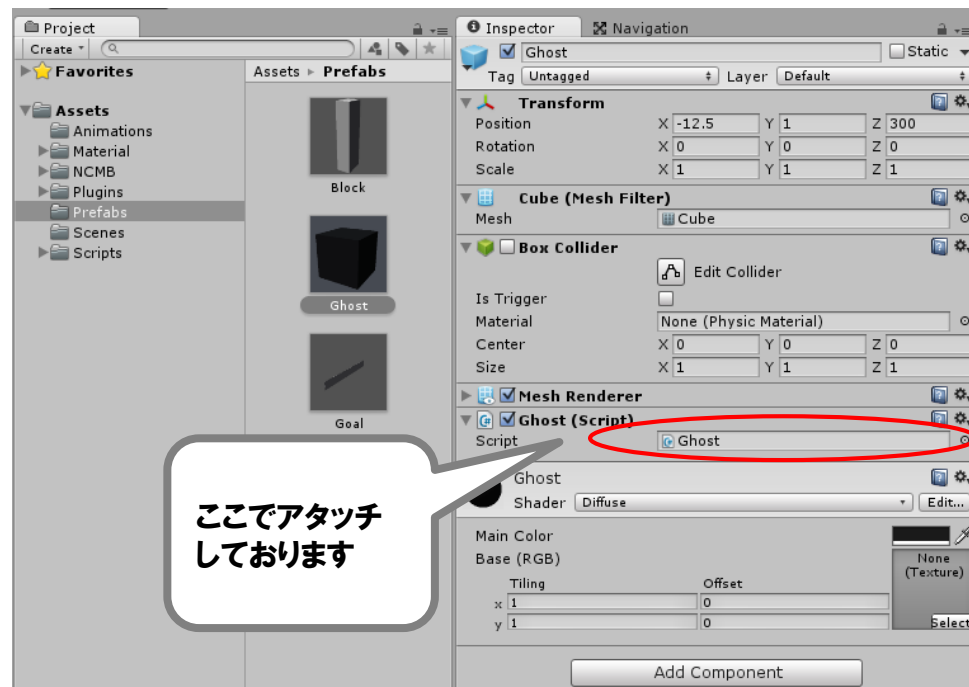
※今回は本ページの作業を行う必要はありません。

Asset>Scripts>Ghostの「Ghost.cs」のvoid Update(){}の中に実装している、下記のコードでゴーストオブジェクトを操作しています。

```
float x =  
    (float)System.Convert.ToDouble(((ArrayList)((ArrayList)Bg_ghost.posObj["Log"])[flameCount])[0]);  
float z =  
    (float)System.Convert.ToDouble(((ArrayList)((ArrayList)Bg_ghost.posObj["Log"])[flameCount])[1]);  
transform.position = new Vector3 (x,0.56f,z);  
flameCount ++;
```

上記の「Ghost.cs」は前頁のゴーストオブジェクトにアタッチされています。
左図のようにご確認ください。

**「Ghost」ボタンを押して
ゲームをスタートしてみてください！
Ghostが表示されます。**



@nifty



ニフティとなら、きっとかなう。
With Us, **You Can.**

NIFTY Cloud
ニフティクラウド